preview

بزرگترین مرجع کتابهای الکترونیکی فارسی وانگلیسی
بزرگترین مرجع نرم افزارهای کاربردی و تخصصی
بزرگترین مرجع دانلود کلیپهای موبایل

www.IranMeet.com

INFO@IRANMEET.COM

آشنای با Java و J2EE

تاليف و ترجمه: مهدي وجداني

mehdi_vojdani@yahoo.com

اين كتاب در ابتدا براي چاپ در نظر گرفته شده بود اما با وجود زياده خواهي ناشران از نويسنده گان جوان به صورت مجاني در اختيار علاقه مندان قرار ميگيرد. كليه حقوق اين كتاب محفوظ است.

به نام خدا

عنوان فصل **صفحه**

فصل اول

مقايسه .NET و J2EE

مقدمه
در ابتدا قرار بر وجود اين فصل نبوده ولي بنده آنرا بدون تغيير در اين جا قرار دادم.

دات نت : يعني يك بستر براي ايجاد نرم افزار و توسط مايكروسافت ايجاد شده اما تمام دات نت چيزي بيشتر از يك "بستر اجراي كد و كتابخانه مقدماتي كلاس" نيست . تمام دات نت يك FrameWork است و يك محيط توليد نرم افزار كه برپايه استفاده از DLL و مستنداتي جهت ايجاد ابزارهاي متفرقه توليد نرم افزار و فقط روی ويندوز كار كرده .

J2EE يك استاندارد است كه براي پاسخ دادن به يك نياز نرم افزاري " سازمان مقياس" چگونه بايد با اجزاء نرم افزار رفتار كرد و براي مديريت طول عمر نرم افزار (Application Lifecycle Management) چكار بايد كرد . اين استاندارد توسط سان ارائه شده . خيلي ها مبتني بر اين استاندارد نرم افزارهاي خودكار سازي ايجاد كرده اند ، سان مايكروسيستمز نيز هم . استاندارد J2EE كه چگونه با "زبان جاوا" يك FrameWork ايجاد كنيم ، چگونه كتابخانهء كلاس براي تمام مقاصد بنويسيم ، بانك اطلاعاتي چطور باشد و ... و شركتهاي متعددي بر اساس اين استاندارد Application Server هاي مبتني بر J2EE ايجاد كرده اند كه برخي شون تمام J2EE رو پياده سازي كرده اند و برخي ها فقط بخشي از آن را
Application Server يك زير ساخت نرم افزاري است كه وظيفه اش Application Lifecycle Management است . يعني از ابتداي تولد يك نرم افزار "سازمان مقياس" تا انتهاي اتمام توليد ، بايد به تمام نيازهاي نرم افزاري پاسخ دهد . يعني اگر برنامه نويس به يك FrameWork احتياج داشت ، Application Server يك Framework به او بدهد ، اگر بانك اطلاعاتي خواست ، Application Server يك بانك اطلاعاتي كامل برايش فراهم كند ، اگر وب سرور خواست ، Application Server يك وب سرور تمام عيار به او بدهد ، اگر براي ارسال نامه هاي الكترونيكي برنامه اش به يك SMTP سرور نياز داشت ، Application Server يك سرور SMTP به او بدهد ، اگر قرار شد براي احراز هويت از Kerberos استفاده كند يك پياده سازي كربرايزد از استك TCP/IP در Application server وجود داشته باشد ، اگر خواست داده هاي كاربري رو از كارتهاي هوشمند (smart Card) دريافت كند ، رابطهاي لازم و API هاي مربوطه را از Application Server بگيرد و ... به ديگر بيان Application Server يك محيط Integrated است براي طراحي و توليد و مديريت و توزيع و كاربرد يك نرم افزار "سازمان مقياس" .
فرض كنيد اگر قرار شد يك گروه نرم افزاري براي بزرگترين سازمان بيمه غير دولتي امريكا يك راهكار جامع ERP توليد كند (يا بخرد و خصوصي سازي كند براي محيطش) بايد به چه بستري اعتماد كنه كه مطمئن باشه تمام درخواستهاي نرم افزاري " سازمان مقياس" ش رو ميتونه جواب بده و مشخصه هاي اون ، سازمانش رو به يك نرم افزار خاص ، سخت افزار خاص ، پروتكل خاص و ... محدود نميكنه ؟ (اصولا" محدوديت در ادبيات آي تي ، سطح اعتماد و قابليت وثوق - Reliability - رو كاهش ميده) اينجاست كه يك Application Server لازم ميشود . يك Application Server تضمين ميكنه كه از بستر اجراي كد گرفته تا وب سرور ، از توابع امنيتي گرفته تا بانك اطلاعاتي ، از IDE گرفته تا ابزارهاي حمايت از UP (يا Unified Process) و ... در بستهء نرم افزاريش وجود دارد .

طبيعي است که زبان جاوا زبان استاندارد توسعه نرم افزارهاي مبتني بر J2EE باشه ، هر چند بر خلاف اظهارات ناشيانه برخي ، J2EE و خصوصا" بستر اجراي کدش ، به زبان جاوا منحصر نيست . يعني همونطور که [مثلا"] بستر دات نت قابليت پذيرش زبانهاي مختلف رو داره ، بسترهاي مبتني بر جاوا هم ميتونن به ساير زبانهاي برنامه نويسي سرويس بده . يعني براحتي ميشه بين جاوا و ساير کتابخانه هائي که توسط ساير زبانهاي برنامه نويسي توليد شده ارتباط برقرار کرد . (Java Native Interface) هر چند که مثل دات نت منعطف نيست .شرکت سان تلاش ميکند يك Application Server مبتني بر استاندارد خودش يعني J2EE توليد کند اما هنوز تکميل نشده . (سان فعاليت گسترده اي براي توسعه خود جاوا و بهينه سازي منطق J2EE و کلاسهاي توليد نرم افزار دارد.

در ايران اغلب اوراکل رو به عنوان يك بانك اطلاعاتي ميشناسن در حاليکه بانك اطلاعاتي اوراکل فقط بخشي از اون چيزي است که اوراکل تحت عنوان e-Bussines Suite منتشر کرده . Application Server اوراکل تمام اون چيزهائي که در وصف يك Application Server عرض کردم داره . بطور مختصر و ليست وار در موردش توضيحاتي عرض ميکنم تا کمي روشنتر بشه بحث :

۱. يك بانك اطلاعاتي کامل : اوراکل فعلا" تنها بانك اطلاعاتي است که نه تنها نسخه هاي متعددي براي MainFrame ها داره ، براي تمام بسترهاي نرم افزاري و سخت افزاري موجود هم نسخه هائي رو ارائه کرده . بزرگترين بانك اطلاعاتي که اين حقير در جريانش هستم و با اوراکل کار ميکنه بانك اطلاعاتي وزارت انرژي ايالات متحده آمريکاست که روي يك MainFrame شرکت IBM اجرا شده . بانك اوراکل يك نسخهء کامل زبان پرس و جوي ساخت يافته يعني PL/SQL ، يك سوئيت کامل بنام PSP که براي توليد صفحات وب بطور مستقيم از PL/SQL استفاده ميکنه است . اوراکل تنها بانك اطلاعاتي است که موتور آن (DB Engine) هم ميتونه بصورت توزيع شده و چند بخشي (Clustered) اجرا بشه . حتي ميشه بخضي از انجين رو روي يك بستر کوچك وينتل(ويندوز + اينتل) و بخشي ديگر رو روي يك ماشين غول پيکر HP مجهز به HP-UX اجرا کرد . حتي ميشه حين سرويس دهي بانك ، بانك رو از يك پلت فرم به پلت فرم ديگه منتقل کرد . (ويژگي هاي منحصر به فردش رو عرض کردم)

۲. يك بستر اجراي کد نرم افزار : Application Server شرکت اوراکل بطور کامل "بخش نرم افزاري J2EE " يعني کتابخانه هاي کلاسش رو پياده سازي کرده .

JDBC Connectors
JSP Engine
JavaBeans Engine
RMI
JMS
JINI
JMX
JIRO
J2EE CORBA ORB
JXTA
JXML
JCP
JNI
Web Service Implementation
و ...

يعني هر کسي هر برنامه اي مبتني بر J2EE نوشته باشه در بستر Application Server اوراکل قابل اجرا و سرويس دهي است . اوراکل J2SE و J2ME رو هم حمايت ميکنه (دومي براي توليد برنامه هاي موبايل براي پورتابل ديوايسها کاربرد داره) . همچنين اوراکل بطور کامل

يك نسخه از Java Smart Card API رو پياد سازي كرده . در حال حاضر جاوا تنها ابزاري است كه ميشه توسط اون تقريبا" براي تمام كارتهاي هوشمند برنامه نوشت ضمن اينكه توسط قابليت فوق الذكر قطعه كدهاي قابل ذخيره سازي در كارتهاي هوشمند هم قابل توليد است . فرض كنيد يك تابع تبديل تاريخ مينويسيد و تابع رو داخل كارت هوشمند قرار ميديد ، هر وقت نرم افزار اون تابع رو صدا زد كارت رو در كارت خوان ميگذاريد و برنامه شما تابع رو روي كارت هوشمند صدا ميزنه و جواب ميگيره بدون اينكه در مورد پياده سازي اش چيزي بدونه .

۳. اوراكل يك وب سرور مخصوص به خود ، همچنين سرورهاي :

POP3
SMTP
FTP
WebDav
Cache Server
Common Internet File system - CIFS
LDAP compatible Directory Service
و...

رو بطور كامل پياده سازي كرده . تمام اين سرورهاي نرم افزاري كاملا" با هم سازگار هستند و براي كار روي يك محيط مبتني بر J2EE بهينه سازي و خصوصي سازي شدن .

۴. اوراكل يك content Management System داره كه قابليت ايجاد پورتال هاي مبتني بر وب روي اينترنت يا اينترانت رو به "نرم افزار" هاي J2EE ميده .
۶. اوراكل يك سرويس (يعني نرم افزارهائي + سرويس دهنده هائي) براي ايجاد ويژوال گزارش از بانك اطلاعاتي داره . گزارشها ميتونن طراحي بشن تا از داده ها استفاده كنن و خروجي بدن ، يا يك سرويس توليد گزارش به يك نرم افزار متصل بشه تا در زمان اجرا مولفه هاي گزارش به سرويس گزارش درخواست داده بشن تا گزارش رو طراحي كنه ، به داده متصل كنه و خروجي بده . بهش ميگن Reporting Service

۷. ابزارهاي مديريتي قدرتمند براي كنترل تراكنشهاي بانك اطلاعاتي خارج از محيط بانك (منحصر به فرد) ، كنترل وضعيت اشياء مثلا" EJB ها و سطح دسترسي آنها ، انتقال سرويسهاي از يك پلت فرم به پلت فرم ديگر بدون توقف روند سرويس دهي ، صف گذاري منطقي و مديريت شده درخواستها و ...

۸. Load Balancer اوراكل كمك ميكنه سرويسها ، بانكهاي اطلاعاتي و سرورها و ساير نرم افزارهاي مبتني بر وب يا شبكه روي يك بستر توزيع شده اجرا بشن و اگر فشار ترافيك روي يك سرور زياد بود ، Load Balancer درخواستها رو به ساير سرورها كه توسط قابليت Replication Service اوراكل بصورت mirror آماده هستند هدايت ميكنه . اين Load Balancer قابليت درك جلسات كاربري (Session) ها يا مثلا" متغيرهاي سطح برنامه (Application - Level Variables) رو داره . يعني اگر شما به كتابخانهء ملي سناي امريكا (Powered By Sun) لاگ اين كنيد و در حال انتقال صفحاتي از يك كتاب به دايركتوري شخصي خودتون باشيد و فشار روي سرور بانك اطلاعاتي زياد بشه ، درخواستهاي بعدي شما بصورت خودكار به سرور خلوت تري ارسال ميشن بدون اينكه State-Less بدون محيط به كانال ارتباطي شما لطمه بزنه ، يعني هويت شما و Session شما همچنان معتبر است اما روي يك سرور ديگر (اين منحصر به فرد نيست اما فقط شركت مكرومديا در JRUN كه اون هم يك Application Server نصفه نيمه است چنين چيزي داره كه در مورد اون هم مطالبي عرض ميكنم)

نتيجه اول : اگر شما يك Application Server كامل و قابل اتكاء ميخواهيد بايد بستهء نرم افزاري فوق العاده گران قيمت Oracle 11i - e bussines Suite رو تهيه كنيد كه هر انچه ذكر شد داخلش موجوده .

نتيجه دوم : براي پاسخ دادن به نيازهاي "خيلي بزرگ" و " سازمان مقياس" كه هزينه هاي ميليوني و ميلياردي براش در نظر گرفته ميشه و خطا در انتخاب معماري يا عدم سازگاري اجزاء نا بخشودني است فقط بايد از يك Application Server استفاده كرد كه ضمن دارا بودن همه چيز يكجا ، سازگاري ، قابليت اتكاء و وثوق ، يك شركت بزرگ و عريض و طويل ازش حمايت كنه . اوراكل يكي از پيشنهادهاي موجوده . در بازار E- bussines دنيا اوراكل قدرتمند ترين فروشنده نرم افزاره ، فقط به عنوان مثال مراجعه كنيد به خبري از اوراكل كه يك پيشنهاد " نه ميليارد دلاري" براي خريد شركت People Soft (توليد كنندهء ERP هاي قدرتمند) داده . لازم به ذكره ٩ ميليارد دلار ، كمي كمتر از نصف بودجهء يكسال جمهوري اسلامي ايران است !!

ياد آوردي اول : يك بستر اجراي كد مثل دات نت ، با يك استاندارد براي توليد Application Server يعني J2EE قابل قياس نيست .

الف. كتابخانه كلاس : دات نت و JDK هر دو كتابخانه هاي قدرتمندي هستند كه اغلب نيازهاي پايه براي توليد نرم افزار رو حمايت ميكنند . اما براي توليد نرم افزارهاي بزرگ مقياس هيچكدام كافي نيستند . دات نت چيز ديگري ندارد اما براي جاوا راهكارهاي ديگري هم وجود دارد . كاربري دات نت راحت تر است . پيچيدگي هاي دات نت هم كمتر است . نمودار يادگيري جاوا بسيار كم شيب است . (اگر نمودار عمودي پيشرفت باشه و نمودار افقي زمان) در حاليكه يادگيري دات نت خيلي سريعتر است .

ب. زمان اجرا : زمان اجراي دات نت تقليدي صرف از زمان اجراي جاوا ست . هيچ بحثي هم درش نيست . حتي كسانيكه مثل بنده عقلشون كم باشه و بشينن و IL رو با ByteCode مقايسه كنن درك خواهند كرد كه مايكروسافت خلاقيتي از خودش نشون نداده . JIT در هر دو محيط خوب است . سرعت اجراي "برنامه" هاي دات نت از برنامه هاي جاوا كندتر است اگر از JIT استفاده نكند . اين حقيقت رو هر كسي با چند آزمايش كوچولو ميتونه درك كنه . سيستم Code Caching و JITC دات نت كمك زيادي به افزايش سرعت برنامه ها كرده . جاوا با عمر طولاني اش به ادعاي اسكات مك نلي حدود پنجاه بار بهينه سازي شده در حاليكه دات نت هنوز جوونه . به نظر ميياد در اين يك مقوله بايد منتظر آينده شد . اما في الحال وضع دات نت در اين راستا خوبه .

ج. اتصالات : دات نت از ريموتينگ ، وب سرويس و كام پلاس حمايت ميكنه (بصورت داخلي) . جاوا بجاي ريموتينگ چيزي بنام ريموت متد اينووكيشن داره ، وب سرويس رو حمايت ميكنه ، CORBA رو حمايت ميكنه ، چيزي بنام EJB داره كه اشياء شناور در يك "مخزن سازماني" هستند كه افراد ، سرويسها و نرم افزارها بنا به ميزان دسترسي ميتونن ازش استفاده كنن . كنترلهاي دات نت هنوز چنين قابليتي ندارند و دات نت هنوز راهي براي ايجاد يك Object Repository سازماني ارائه نكرده . اشياء كام پلاس و محيط MTS ويندوز هم (با اينكه ربطي به دات نت نداره بطور مستقيم) مانند EJB ها منعطف نيستند . EJB ها State-Less نيستند .

د. ارتباط با داده : دات نت چيزي بنام ADO .NET ارائه كرده كه راه حلي است منحصر به فرد . جاوا JDBC رو داره كه چه در connection Pooling و چه در objecy pooling به خوبي ADO .NET. كار ميكنه اما ADO .NET فوق العاده امكانات زيادي داره . چون اينجا دات نت كار زياده لزومي به توضيح نيست . من با تمام وجود به ADO .NET اعتقاد دارم و تصور نميكنم معادلي داشته باشه (يكسال و خورده اي پيش چند مقاله كامل در مورد ADO .NET در سايت ايران دولوپرز نوشتم كه اين مطلب رو اونجا هم عرض كردم . يكسال قبل)

ه. امنيت : امنيت در اين حوزه رو "من" به سه بخش تقسيم ميكنم (تقسيم بندي كاملا شخصي و تجربي)

ه.١) حفاظت از متن كد
ه.٢) حفاظت از ارتباطات

ه.٣) حفاظت از خود بستر و حفظ مانائي

در مورد اول هر دو محيط ضعيف هستند . Obfuscator ها نميتونن به مفهوم واقعي از كد حفاظت كنند و راهكارهاي Third party موجود هم بيشتر به طنز شبيه هستند . با داشتن IL يا بايت كد براحتي كد اصلي يا كدي " با قابليت هاي" كد اصلي قابل باز-توليد است . پسوردها ، اعداد خصوصي ، كلمه هاي عبور و ... براحتي قابل بازيافت هستند اگر در متن نرم افزارهاي Managed دات نت يا برنامه ها جاوا بكار رفته باشند . اينجا واقعا هيچ ايمني " نميتواند" وجود داشته باشد .

در مورد دوم هر دو محيط با Open Standard ها كار ميكنند . از SSL گرفته تا Kerberos و از ارتباط با Directory Service ها گرفته تا CA . در اين مورد تفاوتي وجود نداره .

در مورد سوم تا حالا مستندي كه بر قوت يا ضعف يكي دلالت كنه نخوندم (نميگم نيست ، نخوندم) و تجربه شخصي و عملي هم ندارم .

في المجموع در حوزه امنيت دو محيط چندان متفاوت نيستند .

و. انتقال : جاوا از MainFrame ها تا كارتهاي هوشمند رو حمايت ميكنه . دات قراره بزودي بسترهاي ديگه رو حمايت كنه . پس اصولا" در اين زمينه هيچ رقابتي وجود نداره . جاوا پانزده سال جلو تره . من با مونو (كه قراره بشه دات نت روي لينوكس) كار كردم و فعلا" ناقص و غير قابل اعتماده . مايكروسافت هم يقينا" تا انتهاي ٢٠٠٥ هيچ نسخه اي از دات نت مبتني بر NIX* ها توزيع نخواهد كرد .

ز. توليد محتواي وب : دات نت ASP .NET رو ارائه كرده . جاوا JSP رو . سرعت پاسخگوئي دات نت در كاربردهاي معمولي بالاتره . اما با توجه به محدوديت ويندوز (به عنوان تنها بستر دت نت) براي حمايت از ترافيك و فشار بالا ، اگر كاربردهاي خيلي سنگين مد نظر باشه ASP .NET نميتونه حرفي داشته باشه . موتور JSP هم قابليت Clustring داره و ميشه مجموعه اي از سرورها رو با " يك موتور" راه اندازي كرد . (ميدونم به بحث ربطي نداره اما يكبار يكي ازم پرسيد چرا مايكروسافت براي MSN و هات ميل از فري بي اس دي استفاده ميكنه ؟ و نه ويندوز ؟ جواب بنده اين بود دليل هر چيزي هست ربطي به امنيت نداره . سايت خود مايكروسافت با ترافيك بالا و دشمناني قسم خورده بدون مشكل داره روي ويندوز كار ميكنه . اما وقتي قرار باشه بخاطر ترافيك خيلي بالاي مسنجر و ايميل ، از يك ماشين با مثلا" ٣٠ تا پردازنده استفاده بشه تجربه ويندوز چندان موفقيت آميز نيست ! در حاليكه فري بي اس دي - اچ پي يو ايكس و سولاريس همين حالا روي ماشينهائي بيشتر از پنجاه پردازنده هم خوب كار ميكنند . سان سرور بنام K Fire١٥ داره - ر.ك بخش سرورهاي سايت سان - به قيمت " ده ميليون دلار" ميفروشتش و ١٠٥ تا پردازنده ۶۴ بيتي داره و همين نسخهء سولاريس معمولي روي اون هم كار ميكنه و جواب ميده و توانائيش ۶۵٠٠ ميپسه ! يعني ٢٠٠٠ ميپس قوي تر از بزرگترين مين فريمه IBM - تاريخ اين امار متعلق به يكسال پيشه كه من پروژه اي داشتم در اين مورد)

نتيجه : براي كاربردهاي عمومي وب يعني اونچيزي كه در ٩٩ درصو اوقات مد نظره ASP .NET بهتر است مگر اينكه برنامهء خاصي براي انتقال وجود داشته باشه يا احتمال وجودش قابل تامل باشه .

نتيجهء كلي : تا اون حد كه دات نت امكانات و توانائي داره ، قالبليتهاي مشابهش در بستر جاوا موجوده . در برخي موارد دات نت و در برخي ديگر جاوا برتر است الا اينكه اگر كاربرد خيلي بزرگ باشه يا برنامه خاصي براي انتقال بستر وجود داشته باشه يا احتمال وجودش قوي باشه ، در هر حال " تنها گزينهء موجود" جاوا ست ، در غير اين صورت بايد بررسي كرد .

< اتمام بحث مقايسه بسترهاي دات نت و جاوا

سوال : من متوجه شدم دات نت دقيقا" چيه و جاش كجاست و متوجه شدم يك Application Server چيه و به چه دردي ميخوره و باز هم متوجه شدم فرق اينها در "مقياس" پروژه است ، حالا ميخوام كمي درمورد Application Server هاي ديگه بدونم .

جواب : اينترنت دريائي از اطلاعات است كه ميتونيد ازش كمك بگيريد . تجربه شخصي من به استفاده از اوراكل و اوريون محدوده . در مورد JRUN هم مطالعه كردم . اوريون يك Application Server مبتني بر J2EE است اما براي محيط لينوكس بهينه سازي شده است . در يكي از شركتهاي نفتي ايراني هم داره ازش استفاده ميشه و فوق العاده جوابگوست . اما مثل اوراكل كامل نيست ، مثلا" بانك اطلاعاتي نداره ، بايد از چيزي مثل اوراكل يا ماي اسكوئل استفاده كرد ، و نواقصي از اين دست اما مجاني است و سورس آزاد . www.orionserver.com .
JRUN محصول مكرومدياست . اين هم ناقصه و خيلي از قابليتهاي اوراكل رو نداره (اوراكل خيلي خيلي گرونه) اما برخي مزاياي خاصش باعث ميشه آدم به انتخابش فكر كنه . مثلا" قابليت كلاسترينگ و لود بالانسينگ داره يا مثلا" ColdFusion رو حمايت ميكنه و ... قيمتش هم ارزونه . مثلا" همين حالا سازمان ملي علوم و تحقيقات و تكنولوژي امريكا يعني NIST داره از جي ران استفاده ميكنه و زبان برنامه هاي وب اش ، خصوصا بخش امنيتي اش كه زياد كل كل ميكنه هم كلد فيوژن است . اپليكيشن سرورهاي ديگه اي وجود دارن كه چندان معروف نيستند . مثلا" Borland Application Server كه اين مورد هم مبتني بر J2EE است و بخشي از سايت خود بورلند هم روي همين كار ميكنه . اپليكيشين سرور بورلند از اوريون و جي ران كاملتره هر چند هنوز هم از اوراكل عقب تره . دپارتمان نرم افزار "ارتش امريكا" هم بطور كامل از محصولات بورلند استفاده ميكنه . براي توسعه نرم افزارهاي ويندوزي از دلفي ، براي UP از توگدر و براي ارائه سرويس از اپليكيشن سرور بورلند . (وايت پيپر هاش رو ميتونيد تو سايت خود بورلند پيدا كنيد) و ... موارد متعدد ديگه .

جور اول : (تجربه شخصي)

كار با جاوا يا در مقياسهاي بزرگ Application Server ها جاوا واقعا" سخته . (خصوصا" اگر آدم به محيطهاي قدرتمند و راحتي مثل دلفي عادت كرده باشه) در حاليكه كار با دات نت واقعا" راحته . كاربري جاوا هم مشكل تر از دات نته . در محيط دات نت اغلب تنظيمات يا وظايف كليك اند ران هستند در حاليكه براي آماده سازي يك محيط مبتني بر جاوا براي ارائه واقعي سرويس تخصص و تجربه لازمه و هميشه تجربه هاي اوليه با شكست همراه هستند . دات نت گرون نيست هر چند اگر واقع بين باشيم مجاني هم نيست . جاوا مجاني است و سورس آزاد . اون چيزي كه من بهش فكر ميكنم اينه كه براي كاربردهاي كوچك ، معمولي ، متوسط دات نت مناسبه . براي كاربردهاي واقعا" بزرگ دات نت اصولا" جوابگو نيست كه بخاد مناسب باشه يا نباشه و جاوا تنها گزينه است حالا ميخواد خوب باشه ميخواد بد باشه . يعني اگر قرار باشه سازماني تيم نرم افزار تشكيل بده ، يك الگو و راه حل جامع (Total Solution) براي نرم افزار انتخاب كنه ، براي برنامه نويسهاش پول خرج كنه و پول بيشتري خرج كنه تا بمونن ، قرار نيست برنامه هاش خيلي خيلي بزرگ باشن ، دات نت گزينهء خوبيه . اگر سازماني قراره تيم نرم افزار داشته باشه و برنامه هاي فوق العاده بزرگ بنويسه كه با توجه به نوع كاربرد احتمال تغيير پلت فرم يا خريد ماشينهاي بزرگتر و قوي تر و تغيير پردازنده و .... محتمل باشه ، اون محيط مال جاواست . اميدوارم در تمام متن مطلبم به عبارت " سازمان مقياس" كه برجسته تر بود دقت كرده باشيد .

فصل دوم

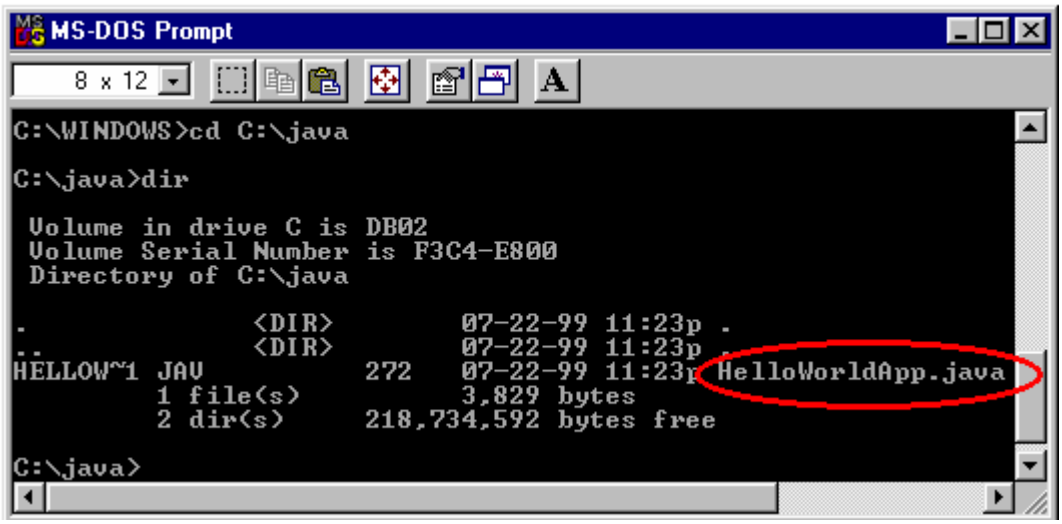
اشنای با Java وJ2EE و نحوه اجرای برنامه ها

نحوه اجراي برنامه هاي جاوا

براي اجراي اولين برنامه سه گام زير را بايد انجام دهيد
- در اولين قدم براي اجراي برنامه هاي خود نياز به نصب Java™ 2 SDK(software Development Kit), Standard Edition داريد. اگر اين برنامه موجود نيست آنرا از آدرس http://java.sun.com/j2se/ دانلود كنيد. اگر از IDEهاي موجود استفاده مي كنيد، آنها به طور خودكار عمل نصب و اجرا را انجام خواهند داد.
- سپس برنامه نوشته شده را در فايل با پسوند java. ذخيره كنيد. بطور مثال HelloWorldApp.java

```
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World!"
        System.out.println("Hello World!");
    }
}
```

- براي كامپايل كردن پنجره Command Prompt را باز كرده و دستور cd c:\java را وارد كنيد. اگر در يك درايو ديگر SDK را نصب كرده ايد به جاي درايو C از آن استفاده كنيد.



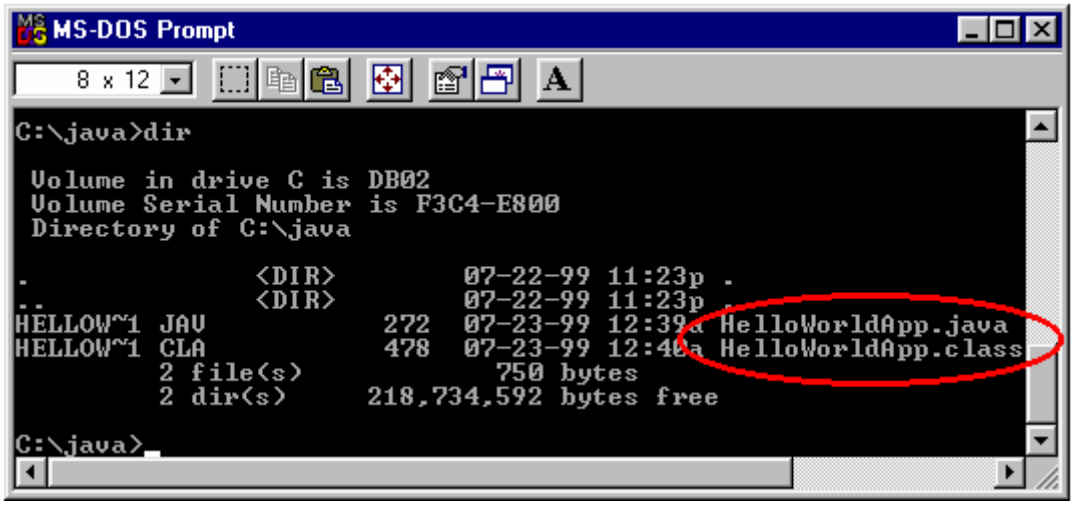
به منظور کامپایل دستور زیر را وارد کند.

Javac HelloWorldApp.java

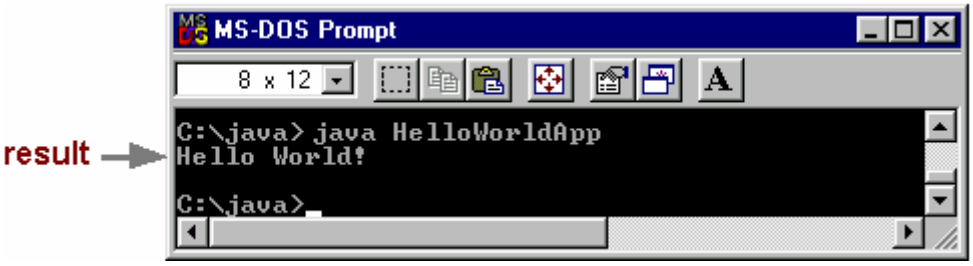
اگر پيغام خطا مبني بر عدم پيدا كردن فايل javacدريافت كرديد يك راه براي پيدا كردن javac اين است كه اگر Java 2 Software Development Kit در آدرس c:\jdk1.4 باشد دستور زير را وارد كنيد.

c:\jdk1.4\bin\javac HelloWorldApp.java

پس از اينكه كامپايل بدون خطا انجام شد يك فايل با پسوند class. به همان فولدر اضافه خواهد شد. اين فايل همان فايل بايت كد است كه در مفسر جاوا اجرا مي شود.



- براي اجراي برنامه در همان فولدر دستور java HelloWorldApp را وارد كنيد وخروجي را به شكل زير خواهيد ديد.



اولين برنامه به زبان جاوا:

در زير اولين برنامه به زبان جاوا را كه پيغام HelloWorld را نشان ميدهد مشاهده مي كنيد

```
public class HelloWorld {
// A program to display the message
// "Hello World!" on standard output
public static void main(String[] args) {
System.out.println("Hello World!");
}
} // end of class HelloWorld
```

تمامي برنامه هاي جاوا در داخل كلاس خود اجرا مي شوند. اولين خط برنامه بالا نام كلاس را كه HelloWorld ميباشد بيان ميكند. اين نكته مهم را بايد در خاطر بسپاريم كه نام برنامه جاوا نام آن كلاس مي باشد و در واقع نام فايل برنامه را كه نيز براي كامپايل و اجرا استفاده مي كنيم بايد همان نام كلاس باشد.

در هنگام اجراي برنامه هميشه متد main() از داخل كلاس برنامه شبيه به شكل زير اجرا خواهد شد(شبيه به برنامه هاي زبان C)

```
public static void main(String[] args) {
statements
}
```

زمان اجرا مفسر جاوا متد main() را اجرا مي كند واگر آن برنامه پارامتر ورودي داشته باشد، انها را در داخل آرايه ورودي از نوع رشته اي به نام args قرار مي دهد.
كلمه public به اين معني است كه متد main()مي تواند از بيرون برنامه فراخواني شود. در اينجا نوع خروجي كه پوچ(null) با كلمه void مشخص شده. در قسمت مشخص شده با statement دستورات برنامه را كه در مثال ما System.out.println("Hello World!") مي باشد وارد مي كنيم.

در زير فرم كلي كلاس برنامه جاوا كه علاوه بر متد main() ساير متدها و متغييرهاي كلاس را به همراه دارد نشان داد شده.

```
public class program-name {
optional-variable-declarations-and-subroutines
public static void main(String[] args) {
statements
}
optional-variable-declarations-and-subroutines
```

}

نام برنامه با **program-name** كه نام كلاس نيز مي باشد مشخص شده و اين كلاس مي بايست در يك فايل به نام **program-name** و پسوند java. ذخيره شود. در مثال ما HelloWorld.java مي باشد وبه اين فايل سورس كد مي گويند. بعد از كامپايل اين فايل يك فايل ديگر به نام HelloWorld.class ساخته خواهد شد كه شامل بايت كد هاي(bytecode) جاوا براي اجرا در مفسر جاوا است.
قسمت optional-variable-declarations-and-subroutines براي تعريف متغيرها و متد هاي عضو كلاس است و از داخل متد ()main قابل فراخواني مي باشند.

شي System.out يك كلاس از نوع PrintStream مي باشد.كلاس PrintStream استاندارد جاوا براي چاپ اطلاعات بوده و هر شي از نوع اين كلاس با متد , print println اطلاعات را به جريان خروجي ارسال ميكند. جريان خروجي مي تواند مقصدهاي مختلف مثل يك فايل، شي ديگر، پورت در شبكه و ساير خروجي ها باشد.

تاریخچه پیدایش زبان جاوا

اولین نسخه زبان جاوا در سال ۱۹۹۱ توسط تیمی از برنامه نویسان شرکت سان میکرو سیستم به نام Oak ساخته شد ودر سال ۱۹۹۲ با نام تجاری جاوا وارد بازار شد.

علت نام گذاری جاوا به این دلیل است هنگام انتخاب نام زبان افراد تصمیم گرفتند که ابتدای نامهای خود یعنی James Gosling ،Arthur Van hoff وAndy bechtolsheim برای این زبان انتخاب کنند وهمچنبن چون مراسم نامگذاریدر کافی شاپ برگذار می شد یک فنجان قهوه داغ به عنوان سمبول زبان جاوا انتخاب گردید.

همانند سایر زبانهای برنامه سازی عناصر واجزای اواج مجرد یا منفک ازهم نیستند و در ارتباط تنگاتنگ با هم هستند . این پیوستگی اجزائی در عین حال توصیف یکی از وجوه خاص این زبان را مشکل میسازد . زبان جاوا یک زبان کاملاً نوع بندی شده می باشد .

در واقع بخشی از امنیت و قدرتمندی جاوا از همین موضوع است . اول اینکه هرمتغیری وعبارتی دارای نوع میباشد و دقیقاً معین می باشد . دوم این که کلیه انتسابها بطور مستقیم وصریح از نظر سازگاری کنترل میشوند . به این دلیل اجبار خودکار یا تلاقی در در هم پیچیدن پیش نخواهد امد کامپایلر اواج کلیه عبارات و پارامترها و عبارتها را برای قابلیت سازگاری کنترل و بررسی میکند تا اطمینان حاصل شود و هرگونه ناسازگاری باید تصحیح شود.

دقت داشته باشید همانند زبان C++ نمیتوانید دادها هم شکل با طول کمتر را به طور مستقیم به نوع با طول بزرگتر تبدیل کنید مثلاً برای انتساب عدد صحیح به اعشاری باید قبل از عدد صحیح نام نوع آن عدد اعشاری را داخل پرانتز ذکر کنید .

استفاده از جاوا در اینترنت اولین بار توسط شرکت Netscape و با توافق با شرکت سان برای اینکه بتوان زبان استانداردی برای اجرای برنامه کوچک که در سمت سرویس گیرنده ها بود .برای همین منظور چون اینترنت ازبه هم

پیوستن انواع مختلف شبکه ها وکامپیوتر ها بود نیاز به طراحی زیر ساخت برای ماشینهای مختلف بود تا یک کد مشترک برای همه ماشینها را گرفته و بروی آن ماشین اجرا کند .
به همین دلیل شرکت سان تصمیم به توسعه ماشین مجازی جاوا یا JVM (Java Virtual Machine) گرفت . حرکت اشیاء در شبکه و اینترنت باعث گسترش روزافزون جاوا شد و پویا شدن کد های فرستاده شده به سمت سرویس گیرنده به کمک Applet (mini application) حجم کا روی سرویس دهنده را کاهش می داد .

Applet

یک برنامه کاربردی جاوا که برای حرکت وانتقال در اینترنت و اجرا توشط یک مرورگر قابل انطباق با جاوا است و چون اکثر مرورگر ها از آن پشتیبانی می کنند و درست مثل یک تصویر با فایل صوتی با این تفاوت که Applet هوشمند است .
میتوان گفت نقش Applet همانند ActiveX در برنامه های Visual Basic می باشد .

امنیت

JAVA IDE

Borland JBuilder 2005 Enterprise Edition

JBuilder IDEنخستين IDE ماژولار بود كه جاوا را پشتيباني كرد. معماري آن به خوبي مستند شده و راهنماهاي موجود براي نوشتن plug-in ها قابل فهم هستند؛ بسياري از فروشندگان واسط و همكاران كدباز (open source) اقدام به نوشتن ابزارهاي plug-in براي اينپلاتفرم نموده‌اند.

JBuilderدر سه نسخه عرضه مي‌گردد: نگارش شخصي يا Personal Edition ، كه به صورت مجاني قابل دانلود است، شامل IDE پايه و تعدادي ابزار اضافي از قبيل يك طراحGUI ، چاچوب يكپارچه JUnit ، و برخي آيتم‌هاي ديگر است؛ ويرايش توسعه‌گر يا Developer Edition، كه گروهي از مشخصه‌هاي مختلف را گرد هم آورده است، به خصوص پشتيباني XML و وب، servletها، JSP، و JSF (JavaServer Faces) ؛ و

ويرايش سازماني ياEnterprise Edition ، كه سرويس‌هاي وب، J2EE، پشتيباني CORBA، و رسم نمودار UML را به مجموعه مي‌افزايد. از بين اين سه ويرايش، JBuilder Enterprise Editionرا مورد بررسي قرار مي دهيم. به دليل تكامل آن، اين IDEدر ميان چهار IDE كه در اينجا مورد بررسي قرار گرفتند بهترين بود: عمل پيمايش در آن بصري و ساده است، و كليك كردن در ميان كارها هرگز به بن‌بست يا پنجره‌هاي غير منتظره منتهي نمي‌گردد.در صورت مواجه شدن با مشكل، يك سيستم help در دسترس است – بهترين در ميان چهار محصول ديگر. و براي توسعه‌گراني كه ترجيح مي‌دهند كار را با آموختارها(tutorial) ها) آغاز كنند، Borlandگزينه‌هايي واضح با طراحي خوب فراهم مي‌نمايد.

پشتيباني JBuilder براي مشخصه‌هايي كه وجود آنها در يك IDE سازماني high-end انتظار مي‌رود ناب، هوشمندانه و كاربردي است. تنها استثنا توليد فايل براي) Ant يك ابزار كمكي ساخت كدباز كه معمولا در جاوا مورد استفاده قرار مي‌گيرد) است، كه آزاردهنده مي‌باشد. با چشم‌پوشي از اين نقصان، محيط مزبور محيطي لذت‌بخش براي استفاده است.

فراتر از قابليتIDE ، JBuilder 2005 پشتيباني براي برخي تكنولوژي‌هاي منحصربفرد، از قبيل يكپارچگي باCORBA ، obfuscation كد، و تحليل امنيت كد با استفاده از يك plug-inمتعلق به Fortify را فراهم مي‌آورد.

JBuilderهمچنين به شكل قابل توجهي پشتيباني براي XML و مشتقات بيشمار آن را تكميل نموده است، و يك ويرايشگر HTML خوب به اضافه‌ي تعداد زيادي ابزار براي توسعه و تست سرويس‌هاي وب، و پشتيباني براي J2ME و WAP را فراهم آورده است. فرقي نمي‌كند كه كدام تكنولوژي با پروژه شما در هم آميخته است، به احتمال قريب به يقين JBuilder آن را پشتيباني مي‌نمايد.

JBuilder در هر صورت مدلسازي ضعيف است. بسته‌ي مزبور تنها دو نمودار UML را پشتيباني مي‌كند. اين نقصان از آنجا ناشي مي‌گردد كه Borland اقدام به خريداري محصول Together نموده است. Together .يك بسته‌ي high-end مدلسازي است كه در صورت نياز شما به مدلسازي زياد شما را به سوي آن هدايت مي‌كند. از ابتداي ماه مارس، Borlandيك مدل فروش را در پيش گرفته است كه Together را با JBuilder و ساير ابزارهاي Borland بر اساس نيازهاي توسعه‌گر تركيب مي‌نمايد. اين مجموعه‌ي محصولات مبتني بر وظيفه، كه Core SDP نام گرفته است، اساس عرضه‌ي سازماني Borlandرا از حالا به بعد شكل خواهد داد.

يك نكته جالب ديگر كه در كنفرانس سالانه Eclipse ، Borlandاعلام نمود كه مجموعه‌اي ازplug-in هاي Eclipse را عرضه خواهد كرد كه قابليت‌هاي JBuilder 2005 را افزايش خواهد داد. اين شركت تا چه زماني به پشتيباني دو GUI براي يك محصول ادامه خواهد داد مشخص نيست، اما از اين اعلام اين گونه به نظر مي‌رسد كه احتمالا JBuilder 2005در نهايت به سوي Eclipse گام بر خواهد داشت. اگر اين اتفاق رخ دهد، آن يك IDE عالي را رها خواهد ساخت، IDEاي كه Borland قابليت‌هاي بسياري را به آن افزوده است.

IBM Rational Software Architect

خط جديد محصولات Rational Software شركت IBM جايگزين خانواده WebSphere Studioمي‌گردد (Rational Software Architect) RSA. نسخه ۶ بر خلاف شماره

نسخه‌اش نخستين نسل تحت نام جديد است. از ميان مجموعه‌هاي متعددي كه كه مورد بررسي قرار ، RSAجامع‌ترين گزينه است.

IBMنيز مانند Borland از مدل مبتني بر وظيفه استفاده مي‌كند .RSA 6.0 به صورت چند لايه ساخته شده است. نخستين لايه Rational Web Developer است، كه بخش مهمي از قابليت مدلسازي را با خود ندارد؛ زير آن Rational Application Developerقرار دارد، كه فاقد قابليت‌هاي طراحي وب‌نما است. در پايين‌ترين سطح نيز Eclipse 3.0قرار دارد، IDEمجاني جاواگرا كه در حال كسب محبوبيت بسيار زيادي است.

متاسفانه، تقريبا تمامي كاستي‌هاي RSA ها ناشي از اين لايه پاييني است .اول از همه اين كه Eclipse يك رابط بصري نيست. تا وقتي كه شما در طي يك مدت زمان طولاني با آن آشنا گرديد، به احتمال زياد به پنجره‌هاي غير منتظره و بن‌بست بر خواهيد خورد. IBM كار را با مجموعه‌اي از آموختارهاي عالي ساده مي‌نمايد، اما همچنان پيمايش IDE در مقايسه با JBuilder 2005 يا Oracle JDeveloper دشوارتر است.

رابط RSA همچنين به ميزان قابل ملاحظه‌اي كندتر از دو محصول مذكور است. درست است كه هرگز به سطحي نمي‌رسد كه كاربران را از خود نااميد سازد، اما كندي آن محسوس است و فاقد سرعت موجود در ساير محصولات است، به خصوص در هنگام سوييچ ميان view هاي يك پروژه (همچون رفتن از طراحي به كدنويسي).

بخشي از اين كندي ناشي از ميزان نرم‌افزاري است كه IBM به دور هسته Eclipse گرد آورده است. مجموعه قابل توجهي از نرم‌افزارهاي سازمان-گرا موجود بر روي ۱۴ عدد .CD اين سرور كاربردي WebSphere شركت IBM را به عنوان يك محيط تست، يك مجموعه نرم‌افزاري كامل براي توسعه پورتال‌هاي وب، و مدلسازي عالي و ابزارهاي طراحي در بر مي‌گيرد.

ابزارهاي مدلسازي ۹ نمودار UML را پشتيباني مي‌نمايد .بيشتر از محصولات Borland و Oracle. اين نمودارها مي‌توانند براي الگوهايي كه در بر مي‌گيرند يا ممكن است در بر گيرند تحليل گردند، همچنين براي چيزي كه IBM آن را ضد الگوها مي‌نامد كه در واقع اشكالات موجود در طراحي برنامه مي‌باشند. براي مثال، كلاس‌هايي كه ضعيف طراحي شده باشند با توصيف‌هايي از الگوهاي طراحي مختل شده و نشانگرهايي كه نشان مي‌دهند چه چيز بايستي اصلاح گردد.

در تركيب با تحليل كد مبتني بر وظيفهIBM ، اين ابزارها به يك معمار كمك مي‌كند كه ببيند پروژه‌هاي خوب چگونه پياده‌سازي گرديده‌اند و چگونه آنها با راهنماهاي طراحي و نيازمندي‌هاي سايت مطابقت دارند.

در حال حاضر تكنولوژي رابط وب برگزيدهRational ، JSFاست. يك تكنولوژي كه فرايند پياده‌سازي را تسهيل مي‌نمايد. ابزارهاي RSA شامل يك ويرايشگر WYSIWYG براي JSFاست كه توسط SDO ها (service data object) براي رابطهاي پايگاه‌داده پشتيباني مي‌گردد.

RSA داراي پشتيباني جزئي براي C/C++ است، به علاوه‌ي پشتيباني كامل براي جاوا. ابزارهاي مدلسازي مي‌توانند كار تبديل به C++ را انجام دهند و ابزارهاي متعدد سورس‌كد مي‌توانند C++ را مورد تحليل قرار دهند. به هر حال، IDE ++Cفاقد يك كامپايلر و ديباگر است، يعني بايد جداگانه تهيه و نصب گردند. اگر شما يكي از اين

ابزارها را در اختيار داريد مي‌توانيد آن را نصب نماييد، يا مي‌توانيد كامپايلرهاي GNU C++را براي اين منظور دانلود كنيد. اين مشخصه‌ها همانند يك افزودني عجيب و ناقص به نظر مي‌رسند.
IBM RSAدر اين بررسي غني‌ترين محصول از لحاظ دارا بودن مشخصه‌هاي متعدد است. براي معماران سازماني كه خواهان تسلط يافتن بر رابط و ابزارها هستند، آن يك محصول برگزيده توسعه جاوا محسوب مي‌گردد.

Oracle JDeveloper

Oracleاز به كار بردن مدل نسخه‌هاي مبتني بر وظيفه‌ي IBM و Borland اجتناب ورزيده است و يك محصول قيمت پايين ارائه داده است. اين شركت از GUI خاص خود استفاده كرده كه داراي يك طراحي بصري است.

JDeveloper تنها محصولي است كه ظاهري زيبا داشته و محيطي تعاملي مي باشد. زمانبندي تاخيرهاي آغاز و ساير توابع بيانگر اين مطلب است كه كارآيي آن در يك ارتباط اساسي با Borland Jbuilder است، اندكي جلوتر از محصولSun ، و به ميزان قابل توجهي سريع‌تر از RSA محصول شركت IBM است. در سطح تعامل شخصي باIDE ، Oracleسريع‌تر به نظر مي‌رسد.

دستيابي به بسياري از مشخصه‌هاي JDeveloper در مقايسه با محصولات رقيب ساده‌تر است، كه اين امر انجام كارها را سرعت مي‌بخشد.

Oracleهمچنين يك سري موارد الحاقي جالب توجه را فراهم مي‌آورد .نخستين مورد ابزاري است كه يك تحليل زمان اجرا از كد شما انجام مي‌دهد. تحليل مزبور بر اساس بررسي خود پيشنهادهايي را درباره كلاس‌هايي كه مي‌توانند نهايي گردند ارائه مي‌دهد. اين پيشنهادات فراتر از موارد معمولي هستند كه تمامي IDE ها (شامل Oracle) همچنان كه شما كد را تايپ مي‌كنيد فراهم مي‌سازند، مواردي از قبيل نحوه بهينه‌سازي دستورات ورودي. در نسخه ويندوز، JDeveloperشامل JVM خاص خود است، كه براي اشكال‌زدايي بهينه گرديده است. (در هر صورت، براي اهداف گسترش، Oracleبه JVM سيستم رجوع مي‌نمايد).

براي كار با سرويس‌هاي وب، JDeveloperيك ناظر TCP را فراهم مي‌نمايد كه توسعه‌گران را قادر به بررسي تك‌تك پكت‌ها مي‌سازد مثل ثبت انتقال آنها و آشكار ساختن داده‌هاي آنها.
ساير ابزارها در مجموعه توسعه نرم‌افزار Oracle پشتيباني سطح بالاتري را براي سرويس‌هاي وب (از قبيل orchestration و BPEL يا Business Process Execution Language)فراهم مي‌آورند.

محدوديت عمده JDeveloper در مدلسازي UML است، جايي كه محصول تنها چهار نمودار اصلي(usecase,activity,class,sequence) را پشتيباني مي‌نمايد. JDeveloper نمودارهاي متعدد غير UML از قبيل طراحي نمودار EJB و Struts را پشتيباني مي‌كند.

در حالي كه اين محصول از لحاظ تعداد مشخصه‌هاي ارائه شده در معماري سازماني به پاي IBM و Borland نمي‌رسد، Oracle JDeveloper تمامي قابليت‌هايي را كه اغلب توسعه‌گران نياز دارند فراهم مي‌نمايد. و با وجود مزيت قابل توجه قيمت آن در مقايسه با دو رقيب ديگر، آن به احتمال قريب به يقين مي‌تواند محصول برگزيده براي بسياري از سايت‌ها قلمداد گردد.

Sun Java Studio Enterprise

در دهه ۹۰، تعداد اندكي از فروشندگان اقدام به عرضه ابزارهاي توسعه همپاي Sun نمودند. Sun نخستين شركتي بود كه نوآوري‌هاي بسياري را ايجاد نمود، از قبيل توانايي تغيير كد در ديباگر و ادامه اجرا.

هنگامي Sun سراغ ابزارهاي توسعه جاوا رفت قافيه را باخت، و به ساير فروشندگان اجازه داد بازاري را كه در واقع متعلق به خودش بود از آن خود نمايند .عرضه JSE (Java Studio Enterprise) 7مخصوصا با هدف حضور مجدد Sun در بازار ابزارهاي جاوا صورت گرفت. اين شركت تلاش فراواني را صرف مهيا ساختن اين محصول نمود و، در عمل، آن مشخصه‌هاي متعدد منحصر بفردي را عرضه كرد.
JSEبر پايه NetBeans بنا شده است، پلاتفرم كدبازي كه با Eclipse رقابت مي‌نمايد. با وجود غلبهEclipse ، NetBeans يك پلاتفرم با ارزش است. كه توانايي انجام تمام كارهايي كه Eclipse مي‌تواند انجام دهد را دارد. و همانندEclipse ، NetBeans از پشتيباني توسعه‌گران plug-in بيشماري برخوردار است، اگر چه Eclipse داراي تعداد بيشتري پروژه‌هاي فعال plug-in است.

از بسياري جهات، JSEمنحصربفرد است: آن مشخصه‌هاي قابل توجهي را كه ساير بسته‌ها فاقد آنها هستند فراهم مي‌نمايد، و آن فاقد برخي ابزارهايي است كه ساير بسته‌ها ارائه مي‌دهند. دو مشخصه بي‌نظير و قابل ستايش JSE عبارتند از طرح‌ريزي اجرا و همكاري.

كاركرد "همكاري" تمامي توسعه‌گران را با استفاده از JSE در يك جلسه خاص مشابه IBMقرار مي‌دهد و از اين رو آنها مي‌توانند پيغام‌ها و كد را با يكديگر تبادل نمايند. پشتيباني Whiteboarding نيز وجود دارد، و كانال‌هاي مجزايي براي مباحثات خصوصي و عمومي در آن موجود است.

هنگامي كه تنظيمات انجام شد، مشخصه همكاري به صورت خودكار در هنگام اجراي JSEاجرا مي‌گردد، از اين رو با نشستن براي انجام كار همه اعضاي يك گروه فوراً وارد تيم مي‌شوند.

مشخصه‌ي جذاب ديگر JSE شبيه‌ساز لود آن است. تست نمودن برنامه‌هاي توزيع‌شده كار دشواري است، آنها اغلب نيازمند تنظيم‌هاي پيچيده به منظور بازتوليد لودهايي هستند كه توانايي بازنمايي فعاليت دنياي واقعي را دارند. قادر بودن به تست كارآيي از درون JSE (با استفاده از تست لود داخلي آن) به معني صرفه‌جويي زماني قابل توجه براي توسعه‌گراني است كه بر روي برنامه‌هاي سازماني كار مي‌كنند.

پشتيباني Sun از نمودارهاي بيشمار UML اگر چه منحصربفرد نيست اما جذاب است. از اين لحاظ تنها RSA محصول IBM با آن برابري مي‌نمايد.

متاسفانه، از ساير لحاظ JSE دچار كاستي‌هايي است. آن Struts يا JSF را پشتيباني نمي‌كند. در عوض، آن از WAF (Web Application Framework) خود SUN استفاده مي‌نمايد. آن يك جايگزين است كه به سختي جذاب واقع خواهد شد، چرا كه در حال حاضر تعداد بسيار زيادي تكنولوژي جاوا وجود دارد كه در حوزه Web UI برنامه‌هاي سازماني رقابت مي‌كنند.

در سطح كدنويسي، JSEپيشنهادهايي را براي بهبود كد ارائه مي‌دهد. نهايتا، رابط داراي برخي جنبه‌هاي آشفته است، كه اين فكر را به ذهن مي‌آورد كه نسخه ويندوز JSEيك تبديل مستقيم از نسخه Solaris است.

براي مثال، يك سري پنجره‌ها برخي مواقع از شما در مورد مكان سيستم‌هاي فايل mountشده يا نقاط اتصال سوال مي‌كنند. اين اصطلاحات در دنياي ويندوز وجود ندارند، اما در يونيكس معمول هستند. اين مساله از آنجا ناشي مي‌شود كه سيستم help هيچ كاري براي ياري رساندن به توسعه‌گر ويندوز براي درك چيزي كه JSE در اين پنجره‌ها خواهان آن است انجام نمي‌دهد.

JSE 7شركت Sun را براي توسعه جاواي سازماني در جايگاه مناسبي قرار مي‌دهد. اما اگر شما نيازي به مشخه‌هاي بي‌نظير آن نداريد كه واقعا هم جذاب هستند، ساير محصولاتي كه در اينجا مورد بررسي قرار گرفتند گزينه‌هاي بهتري خواهند بود.

قابليتها

در حال حاضر هيچ زبان برنامه‌نويسي ديگري داراي چنين محيطهاي قدرتمندي (كه در اينجا بررسي شد) نيست. حتي Visual Studio .Net 2003 در قياس با آنها كم فروغ جلوه مي‌نمايد، اگر چه انتظار مي‌رود Visual Studio .Net 2005 اين فاصله را كاهش دهد.

اما با وجود كيفيت و قابليت‌هاي گسترده اين چهارIDE ، آنها فاقد عناصري هستند مدت‌ها پيش مي‌بايست در آنها ايجاد مي‌گرديد.

جاي برخي كاركردهاي كدنويسي ساده خالي است. براي مثال، چرا نبايد قادر باشيم ليترال‌ها يا بسته‌هاي منبع را از لحاظ املايي بررسي نماييم. همچنين، مشخصه‌هاي پيشرفته‌تر تنها حداقل پياده‌سازي‌ها را دارند. به عنوان مثال، سازندگان GUIبرايBorland ، Oracle، و Sun تنها كد Swing را توليد مي‌نمايند، تقريبا چنان كه گويي Standard Widget Toolkit وجود ندارد.

به استثنايBorland ، ويرايشگرهاي XML هنگامي كه مي‌بايست قدرتمند باشند ضعيف ظاهر گرديدند، و همچنين ويرايشگرهاHTML هيچ محصولي توانايي توليد تست‌هاي مهم واحد به روشي كه ابزارهاي واسط انجام مي‌دهند را ندارد. در عوض، اين IDE ها بنيان‌ها (stub ها) را فقط براي JUnit توليد مي‌كنند، حتي هنگامي كه تست‌ها براي يك كلاس خاص واضح هستند.

ارزيابي گزينه‌هايIDE

تنها راه براي خريداري تكنولوژي چه سخت‌افزار باشد و چه نرم‌افزار شناخت صحيح نيازهاي خود است. شما در مورد IDE هاي جاوا ناگزير به پيمودن اين گام مقدماتي خواهيد بود، چرا كه اين چهار محصول همگي به خوبي طراحي و پياده‌سازي شده‌اند.

يك ارتباط طبيعي ميان محصولات Borland و IBM وجود دارد، زيرا آنها بسته‌هايي هستند كه پروژه‌هاي بزرگ سازماني را با پشتيباني خاص براي معماران نرم‌افزار مد

نظر قرار داده‌اند. در مورد هر دو، محصولات اضافي كه توسط اين فروشندگان به فروش مي‌رسد مي‌تواند قابليت‌هاي بيشتري را فراهم آورند.

از بين اين دو، IBMداراي مشخصه‌هاي كامل‌تري مي‌باشد، اما آن قدري كندتر اجرا مي‌گردد و رابط آن كمتر از Borland بصري است.

اگر نيازهاي شما تماما در سطح معماري نيستند، در اين صورت JDeveloper شركت Oracleانتخاب خوبي به شمار مي‌آيد، چون به خوبي با ويندوز كاركرده و اگر چه Borlandنيز در اين حوزه رقيب شايسته‌اي مي‌باشد. اگر قيمت را در نظر بگيريم، در اين صورت Oracle برنده‌ي رقابت است.
فقط Sun JSE 7 مي‌تواند براي آن دسته از توسعه‌گراني كه نيازمند مشخصه‌هاي بي‌نظير آن يعني تست بارگذاري، همكاري گروهي، و طرح‌ريزي پروژهاي بزرگ هستند توصيه گردد.

هر چهار فروشنده نسخه‌هاي مخصوص ارزيابي را براي دانلود مجاني در اختيار عموم قرار داده‌اند تا در صورت تمايل بتوانيد اجراي آزمايشي آنها را تجربه نماييد. به هر حال، نصب، تنظيم، و تست اين محصولات كار ساده‌اي نيست .پيشنهاد اين است كه شما با محصول Oracle كار خود را آغاز نماييد، كه نصب آن ساده‌تر از ساير‌ين است، و در اغلب موارد بيشتر چيزهايي كه نياز داريد را فراهم مي‌نمايد.
اگر محدوديت‌هاي مدلسازي JDeveloper شركت Oracle شما را محدود مي‌سازد، توصيه مي‌شود JBuilder شركت Borland (Enterprise Edition) يا Rational RSA شركت IBM را دانلود كنيد
منبع:

آشنای مقدماتی با جاوا و قواعد نحوی ان

اعلان متغیر :
بصورت ; [=value] type identifier در آن Identifier نام متغیر و type نوع داده آن در جاوا میباشد . میتوانید باگذاشتن = ومقدار داخواه آن را مقدار دهی اولیه نماید .با استفاده از کاما میتوانید بیش از یک متغیر بایک نوع تعریف کنید
توضیحات :
برای خوانا تر شدن برنامه خود می توانید از توضیحات در برنامه خود استفاده کنید همچنین وقتی برنامه شما بزرگ بوده فهم آن را ساده تر میکند .
برای این کار میتوانید از // یا از */ *برای توضیحات چند خطی استفاده کنید .

```
int d = 3, e, f = 5; // declares three more ints/ initializing  // d and f.
byte z = 22; // initializes z.
double pi = 3.14159; // declares an approximation of pi.
char x = 'x'; // the variable x has the value 'x'.
```

قلمرو و زمان حیاط متغیر ها :

جاوا به متغیر ها اجازه می دهد تا در درون یک بلوک تعریف شود(داخل { }) پس هر بار که یک بلوک جدید را شورع میکنید یک قلمرو جدید ایجاد میشود ودر داخل آنها اشیاء قابل رویت و زمان حیاط انها مشخص میشود . تعریف اشیائ میتواند بصورت سراسری یا محلی باشد.

اگریک متد دارای پارامتر های ورودی باشد همانند یک متغیر ازآن استفاده میشود .اگر برنامه شامل بلوکهای مختلف تودر تو باشد اشیاءی اعلان شده در بلوک بیرونی برای کدهای بلوک درونی قابل رویت میباشد اما معکوس این قضیه درست نمی باشد .

همانطور که گفته شد زمان حیاط یک متغیر محدود به به آن بلوک میباشد و اگر اعلان یک متغیر شامل مقدار دهی اولیه باشد انگاه هر زمان که به بلوک وارد میشویم ان متغیر مقدار دهی اولیه میشود .

تبدیل خودکار :

برای اینکار میباید نوع مقصد از نوع منبع باشد وهمچنین آن دو با هم سازگار باشند مثلآ نوع int با char یا Boolean سازگار نیستند بطور مثال اگر i ازنوع int و L ازنوع long باشد دستور ; L=i درست می باشد.

تبدیل غیرخودکار انواع ناسازگار:
استفاده ان در جای که بطور مثال می خواهید مقدارa int را به یک byte b نسبت دهید برای اینکار از تبدیل target =(target type)value; که به فرم b=(byte)a; میشود.

boolean	True , False	void	بدون نوع یا null
byte	۱بایت	char	۲ بایتی کارکتر
Short	۲ بایت صحیح	float	۴ بایت اعشاری
long	۸ بایت صحیح	double	۸ بایت اعشاری
int	۴ بایت صحیح	string	رشته طول متغیر

عملگر های منطقی و بولی:

این عملگر ها بر روی عملوندهای بولی عمل می کنند و دو مقدار Boolean را ترکیب می کنند تا یک مقدار Boolean ایجاد شود.
AND با &
OR با |
XOR با ^
NOT با !
آنهای که در بالا گفته شد برای عملگرهای بیتی بود وبرای Boolean :
AND با &&
OR با ||
مساوی با ==
نامساوی با =!
خروجی برنامه زیر مثالها رابهتر شرح میدهد.

```
class BoolLogic {
 public static void main(String args[] ){
```

```
boolean a = true;
boolean b = false;
boolean c = a | b;
boolean d = a & b;
boolean e = a ^ b;
boolean f =( !a & b )|( a & !b);
boolean g = !a;
System.out.println(" a = " + a);
System.out.println(" b = " + b);
System.out.println(" a|b = " + c);
System.out.println(" a&b = " + d);
System.out.println(" a^b = " + e);
System.out.println("!a&b|a&!b = " + f);
System.out.println(" !a = " + g);
}
}
```

خروجی:

```
a = true
b = false
a|b = true
a&b = false

a^b = true
!a&b|a&!b = true
!a = false
```

اعلان اشياء :

براي اينكار بايد از نوع آن شي يك متغير تعريف كرده سپس بايد يك نمونه فيزيكي از شي بدست آورده و با كمك متغير با آن كار كنيد. براي گرفتن شي فيزيكي در حافظه Ram بايد از عملگر new استفاده كنيد. اين عملگر حافظه به شي تخصيص ميدهد و و يك ارجاع(آدرس حافظه آن شي) را به متغير تعرف شده از آن نوع را برميگرداند.
در زبان جاو ا كليه اشياء كلاس مي بايد بصورت پويا تخصيص يابند.

```
Box mybox; // declare reference to object

mybox = new Box(); // allocate a Box object
```

ابتدا متغير mybox به عنوان اشاره گر به شي Box تعريف ميكنيم كه مقدار اوليه آن (null) يا تهي خواهد بود يعني هنوز به شي واقعي اشاره نمي كند. در خط بعدي به mybox را به يك شي واقعي انتساب مي دهد.

شكل عمومي عملگر new :

شكل عمومي به صورت

```
class-var = new classname();
```

در اينجا class-var متغير ازنوع كلاس مورد نظر و classname نام كلاسي كه ميخواهيم از آن شي معرفي كنيم ()classname به عنوان سازنده كلاس به كار مي رود .سازنده

(constructor)در كلاسها به عنوان تعريف كننده نحوه ايجاد شي مي باشد و بايد صريح در داخل كلاس خود معرفي شده باشد .اگر در داخل كلاس خود سازنده صريح موجود نباشد جاوا يك سازنده پيش فرز براي آن ارائه ميكند .
مزيت استفاده از new در اين است كه برنامه در زمان اجرا هر وقت كه نياز به شي آنرا ايجاد مي كند. توجه كنيد كه اگر به علت كمبود حافظه new نتواند حافظه تخصيص يك استثنا در زمان اجرا رخ مي دهد.

عملگرها

عملگر هاي جاوا را ميتوان درچهارگروه دسته بندي كرد : عملگرهاي حسابي ، بيتي ، رابطه اي و منطقي .اگر با زبان C / C++ اشنا باشيد كاربرد عملگرها دقيقآ همانند كاربرد آنها در جاوا است .

عملگرهاي حسابي (Arithmetic)

اين عملگر ها درعبارات رياضي روي اعداد استفاده مي شوند

+ جمع

- منها

* ضرب

/ تقسيم

% باقيمانده

++ افزايش يك واحد

-- كاهش يك واحد

=+ انتساب به همراه اضافه كردن

=- انتساب به همراه تفريق كردن

=* انتساب به همراه ضرب كردن

=/ انتساب به همراه تقسيم كردن

=% انتساب به همراه تعيين باقيمانده كردن

برخي از اين عملگرها روي نوع char قابل استفاده اند. اما نمي توان روي boolean استفاده كرد. در عبارات رياضي عملگر ها با اولويت يكسان بسته به ترتيب آنها از سمت چپ پردازش ميشوند. اگر بخواهيم عملگر با اولويت كمتر زود تر از عملگر با اولويت بالاتر پردازش شود بايدآن را در داخل دو پرانتز () گذاشت.
مثالهاي زير نحوه استفاده از عملگر ها را نشان ميدهد .

```
class BasicMath {
public static void main(String args[] ){
// arithmetic using integers
System.out.println("Integer Arithmetic");
```

```
int a = 1 + 1;
int a = a * 3;
int a = b / 4;
int a = c - a;
int a =- d;
System.out.println("a = " + a);
System.out.println("a = " + b);
System.out.println("a = " + c);
System.out.println("a = " + d);
System.out.println("a = " + e);
// arithmetic using doubles
System.out.println("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de =- dd;
System.out.println("da = " + da);
System.out.println("db = " + db);
System.out.println("dc = " + dc);
System.out.println("dd = " + dd);
System.out.println("de = " + de);
} }
```

خرو جي برنامه به شكل زير است

```
integer Arithmetic
a=2
b=6
c=1
d=-1
e=1
floating point arithmetic
da=2
db=6
dc=1.5
dd=-0.5
de=0.5
```

عملگرها بيتي (bitwise)

در جاوا چندين عملگر رفتا بيتي تعريف شده ميباشد كه روي انواع عددي و char اعمال مي شوند

~ مكمل بيتي

& and بيتي

| بيتي or

^ بيتي xor

<< شيفت به راست بيتي و با صفر پر شدن

>> شيفت به چپ بيتي و با صفر پر شدن

=& انتساب و بيتي and

=| انتساب و بيتي or

=^انتساب و بيتي xor

<<= انتساب و شيفت به راست بيتي و با صفر پر شدن

>>= انتساب و شيفت به چپ بيتي و با صفر پر شدن

به كمك مثال طرز كار اين عملگر ها را ميتوان فهميد .

```
class BitLogic {
public static void main(String args[] ){
String binary[] = {
"0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111",
"1000", "1001", "1010", "1011", "1100", "1101", "1110", "1111"
};
int a = 3; // 0 + 2 + 1 or 0011 in binary
int b = 6; // 4 + 2 + 0 or 0110 in binary
int c = a | b;
int d = a & b;
int e = a ^ b;
int f =( ~a & b )|( a & ~b);
int g = ~a & 0x0f;
System.out.println(" a = " + binary[a]);
System.out.println(" b = " + binary[b]);
System.out.println(" a|b = " + binary[c]);
System.out.println(" a&b = " + binary[d]);
System.out.println(" a^b = " + binary[e]);
System.out.println("~a&b|a&~b = " + binary[f]);
System.out.println(" ~a = " + binary[g]);
} }
```

خروجي به شكل زير است .

```
a=1011
b=0110
a^Eb=0111
a&b=0010
a&b=0101
```

~a&b^Ea&~b=0101
~a=1100

عملگر انتساب(Assignment)

علامت = و شكل كلي var=expression; در اينجا نوع متغير var بايد با خروجي عبارت expression سازگار باشد. در زبانهاي C /C++ و جاوا اين امكان وجود دارد كه تا زنجيره اي از انتسابها رابه وجود آورید .

```
int x, y, z;

x = y = z = 100; // set x, y, and z to 100
```

قوانين ارتقاء انواع

علاو بر ارتقاءshort,byte به int جاوا چندين قانون ارتقا انواع را تعريف كرده كه در عبارات قابل استفاده اند.قانون اول اينكه همان طور كه گفته شد كليه مقادير short,byte به int ارتقاء مي يابند.انگاه اگر يك عملوند از نوع long باشدكل عبارت به long ارتقاء مي يابد. اگر يك عملوند از نوع float باشد كل عبارت به float ارتقاء مي يابد. اين قانون در مورد double نيز صادق مي باشد.برنامه زير نشان مي دهد كه چگونه هر يك از مقادير در عبارت ارتقاء مي يابد تا با آرگمان دوم به هر يك از عملگرها دودوي، مطابقت يابد.

```
class Promote {
 public static void main(String args[] ){
 byte b = 42;
 char c = 'a';
 short s = 1024;
 int i = 50000;

 float f = 5.67f;
 double d =. 1234;
 double result =( f * b )+( i / c( - )d * s);
 System.out.println((f * b )+ " + " +( i / c )+ " - " +( d * s));
 System.out.println("result = " + result);
 }
 }
```

در عبارت double result =(f * b)+(i / c(-)d * s); حاصل f*b ، b به float ارتقاء يافته و جواب عبارت نيز از نوع float خواهد بود.در عبارت بعدي يعني i/c ، c به نوع int ارتقاء يافته و جواب از نوع int خواهد بود.نتيجه عبارت d*s ، نيز ازنوع double خواهد بود. در نهايت جون نوع double از ساير انواع بزرگتر مي باشد جواب نهاي از نوع double خواهد بود.

عملگر ?

همانند زبان C++ در جاوا ميتوانيم از علامت ? براي عملگر سه تاي ويژه براي جايگزيني انواع مشخصي از دستورات if-then-else استفاده كنيم.شكل كلي اين عملگر به فرم زير است.

```
experssion1 ? expersion2 : experssion3
```

در اينجا experssion1 هر عبارت بولي ميتواند باشد. اگر experssion1 صحيح باشد عبارت experssion2 اجرا خواهد شد در غير اينصورت experssion3 اجرا ميشود. در زير مثال استفاده از ? بيان شده.

```
ratio = denom == 0 ? 0 : num / denom;
```

زماني كه جاوا اين عبارت انتساب را ارزيابي مي كند ابتدا عبارت سمت چپ علامت سوال بررسي مي شود اگر denom برابر با صفر باشد آنگاه عبارت بين علامت سوال وعلامت (colon) ارزيابي مي شود و به عنوان مقدار كل عبارت ? استفاده مي شود. حال اگر denom مساوي با صفر نباشد آنگاه عبارت بعد از (colon) ارزيابي شده و براي مقدار كل عبارت ? استفاده مي شود. نتيجه توليد شده توسط عملگر ? سپس به ratio نسبت داده مي شود.
در برنامه زير نحوه استفاده از ? را براي قدر مطلق مشاده مي كنيد.

```
class Ternary {
 public static void main(String args[] ){
 int i/ k;

 i = 10;
 k = i < 0 ?- i : i; // get absolute value of i
 System.out.print("Absolute value of ");
 System.out.println(i + " is " + k);
 }
 }
```

خروجي برنامه به شكل زير ميباشد.

```
Absolute value of -10 is 10
```

كلاسها در جاوا

در زماني كه يك كلاس را تعريف مي كنيد در حقيقت شكل و طبيعت آن شي را به كمك متدها و صفات آن را اعلان مي كنيد براي اعلان يك كلاس از واژه كليدي class استفاده مي كنيم و كلاسهاي كه تا به كنون از آنها استفاده كرده ايد كلاسهاي محدودي از شكل كامل كلاسها بوده اند.

```
Class class_name {

Member_Variables

Member_Method
}
```

داده ها يامتغيير هاي تعريف شده در داخل يك كلاس را متغيير هاي نمونه (instance variables) مي نامند. اين نام گذاري به اين دليل است كه هر نمونه از كلاس شامل يك كپي از اين متغيير هاي خاص خودش است. بنا براين داده هاي مربوط به يك شي ، جدا و منحصر به فرد از داده هاي اشياء ديگر است.

کدهاي کلاس که روي متغير هاي نمونه اعمال را انجام ميدهند را در داخل متدهاي آن کلاس قرار مي دهند. اين اعمال مي تواند اعمال محاسباتي ، اعمال نمايش خروجي ، اعمال تبديل و غيره باشند. متد main() را زماني در کلاسها استفاده مي کنند که نياز به اجراي آن کلاس توسط کامپايلر جاوا داريد و يا آن کلاس نقطه شروع در برنامه شما باشد.
در اين جا يك مثال ساده از کلاس نشان داده شده. نام اين کلاس Box مي باشد و سه متغير نمونه در داخل خود دارد و متدي در داخل آن پياده سازي نشده.

```
class Box {
double width;
double height;
double depth;
}
```

هر کلاس زماني که تعريف مي شود يك نوع جديدي از داده ها را ايجاد مي کند و در مثال ما اين نوع جديد Box است. توجه داشته باشي که اعلان يك کلاس يك شي واقعي را به وجود نمي آورد و فقط الگو را ايجاد مي کند. براي ايجاد يك شي واقعي از نوع Box بايد از دستور زير استفاده کنيد.

```
Box mybox = new Box(); // create a Box object called mybox
```

پس از اجرا mybox نمونه اي از Box خواهد بود. به منظور دسترسي به متغيير هاي نمونه شي از عملگر نقطه استفاده کنيد. اين عملگر نام يك شي را اب يك متغيير نمونه پيوند مي دهد.به عنوان مثال براي انتساب مقدار 100 به متغيير width از دستور زير بايد استفاده کرد.

```
mybox.width = 100;
```

بطور کلي عملگر نقطه براي دسترسي به متغيير نمونه و هم براي متدهاي موجود در شي استفاده مي شود. در اينجا يك برنامه کامل که از کلاس Box استفاده مي کند را مشاهده مي کنيد.

```
/* A program that uses the Box class.
Call this file BoxDemo.java
/*
class Box {
double width;
double height;
double depth;
}
// This class declares an object of type Box.
class BoxDemo {
public static void main(String args[] ){
Box mybox = new Box();
double vol;
// assign values to mybox's instance variables
mybox.width = 10;
mybox.height = 20;
mybox.depth = 15;
// compute volume of box
vol = mybox.width * mybox.height * mybox.depth;
System.out.println("Volume is " + vol);
```

```
}
 }
```

حال برنامه نوشته شده را در فايلي به نام BoxDemo.java قرار بدهيد. دليل اين كار هم مشخص است و اين اينكه متد main() در داخل اين كلاس قرار دارد و كامپايلر جاوا به طور خود كار آنرا اجرا خواهد كرد.

بعد كامپايل كردن دو فايل با پسوند class. ايجاد خواهند شد كه يكي از آن دو براي كلاس Box و ديگري براي BoxDemo .ضرورتي براي اين وجود ندارد كه هر دو كلاس BoxDemo, Box در داخل يك فايل منبع قرار بگيرند. اين امكان وجود دارد كه هر كلاس را در فايل جدا مخصوص خود گذاشته و آنها را به ترتيب Box.java و BoxDemo.java نامگذاري كنيد. براي اجراي اين برنامه BoxDemo.class را اجرا كنيد.

پس از اجرا خروجي زير حاصل خواهد شد.

```
Volume is 3000
```

همان طور كه قبلاً اشاره كرديم هر شي داراي كپي نمونه هاي خاص خود است و اگر دو نمونه از شي Box داشته باشيم هر كدام متغيير هاي خود را خواهند داشت. تغيير روي هركدام از آنها تاثيري بر روي ديگري ندا شته. اين نكته را مثال زير نشان مي دهد.

```
// This program declares two Box objects.
class Box {
double width;
double height;
double depth;
}
class BoxDemo2 {
public static void main(String args[] ){
Box mybox1 = new Box();
Box mybox2 = new Box();
double vol;
// assign values to mybox1's instance variables
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
/* assign different values to mybox2's
instance variables */
mybox2.width = 3;
mybox2.height = 6;
mybox2.depth = 9;
// compute volume of first box
vol = mybox1.width * mybox1.height * mybox1.depth;
System.out.println("Volume is " + vol);
// compute volume of second box
vol = mybox2.width * mybox2.height * mybox2.depth;
System.out.println("Volume is " + vol);
}
 }
```

خروجي توليد شده به فرم زير خواهد بود.

```
Volume is 3000
Volume is 162
```

تعريف كنترل دسترسي

همان طور كه در شي گراي گفته شده قابليت كپسوله سازي داده را با كدي كه با آن داده سرو كار دارد پيوند ميدهد. خطلت مهم ديگر كپسوله سازي كنترل دستيابي (access control) است. به كمك كپسوله سازي مي توانيد كنترل كنيد كه چه بخشهاي از يك برنامه مي توانند به اعضاي يك كلاس دسترسي داشته باشند و از سوء استفاده جلوگيري كنيد. اجازه دادن دسترسي به دادها فقط از طريق يك مجموعه خوش تعريف از روشها مانع سوء استفاده از داده هاي آنها مي شود. به كار ساخت جعبه سياه (black box) مي گويند كه از آن كلاس استفاده مي كنيم ولي از نحوه كار در داخل آن خبر نداريم.
چگونه گي دسترسي به يك متغيير يا متد عضو كلاس توسط توصيف گر دسترسي تعريف مي شود.جاوا مجموعه غني از توصيف گر دسترسي را در اختيار دارد.برخي جوانب كنترل دسترسي به شدت با وراثت و بسته ها(packages) مرتبط هستند.
بسته ها يك نوع گروح بندي براي كلاسها است.اگر يك بار پايه كنترل دسترسي را فرا بگيريد آنگاه بقيه مطالب را به راحتي خواهيد آموخت.
توصيف گر هاي دسترسي در جاوا با كلمات كليدي public, private, protected مشخص مي شوند. همچنين جاوا به طور پيش فرز از protected براي سطح دسترسي در وراثت استفاده مي كند.
اگر يك عضو كلاس را با public توصيف كنيم، آن عضو توسط هر كد ديگري در برنامه قابل دسترس خواهد بود. اگر يك عضو كلاس را با private توصيف كنيم، آن عضو توسط پس از آن فقط توسط متدهاي عضو آن كلاس قابل دسترس خواهد بود. اكنون متوجه شده ايد كه چرا در قبل از متد ()main توصيف گر public قرار مي گيرد. اين متد توسط اجرا كننده برنامه هاي جاوا يعني فايل اجراي java.exe فراخواني خواهد شد.
به طور پيش فرز اگر از توصيف گر در تعريف اعضاء استفاده نكنيم، آنها در كلاسهاي مربوط به بسته اي(package) كه آن كلاس داخل آن قرار دارد قابل دسترس خواهند بود اما در بيرون از بسته قابل دسترسي نمي باشند.
در كلاسهاي كه قبلآ استفاده كرديم به طور پيش فرز از public استفاده كرده اند. شما معمولآ مايل هستيد كه دسترسي به اعضاي داده هاي يك كلاس را محدود كنيد فقط برخي متدها قابل دسترس باشند. همچنين شرايطي وجود دارند كه مايليد كه متدهاي كه مخصوص يك كلاس اختصاص مي باشند را توسعه دهيد.
يك توصيف گر دسترسي قبل از ساير مشخصات نوع عضو يك كلاس قرار مي گيرند واين به اين معني است كه توصيفگر بايد شروع كننده دستور اعلان يك عضو باشد. در زير يك مثال را مشاهده مي كنيد.

```
public int i;
private bouble j;
private int myMethod(int a/ char b ){ //...
```

برا يدرك تاثيرات دسترسي عمومي(public) و اختصاصي(private) برنامه زير را در نظر بگيريد.

```
/* This program demonstrates the difference between
public and private.
*/
class Test {
int a; // default access
public int b; // public access
private int c; // private access
```

```
// methods to access c
void setc(int i ){ // set c's value
c = i;
}
int getc (){ // get c's value
return c;
} }
class AccessTest {
public static void main(String args[] ){
Test ob = new Test();
// These are OK/ a and b may be accessed directly
ob.a = 10;
ob.b = 20;
// This is not OK and will cause an error
// ob.c = 100; // Error!
// You must access c through its methods
ob.setc(100); // OK
System.out.println("a/ b/ and c :" + ob.a + " " +
ob.b + " " + ob.getc());
}
 }
```

همان طور كه ملاحظه مي كنيد داخل كلاس Test، متغيير a از دسترسي پيش فرز استفاده مي كند كه در اين مثل مطابق مشخص نمودن با public است. متغيير b به طور صريح با public مشخص شده . متغيير c داراي دسترسي اختصاصي است. اين معني است كه توسط كدهاي خارج از كلاس قابل دسترس نخواهد بود و در داخل كلاس Access Test نمي توان c را به طور مستقيم مورد استفاده قرار داد. اين عضو توسط متدهاي عمومي آن يعني getc() و setc() بايد .ورد دسترسي قرار گيرد.

اگر دستور ob.c=100; را دراخل برنامه استفاه كنيد كامپايلر از برنامه خطا خواهد گرفتو برنامه قابل اجرا نخواهد بود. براي اينكه با جنبه هاي عملي كنترل دسترسي آشنا شويد كلاس زير براي ساخت يك stack ايجاد شده.

```
// This class defines an integer stack that can hold 10 values.
class Stack {
/* Now/ both stck and tos are private .This means
that they cannot be accidentally or maliciously
altered in a way that would be harmful to the stack.
*/
private int stck[] = new int[10];
private int tos;
// Initialize top-of-stack
Stack (){
tos =- 1;
}
// Push an item onto the stack
void push(int item ){
if(tos==9)
```

```
System.out.println("Stack is full.");
else
stck[++tos] = item;
}
// Pop an item frome the stack
int pop (){
if(tos < 0 ){
System.out.println("Stack underflow.");
return 0;
}
else
return stck[tos--];
}
 }
```

همان طور كه ملاحظه مي كنيد هم آرايه stck (نگه دارنده پشته) و هم متغيير tos(كه نمايان گر بالاي پشته) به عنوان privateمعرفي شده اند. اين به اين معني است كه آنها قابل دسترسي و جايگزين نيستند. به عنوان مثال اختصاصي تعريف نمودن tos آن را در مقابل قرار دادن غير عمدي مقداري كه فراتر از انتهاي آرايه stck است محافظت مي كند. برنامه بعدي نشان دهنده استفاده از كلاس توسعه يافته stack است .

```
class TestStack {
public static void main(String args[] ){
Stack mystack1 = new Stack();
Stack mystack2 = new Stack();
// push some numbers onto the stack
for(int i=0; i<10; i++ )mystack1.push(i);
for(int i=10; i<20; i++ )mystack2.push(i);
// pop those numbers off the stack
System.out.println("Stack in mystack1:");
for(int i=0; i<10; i++)
System.out.println(mystack1.pop());
System.out.println("Stack in mystack2:");
for(int i=0; i<10; i++)
System.out.println(mystack2.pop());
// these statements are not legal
// mystack1.tos =- 2;
// mystack2.stck[3] = 100;
}
 }
```

اگر چه متدها معمولا دسترسي به داده هاي تعريف شده يك كلاس را كنترل مي كنند، اما هميشه هم اين طور نيست .كاملآ به جاست كه هرگاه دليل خوبي براي اين كار داشته باشيم ، اجازه دهيم تا يك متغيير داخل كلاس از دسترسي public استفاده كند. به عنوان مثال اكثر كلاسهاي ساده با كمترين توجه نسبت به كنترل دسترسي به متغييرهاي نمونه ايجاد شده اند واين بي توجهي فقط براي ساده بودن مثالها بوده.

محافظت دسترسي (Access Protection)

همان طور كه مي دانيد دسترسي به يك عضو private در يك كلاس فقط به ساير اعضاي ان كلاس محدود شده. بسته ها (package) قابليت اين ار دارند كه كنترل ديگري به برنامه اضافه كنند. همان طوري كه خواهيد ديد جاوا سطوح مختلفي براي كنترل طبقه بندي شده و رويت پذيري متغييرها و متدها در داخل كلاسها و زير كلاسها ي داخل بسته فراهم ميكند.

بسته ابزاري براي كپسوله سازي بالاتر از سطح كپسوله سازي در كلاسها است و در بر گيرنده فضاي نام و قلمرو متغير ها و متدها است.زماني كه تعداد كلاسها و اشياء در توسعه برنامه هاي كاربردي افزايش پيدا مي كند كه امروزه در عمل بسيار افزايش يافته براي طبقه بندي آنها در سطح بلاتر از كلاس از بسته هاي جاوا استفاده ميكنيم.

بسته ها به عنوان ظروفي براي براي كلاسها و ساير بسته هاي وابسته هستند. كلاسها به عنوان ظروفي براي داده ها و كدها مي باشند و كوچكترين واحد مجرد سازي در جاوا هستند. به لحاظ نقش متقابل بين كلاسها و بسته ها ، جاوا چهار طبقه بندي براي رويت پذيري اعضاي كلاس مشخص كرده:

١- زير كلاسها در همان بسته
٢- غير زير كلاسها در همان بسته
٣- زير كلاسها در بسته هاي مختلف
٤- كلاسهاي كه نه در همان بسته و نه در زيرك لاسها هستند.

همچنين سه توصيفگر گفته شده يعني public , private ,protected را هم فراهم كرده. با ايين روشها مي توانيم سطوح مختلفي از دسترسي ايجاد كنيم.

اگر چه مكانيزم كنترل دسترسي در جاوا ممكن است در ابتدا پيچيده باشد، اما مي توان به شكل آنها توضيح داد. هر چيزي كه در تعريف آن از public استفاده شود از هرجا قابل دسترس مي باشد. هر چيزي كه در تعريف آن از private استفاده شود خارج از كلاس خودش قابل رويت نمي باشد. زماني كه يك عضو فاقد مشخصات دسترسي صريح باشد، آن عضو براي كلاسها و زير كلاسهاي موجود در بسته قابل رويت مي باشد و به صورت دسترسي پيش فرز است. اگر شما خواستار آن هستيد كه يك عضو خواه متد يا متغيير خارج از بسته جاري و فقط به كلاسهاي مستقيمآ از كلاس شما به صورت زير كلاس در آمده اند قابل رويت باشد، بايد آنرا با توصيفگر protected تعريف نمايد.

يك كلاس فقط دو سطح دسترسي ممكن دارد: پيش فرز و عمومي . عمومي زماني كه يك كلاس به عنوان public اعلام مي شود، توسط هر كد ديگري قابل دسترسي است . اگر كلاس دسترسي پيش فرز را داشته باشد، فقط توسط ساير كدهاي داخل همان بسته قابل دسترس خواهد بود.

يك مثال از دسترسي

اين مثال كليه تركيبات مربوط به اصلاح گرهاي كنترل دستري را نشان مي دهد و داراي دو بسته و پنج كلاس است. به ياد داشته باشيد كه كلاسهاي مربوط به دو بسته متفاوت، لازم مي باشد كه در دايركتوريهاي به نام p1 و p2 كه نام بسته ها نيز مي باشند ذخيره خواهند شد. در بسته اول سه كلاس Protection و Derived و samepackage محود است. كلاس اول يعني Protection داراي چهار متغيير int كه هر يك در حالتهاي مختلف مجاز تعريف شده اند.

متغيير n با حفاظت پيش فرز اعلان شده. m-pri به عنوان private و n-pro با protected همچنين n-pubبه عنوان public تعريف شده.

كلاسهاي بعدي در اين بسته سعي مي كنند كه به متغيير هاي نمونه از يك كلاس دسترسي پيدا كنند. خطهاي كه به لحاظ محدوديتهاي دسترسي كامپايل نمي شوند با استفاده از توضيح خطي يعني با // مشخص شده اند. قبل از هر يك از اين خطوط توضيحي قرار دارد كه مكانهاي را كه از آنجا اين سطح از حفاظت اجازه دسترسي مي يابد را فهرست مي نمايد.
دومين كلاس Derived يك زير كلاس از Protection در همان بسته p1 است. اين مثال دسترسي Drived را به متغييرهاي Protection برقرار يم كند به جز n-pri كه يك متغير از نوع دسترسي private است.
سومين كلاس Samepackage يك زير كلاس از Protection نبوده اما در همان بسته p1 قرار دارد. بنابراين به كليه متغيير ها به جز n-pri دسترسي خواهد داشت.

```
package p1;
public class Protection {
int n = 1;
private int n_pri = 2;
protected int n_pro = 3;
public int n_pub = 4;
public Protection (){
System.out.println("base constructor");
System.out.println("n = " + n);
System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pub = " + n_pub);
} }
class Derived extends Protection {
Derived (){
System.out.println("derived constructor");
System.out.println("n = " + n);
// class only
// System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pub = " + n_pub);
} }
class SamePackage {
SamePackage (){
Protection p = new Protection();
System.out.println("same package constructor");
System.out.println("n = " + p.n);
// class only
// System.out.println("n_pri = " + p.n_pri);
System.out.println("n_pro = " + p.n_pro);
System.out.println("n_pub = " + p.n_pub);
}
 }
```

اكنون كد منبع يك بسته ديگر به نام p2را ملاحظه مي كنيد. دو كلاس تعريف شده در p2 دو شرايطي را كه توسط كنترل دسترسي تحت تاثير قرار گرفته اند را پوشش داده است.

اولين كلاس يعني Protection2 يك زير كلاس از p1.Protection است. اين كلاس دسترسي به كليه متغييرهاي مربوط به p1.Protection را به غير از n-pri كه از نوع private است بدست مي آورد. به ياد داشته باشيد كه به طور پيش فرز فقط اجازه دسترسي از داخل كلاس يا بسته را مي دهد نه از زير كلاسهاي بسته هاي اضافي. در نهايت كلاس OtherPackage فقط به يك متغير n-pub كه به عنوان public اعلان شده دسترسي دارد.

```
package p2;
class Protection2 extends p1.Protection {
Protection2 (){
System.out.println("derived other package constructor");
// class or package only
System.out.println("n = " + n);
// class only
// System.out.println("n_pri = " + n_pri);
System.out.println("n_pro = " + n_pro);
System.out.println("n_pub = " + n_pub);
} }
class OtherPackage {
OtherPackage (){
p1.Protection p = new p1.protection();
System.out.println("other package contryctor");
// class or package only
System.out.println("n = " + p.n);
// class only
// System.out.println("n_pri = " + p.n_pri);
// class/ subclass or package only
// System.out.println("n_pro = " + p.n_pro);
System.out.println("n_pub = " + p.n_pub);
}
 }
```

اگر مايل به اجراي اين دو بسته هستيد دو فايل آزمايشي وجود دارد كه مي توانيد از آنها استفاده نمايد. يكي از اين فايلها را براي بسته p1 در اينجا نشان داده ايم.

```
// Demo package p1.
package p1;
// Instantiate the various classes in p1.
public class Demo {
public static void main(String args[] ){
Protection ob1 = new Protection();
Derived ob2 = new Drived();
SamePackage ob3 = new SamePackage();
}
}
```

فايل آزمايشي براي p2 به فرم زير مي باشد.

```
// Demo package p1.
```

```
package p1;
// Instantiate the various classes in p1.
public class Demo {
public static void main(String args[] ){
Protection2 ob1 = new Protection2();
OtherPackage ob2 = new OtherPackage();
}
}
```

سازنده گان(Constructors)

براي مقدار دهي متغيير هاي يك كالس مي توانيم آنها را تك تك مقدار دهي اوليه كنيم.اما اين روش مشكل مي باشد حتي اگر براي مقداري دهي متغير هاي يك كلاس تابعي را ايجاد كنيم، در زمان اجرا بايد خود كد مربوط به فراخواني آن را بنويسيد. جاوا به اشياء اين امكان را مي دهد تا در زمان ايجاد شدن خودشان را مقدار دهي اوليه كنند.

اين مقدار دهي اوليه خودكار با استفاده از سازنده انجام ميگيرد.سازنده به محض ايجاد يك شي آن را مقدار دهي مي كند و نام آن معادل با نام همان كلاس مي باشد. هنگامي كه يك كلاس سازنده دارد، سازنده در زمان تعريف استفاده از كلاس و قبل از تكميل new فراخواني مي شود. دقت داشته باشيد كه سازنده هيچ نوع بازگشتي حتي void ندارد واين به اين دليل است كه مقدار باز گشتي سازنده يك كلاس از نوع خود كلاسي كه در آن كامل شده است.

در زير مثال مربوط به كلاس Box را به همراه سازنده آن تكميل تر كرده ايم.

```
/* Here/ Box uses a constructor to initialize the
dimensions of a box.
*/
class Box {
double width;
double height;
double depth;
// This is the constructor for Box.
Box (){
System.out.println("Constructing Box");
width = 10;
height = 10;
depth = 10;
}
// compute and return volume
double volume (){
return width * height * depth;
} }
class BoxDemo {
public static void main(String args[] ){
// declare/ allocate/ and initialize Box objects
Box mybox1 = new Box();
Box mybox2 = new Box();
```

```
double vol;
// get volume of first box
vol = mybox1.volume();
System.out.println("volume is " + vol);
// get volume of second box
vol = mybox2.volume();
System.out.println("volume is " + vol);
}
 }
```

پس از اجراي اين برنامه خروجي به شكل زير خواهد بود.

```
Constructing Box
Constructing Box
Volume is 1000
Volume is 1000
```

همان طوري كه مشاهده مي كنيد mybox1 و mybox2 هنگام ايجاد شدن به وسيله Box() مقدار دهي اوليه مي شوند و چون سازنده به همه اشياء اينجاد شده از نوع Box مقدار ابعاد اوليه 10*10*10 را مي دهد هر دو شي تعريف شده ابعاد يكسني خواهند داشت.

متد println()در داخل Box فقط براي به منظور توصيف بهتر قرار گرفته. اكثر متدهاي سازنده چيزي را نشان نمي دهند و فقط وظيفه دادن مقادير اوليه را دارند. همان طوري كه مي دانيد استفاده از عملگر new در تخصيص يك شي ايجاد شده به فرم زير مي باشد.

```
class-var = new classname();
```

اكنون مي فهميد كه چرا پرانتزهاي بعد از نام كلاس مورد نياز مي باشد و چيزي كه واقعآ اتفاق مي افتد به فرم Box mybox1=new Box(); است. عملگر new سازنده را فراخواني ميكند و اگر سازنده اي براي كلاس تعريف نشده باشد جاوا از سازنده پيش فرز براي اين كار استفاده مي كند. سازنده پيش فرز به طور خود كار كليه متغيرهاي نمونه ار با عدد صفر مقدار دهياوليه مي كند. اين سازنده براي كلاسهاي ساده كفايت مي كند وبراي كلاسهاي پيچيده بايد سازنده تعريف كنيد.

اين قابليت وجود دارد كه به سازنده پارامتر اضافه كنيم و در زمان ايجاد به طور پويا مقدار دهي اوليه كنيم. اين عمل مفيد بودن سازنده را افزايش مي دهد.

در زير مثال Box را با سازنده پارامتردار مشاهده مي كنيد.

```
/* Here/ Box uses a parameterized constructor to
initialize the dimensions of a box.
*/
class Box {
double width;
double height;
double depth;
// This is the constructor for Box.
Box(double w/ double h/ double d ){
width = w;
height = h;
depth = d;
}
```

```
// compute and return volume
double volume (){
return width * height * depth;
}
}
class BoxDemo {
public static void main(String args[] ){
// declare/ allocate/ and initialize Box objects
Box mybox1 = new Box(10/ 20/ 15);
Box mybox2 = new Box(3/ 6/ 9);
double vol;
// get volume of first box
vol = mybox1.volume();
System.out.println("volume is " + vol);
// get volume of second box
vol = mybox2.volume();
System.out.println("volume is " + vol);
}
 }
```

خروجي برنامه به شكل زير خواهد بود.

```
Volume is 3000
Volume is 162
```

وراثت در جاوا

به كمك وراثت مي توان يك كلاس عمومي ساخته و ويژگي هاي كه در يك مجموعه از اغلام مشترك مي باشند را در داخل آن قرار داده و سپس از اين كلاس پايه براي ايجاد ساير كلاسها كه تفاوت جزئي با ساير كلاسها دارند استفاده كنيد. در وراثت كلاس به ارث برنده چيزهاي را كه منحصر به فرد خودش باشد را به آن اضافه مي كند.در جاوا كلاسي كه به ارث برده مي شود را ابر كلاس (superclass) مي نامند. كلاسي راكه عمل ارث بري را انجام داده را زير كلاس (subclass) مي نامند.

بنابراين زير كلاس روايت تخصصي تر و مشخص تر از كلاس بالاتر يعني ابر كلاس است.زير كلاس كليه متغير ها و متد هاي ابر كلاس را به متد ها و متغيرهاي منحصر به فرد خود اضافه مي كند.

براي ارث بردن از يك كلاس، خيلي ساده كافي است كه يك كلاس را با استفاده از واژه كليدي extend در كلاس ديگر قرار دهيد. براي فهم كامل اين مطلب يك مثال ساده را نشان مي دهيم. در اينجا يك ابر كلاس به نام A ويك زير كلاس به نام B ايجاد مي كنيم. دراستفاده از واژه extend دقت فرماييد.

```
// A simple example of inheritance.
// Create a superclass.
class A {
int i, j;
void showij (){
System.out.println("i and j :" + i + " " + j);
} }
// Create a subclass by extending class A.
class B extends A {
```

```
int k;
void showk (){
System.out.println("k :" + k);
}
void sum (){
System.out.println("j+j+k :" +( i+j+k));
} }
class SimpleInheritance {
public static void main(String args[] ){
A superOb = new A();
B subOb = new B();
// The superclass may be used by itself.
superOb.i = 10;
superOb.j = 20;
System.out.println("Contents of superOb :");
superOb.showij();
System.out.println();
/* The subclass has access to all public members of
its superclass .*/
subOb.i = 7;
subOb.j = 8;
subOb.k = 9;
System.out.println("Contents of subOb :");
subOb.showj();
subOb.showk();
System.out.println();
System.out.println("Sum of i/ j and k in subOb:");
subOb.sum();
}
 }
```

خروجي اين برنامه به شكل زير مي باشد.

```
Contents of superOb:
i and j :10 20
Contents of subOb:
i and j :7 8
k :9
Sum of i/ j and k in subOb:
i+j+k :24
```

همان طور كه مشاهده مي كنيد زير كلاس B در بر گيرنده كليه اعضاي كلاس A است. به همين دليل است كه subclass مي تواند به i و j دسترسي داشته باشد و showij() را فرا خواني نمايد. همچنين در داخل متد sum() مي توان به طور مستقيم به i و j راكه بخشي از B بودند ارجاع نمود. ابر كلاس بودن به معني اين نست كه نمي توان خود آن ابر كلاس را به تنهاي استفاده كرد و همانند يك كلاس عادي با آن بر خورد مي شود. نكته ديگري كه وجود دارداين كه يك زير كلاس مي تواند ابر كلاس يك زير كلاس باشد.
شكل عمومي اعلان يك كلاس كه از يك ابر كلاس ارث مي برد به فرم زير مي باشد.

```
class subclass-name extends superclass-name {
// body of class
}
```

اگر چه يك زير كلاس در برگيرنده كليه اعضاء ابر كلاس خود مي باشد اما نمي تواند به اعضائي از كلاس بالا كه به عنوان private اعلان شده اند، دسترسي داشته باشد.
به عنوان مثال سلسله مراتب ساده كلاس زير را در نظر بگيريد.

```
/* In a class hierarchy/ private members remain
private to their class.
This program contains an error and will not
compile.
*/
// Create a superclass.
class A {
int i; // public by default
private int j; // private to A
void setij(int x/ int y ){
i = x;
j = y;
} }
// A's j is not accessible here.
class B extends A {
int total;
void sum (){
total = i + j; // ERROR/ j is not accessible here
} }
class Access {
public static void main(String args[] ){
B subOb = new B();
subOb.setij(10, 12);
subOb.sum();
System.out.println("Total is " + subOb.total);
}
 }
```

اين برنامه كاملتر نخواهد شد زيرا ارجاع به j در داخل متد sum() در B سبب خطا در دسترسي خواهد شد. از آنجاي كه j به عنوان private اعلان شده، فقط توسط ساير اعضاي كلاس خود قابل دسترسي است و زير كلاسهاي آن هيچ گونه دسترسي به آن ندارند. همان طور كه قبلاً اعلان كرديم عضو از كلاس كه توسط private اعلان شده براي كلاس خود اختصاصي شده و براي كدهاي خارج از خود قابل دسترس نيست.

براي نشان دادن قدرت واقعي وراثت يك مثال را بررسي خواهيم كرد. در اين جا كلاس Box به نحوي گسترس يافته تا يك عنصر چهارم تحت نام weight را در بر گيرد و كلاس جديد شامل weight, depth, height, width و يك box خواهد بود.

```
// This program uses inheritance to extend Box.
class Box {
double width;
```

```
double height;
double depth;
// construct clone of an object
Box(Box ob ){ // pass object to constructor
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w/ double h/ double d ){
width = w;
height = h;
depth = d;
}
// constructor used when all dimensions specified
Box (){
width =- 1; // use- 1 to indicate
height =- 1; // an uninitialized
depth =- 1; // box
}
// compute and return volume
double volume (){
return width * height * depth;
}
}
// Here/ Box is extended to include weight.
class BoxWeight extends Box {
double weight; // weight of box
// constructor for BoxWeight
BoxWeight(double w/ double h/ double d/ double m ){
width = w;
height = h;
depth = d;
weight = m;
} }
class DemoBoxWeight {
public static void main(String args[] ){
Boxweight mybox1 = new BoxWeight(10/ 20/ 15/ 34.3);
Boxweight mybox2 = new BoxWeight(2/ 3/ 4/ 0.076);
double vol;
vol = mybox1.volume();
System.out.println("Volume of mybox1 is " + vol);
System.out.println("Weight of mybox1 is " + mybox1.weight);
System.out.println();
vol = mybox2.volume();
System.out.println("Volume of mybox2 is " + vol);
System.out.println("Weight of mybox2 is " + mybox2.weight);
}
```

}

نتيجه خروجي به فرم زير خواهد بود.

```
Volume of mybox1 is 3000
Weight of mybox1 is 34.3
Volume of mybox2 is 24
Weight of mybox2 is 0.076
Boxweight
```

كلاس Boxeight كليه متدها و متغير هاي كلاس Box را به ارث برده و متغير weight را به آنها اضافه مي كند. براي زير كلاس ضرورتي ندارد كه اعضاء موجود در كلاس بالاتر يعني ابر كلاس را مجدداً پياده سازي كند. از مزيتهاي اصلي وراثت اين است كه كافي است فقط يك بار يك ابر كلاس را ايجاد كرده و در داخل آن خصلتهاي مشترك را قرار داده آنگاه از آن براي ايجاد هر تعداد زير كلاس استفاده كرد.
بطور مثال در زير يك كلاس از كلاس Box به ارث برده شده و خصلت رنگ را به آن اضافه كرده.

```
// Here/ Box is extended to include color.
class ColorBox extends Box {
int color; // color of box
ColorBox(double w/ double h/ double d/ double c ){
width = w;
height = h;
depth = d;
color = c;
}
}
```

يك متغير ابر كلاس مي تواند به يك شي زير كلاس ارجاع كند. يك متغير ارجاع مربوط به يك كلاس بالا را مي توان به ارجاعي، به هر يك از زير كلاسهاي مشتق شده از آن كلاس بالا انتساب نمود. در بسياري از شرايط اين جنبه از وراثت كاملاً سودمند است و به عنوان نمونه مثال زير را در نظر بگيريد.

```
class RefDemo {
public static void main(String args[] ){
Boxweight weightbox = new BoxWeight(3/ 5/ 7/ 8.37);
Box plainbox = new Box();
double vol;
vol = weightbox.volume();
System.out.println("Volume of weightbox is " + vol);
System.out.println("Weight of weightbox is " +
weightbox.weight);
System.out.println();
// assign BoxWeight reference to Box reference
plainbox = weightbox;
vol = plainbox.volume(); // OK/ volume ()defined in Box
System.out.println("Volume of plainbox is " + vol);
/* The following statement is invalid because plainbox
dose not define a weight member .*/
```

```
// System.out.println("Weight of plainbox is " + plainbox.weight
}
 }
```

در اينجا weightbox يك ارجاع به شي Boxweight است و plainbox يك ارجاع به اشيائ Box است و مي توان plainbox را به عنوان يك ارجاع به شي weightbox انتساب نمود. نكته مهم كه در اين جا قابل بيان است اين كه نوع متغير ارجاع و نه نوع شيئ كه به آن اشاره شده كه تعين مي كند كدام اعضاء قابل دسترس هستند. اين به اين معني است كه هنگامي كه يك ارجاع مربوط به يك شي زير كلاس، به يك متغير ارجاع ابر كلاس انتساب مي شود، شما فقط به بخشهاي از شي دسترسي داريد كه توسط ابر كلاس تعريف شده باشد.

براي همين دليل مي باشد كه plainbox نمي تواند به weight دسترسي داشته باشد چون ابر كلاس نمي تواند آگاهي به موارد اضافه شده به زير كلاس داشته باشد. براي همين دليل آخرين خط كد در توضيح درج شده.

ايجاد سلسه مراتب چند سطحي (Multilevel)

مي توان سلسه مراتبي ساخت كه كه شامل چندين لايه وراثت باشد. كاملآ قابل توجه است كه كه از يك زير كلاس هم مي توان زير كلاسهي ديگري را نيز بدست آورد. به عنوان مثال اگر سه كلاس A,B,C داشته باشيم انگاه C مي تواند يك زير كلاس از B و خود B يك زير كلاس از A باشد و C كليه خصلتهاي موجود در A,B را در خود دارد.

در مثال بعدي زير كلاس boxweight به عنوان يك ابر كلاس استفاده شده تا زير كلاسهاي تحت عنوان shipment را ايجاد كند. كلاس shipment كليه خصلتهاي Boxweight و Box را به ارث برده و يك فيلد به نام cost به آن االفه شده كه هزينه كشتيراني يك محلموله را نگهداري ميكند.

```
// Extend BoxWeight to include shipping costs.
// Start with Box.
class Box {
private double width;
private double height;
private double depth;
// construct clone of an object
Box(Box ob ){ // pass object to constructor
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w, double h, double d ){
width = w;
height = h;
depth = d;
 }

 // constructor used when no dimensions specified
 Box (){
```

```
 width =- 1; // use- 1 to indicate
 height =- 1; // an uninitialized
depth =- 1; // box
}
// constructor used when cube is created
Box(double len ){
width = height = depth = len;
}
// compute and return volume
double volume (){
return width * height * depth;
}
}
// Add weight.
class BoxWeight extends Box {
double weight; // weight of box
// construct clone of an object
BoxWeight(BoxWeight ob ){ // pass object to constructor
super(ob);
weight = ob.weight;
}
// constructor used when all parameters are specified
BoxWeight(double w/ double h/ double d/ double m ){
super(w, h, d); // call superclass constructor
weight = m;
}
// default constructor
BoxWeight (){
super();
weight =- 1;
}
// constructor used when cube is created
BoxWeight(double len/ double m ){
super(len);
weight = m;
}
}
// Add shipping costs
class Shipment extends BoxWeight {
double cost;
// construct clone of an object
Shipment(Shipment ob ){ // pass object to constructor
super(ob);
cost = ob.cost;
}
// constructor used when all parameters are specified
BoxWeight(double w, double h, double d,
double m, double c ){
```

```
super(w, h, d); // call superclass constructor
cost = c;
}
// default constructor
Shipment (){
super();
cost =- 1;
}
// constructor used when cube is created
BoxWeight(double len, double m, double c ){
super(len, m);
cost = c;
}
}
class DemoShipment {
public static void main(String args[] ){
Shipment shipment1 = new Shipment(10, 20, 15, 10, 3.41);
Shipment shipment2 = new Shipment(2, 3, 4, 0.76, 1.28);
double vol;
vol = shipment1.volume();
System.out.println("Volume of shipment1 is " + vol);
System.out.println("Weight of shipment1 is " + shipment1.weight);
System.out.println("Shipping cost :$" + shipment1.cost);
System.out.println();
vol = shipment2.volume();
System.out.println("Volume of shipment2 is " + vol);
System.out.println("Weight of shipment2 is " + shipment2.weight);
System.out.println("Shipping cost :$" + shipment2.cost);
}
 }
```

خروجي اين برنامه بصورت زير مي باشد:

```
Volume of shipment1 is 3000
Weight of shipment1 is 10
Shipping cost :$3.41
Volume of shipment2 is 24
Weight of shipment2 is 0.76
Shipping cost :$1.28
```

وراثت امكان استفاده مجدد از كدهاي قبلي را به خوبي به وجود مي آورد يعني ديگر نياز به تكرار كد متدها در داخل زير كلاسها نداريم. نكته مهم ديگر اين كه متد super() همواره به سازنده موجود در نزديكترين ابر كلاس ارجاع مي كند. متد super() در shipment ، سازنده Boxweight را فراخواني مي كند. Super() در Boxweight سازنده موجود در Box را فراخواني مي كند.
در يك سلسه مراتب كلاس، اگر يك سازنده كلاس بالا نيازمند پارامترها باشد، آگاه كليه زير كلاسها بايد ان پارامترها را بالاي خط (up the line) بگذارند. اين امر چه يك زير كلاس پارامترهاي خودش را نياز داشته باشد و چه نياز نداشته باشد، صحت خواهد داشت.

توجه داشته باشيد كه در مثال گفته شده فقط براي راحتي شما هر سه كلاس در داخل يك فايل قرار گرفته و در جاوا هر كدام از كلاسها بايد در فايلهاي خاص خودشان قرار گرفته و جداگانه كامپايل شوند.

وقتي كه سلسه مراتب كلاس ايجاد مي شود سازنده گان كلاسها سلسله مراتب را تشكيل مي دهند. سازنده گان به ترتيب مشتق شدنشان از ابر كلاس به زير كلاس فراخواني مي شوند. چون متد super() بايد اولين دستوري باشد كه در يك سازنده زير كلاس اجرا مي شود، اين ترتيب هماطور حفظ مي شود، خواه متد super() استفاده شود يا نه. اگر super() استفاده نشود انگاه سازنده پيش فرض يا سازنده بدون پارامتر هر يك از زير كلاسها اجرا خواهند شد.
برنامه زير نشان مي دهد كه چه زماني سازنده ها اجرا مي شود.

```
// Demonstrate when constructors are called.
// Create a super class.
class A {
A (){
System.out.println("Inside A's constructor.")
} }
// Create a subclass by extending class A.
class B extends A {
B (){
System.out.println("Inside B's constructor.")
} }
// Create another subclass by extending B.
class C extends B {
C (){
System.out.println("Inside C's constructor.")
} }
class CallingCons {
public static void main(String args[] ){
C c = new C();
}
 }
```

خروجي به فرم زير مي باشد:

```
Inside A's constructor
Inside B's constructor
Inside C's constructor
```

همان طوري كه ملاحظه كرديد سازنده گان كلاسها به ترتيب مشتق شدنشان فراخواني مي شوند زيرا يك ابر كلاس نسبت به زير كلاسهاي خود آگاهي ندارد، هرگونه مقدار دهي اوليه كه براي اجرا شدن نياز داشته باشد، جدا از هر گونه مقدار دهي اوليه انجام شده توسط زير كلاس است.

متد super()

در جاوا امكان اين وجود دارد يك ابر كلاس را ايجاد كرده و در آن جزئيات پياده سازي خود را در خودش نگهداري كند. در اين شرايط راهي براي زير كلاس وجود ندارد تا مستقيماً به اين متغير ها مربوط به خودش دسترسي داشته باشد و يا آنها را مقدار دهي اوليه كند. از انجاي كه كپسوله سازي يك خصلت اوليه در شي گراي است، اين باعث تعجب نخواهد بود كه جاوا راه حل را با كمك متد ()super حل كرده. هر زمان كه لازم باشد تا يك زير كلاس به ابر كلاس قبلي خود ارجاع كند با استفاده از ()super اين كار را انجام مي دهد.
متد ()super دو شكل عمومي دارد، اولين شكل سازنده ابر كلاس را فراخواني ميكند. دومين شكل به منظور دسترسي به يك عضو ابر كلاس كه توسط يك عضو زير كلاس مخفي مانده استفاده مي شود.
يك زير كلاس مي تواند متد سازنده تعريف شده توسط ابر كلاس مربوطه را با استفاده از فرم super فراخواني مي كند:

super(parameter-list);

در اين متد parameter-list مشخص كننده هر پارامتري است كه توسط سازنده در ابر كلاس مورد نياز مي باشد. اين متد بايد همواره اولين دستور اجرا شده داخل يك سازنده زير كلاس باشد.به نحوه استفاده از اين متد در مثال زير دقت كنيد.

```
// BoxWeight now uses super to initialize its Box attributes.
class BoxWeight extends Box {
double weight; // weight of box
// initialize width/ height/ and depth using super()
BoxWeight(double w/ double h/ double d/ double m ){
super(w, h, d); // call superclass constructor
weight = m;
}
}
```

متد ()Boxweight فراخواني ()super را با پارامتر هاي d,h و w انجام ميدهد. اين كار سبب ميشود تا سازنده ()Box فراخواني شده و با استفاده از مقادير depth ,height و width مقداري دهي اوليه كند و ديگر Boxweight خود اين مقدار دهي اوليه را انجام نمي دهد. فقط لازم است تا مقدار دهي منحصر به فرد خود weight را مقدار دهي دهي اوليه نمايد. انجام اين عمل Box را آزاد مي گذارد تا در صورت تمايل اين مقادير را private بسازد.
در مثال قبلي متد()super با سه پارامتر فراخواني شد. اما چون سازندگان ممكن است انباشته شوند، مي توان متد ()super را با استفاده از هر شكل تعريف شده توسط ابر كلاس فراخواني كرد. سازنده اي كه اجرا مي شود همان سازنده اي است كه با پارامترها مطابقت دارد.
به طور مثال در اين جا يك پياده سازي كامل از كلاس Boxweight موجود است كه سازنده گان را براي روش هاي گوناگون ممكن ساخته شدن يك box فراهم ميكند. در هر حالت متد ()super با استفاده از پارامترهاي تقزيبي فراخواني مي شود. دقت كنيد كه depth ,height و width در داخل Box به صورت اختصاصي در آمده اند.

```
// A complete implementation of BoxWeight.
class Box {
private double width;
private double heght;
private double deoth;
// construct clone of an object
Box(Box ob ){ // pass object to constructor
```

```
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w/ double h/ double d ){
width = w;
height = h;
depth = d;
}
// constructor used when no dimensions specified
Box (){
width =- 1; // use- 1 to indicate
height =- 1; // an uninitialized
depth =- 1; // box
}
// constructor used when cube is created
Box(double len ){
width = height = depth = len;
}
// compute and return volume
double volume (){
return width * height * depth;
}
}
// BoxWeight now fully implements all construtors.
class BoxWeight extends Box {
double weight; // weight of box
// construct clone of an object
BoxWeight(BoxWeight ob ){ // pass object to consructor
super(ob);
weight = ob.weight;
}
// constructor used when all parameters are specified
Box(double w, double h, double d, double m ){
super(w, h, d); // call superclass constructor
weight = m;
}
// default constructor
BoxWeight (){
super();
weight =- 1;
}
// constructor used when cube is created
BoxWeight(double len, double m ){
super(len);
weight = m;
}
```

```
}
class DemoSuper {
public static void main(String args[] ){
BoxWeight mybox1 = new BoxWeight(10 ,20 ,15 ,34.3);
BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
BoxWeight mybox3 = new BoxWeight(); // default
BoxWeight mycube = new BoxWeight(3, 2);
BoxWeight myclone = new BoxWeight(mybox1);
double vol;
vol = mybox1.vilume();
System.out.println("Volume of mybox1 is " + vol);
System.out.println("Weight of mybox1 is " + mybox1.weight);
System.out.println();
vol = mybox2.vilume();
System.out.println("Volume of mybox2 is " + vol);
System.out.println("Weight of mybox2 is " + mybox2.weight);
System.out.println();
vol = mybox3.vilume();
System.out.println("Volume of mybox3 is " + vol);
System.out.println("Weight of mybox3 is " + mybox3.weight);
System.out.println();
vol = myclone.vilume();
System.out.println("Volume of myclone is " + vol);
System.out.println("Weight of myclone is " + myclone.weight);
System.out.println();
vol = mycube.vilume();
System.out.println("Volume of mycube is " + vol);
System.out.println("Weight of mycube is " + mycube.weight);
System.out.println();
}
}
```

خروجي اين برنامه به شكل زير خواهد بود:

```
Volume of mybox1 is 3000
Weight of mybox1 is 34.3
Volume of mybox2 is 24
Weight of mybox2 is 0.076
Volume of mybox3 is- 1
Weight of mybox3 is- 1
Volume of myclone is 3000
Weight of myclone is 34.3
Volume of mycube is 27
Weight of mycube is 2
```

سازنده در كلاس Boxweight كه به شكل زير است:

```
// construct clone of an object
BoxWeight(BoxWeight ob ){ // pass object to constructor
```

```
super(ob);
weight = ob.weight;
}
```

به نكته توجه كنيد كه متد super() با يك شي از نوع Boxwieght و نه از نوع Box فراخواني شده است سازنده كلاس Box يعني Box(Box ob) را فراخواني مي كند. همان طوري كه مي دانيد يك متغير ابر كلاس را مي توان براي ارجاع به هر شئي مشتق شده ار آن كلاس مورد استفاده قرار داد. براي همين دليل مي باشد كه ما قادر هستيم يك شي Boxweight را به سازنده Box پاس كنيم و البته Box نسبت به اعضاي خود آگاهي دارد. زماني كه يك زير كلاس متد super() را فراخواني مي كند، در اصل اين متد به ابر كلاس كلاسي كه از آن فراخواني شده ارجاع مي كند. اين امر حتي در يك سلسله مراتب چند سطحي هم نيز صادق است و دقت داشته باشيد كه متدsuper() بايد اولين دستوري باشد كه در داخل يك سازنده زير كلاس اجرا مي شود.

دومين كاربرد super همانند this عمل مي كند با اين تفاوت كه super به ابر كلاس زير كلاسي كه در آن استفاده ميشود اشاره دارد و فرم عمومي استفاده از به صورت زير مي باشد.

Super .member

در اينجا member مي تواندیك متد يا يك متد در ابر كلاس باشد.اين روش زماني كاربرد پيدا مي كند كه در آن اسامي اعضائ يك زير كلاس، اعضائ با هما اسامي را در ابر كلاس مخفي مي سازد.

اين سلسه مراتب ساده كلاس را در نظر بگيريد:

```
// Using super to overcome name hiding.
class A {
int i;
}
// Create a subclass by extending class A.
class B extends A {
int i; // this i hides the in A
B(int a,int b ){
super.i = a; // i in A
i = b; // i in B
}
void show (){
System.out.println("i in superclass :" + super.i);
System.out.println("i in subclass :" + i);
} }
class UseSuper {
public static void main(String args[] ){
B subOb = new B(1,2);
subOb.show();
}
 }
```

بعد از اجراي اين برنامه خروجي زير نمايش داده خواهد شد.

```
i in superclass :1
i in subclass :2
```

اگر چه متغير نمونه i در B متغير نمونه I در A را پنهان مي كند اما super امكان دسترسي به I تعريف شده در كلاس بالا به وجود مي آورد. همچنين ميتوان از super مي توان براي فرخواني متدهاي كه توسط زير كلاس مخفي شده اند استفاده كرد.

متد finalize()

در بعضي از مواقع لازم است كه تا يك شئ هنگامي كه در حال خراب شد است يك عمل خاص را انجام دهد. خراب شدن شي زماني است كه در اتمام استفاده از شي قصد برداشتن آن از حافظه و حذف اشاره گر به آن را داريم. به عنوان مثال ممكن است كه يك شي در برگيرنده منابع غير جاوا نظر يك دستگير فايل (file handle) يا فونت كاراكتر ويندوز باشد، و ميخواهيد اطمينان پيدا كنيد كه قبل از اينكه آن شي خراب شود منابع فوق رها شوند.
براي اداره چنين شرايطي جاوا مكانيزمي تحت نام تمام كننده (finalization) را فراهم آورده. به كمك اين مكانيزم مي توان عمليات مشخصي تعريف نمايد كه كه در زماني كه اشغال جمع كن(garbage collector) جاوا قصد گرفتن حافظه آن شي و افزودن آن به به فضاي آزاد سيستم را دارد آن عمليات اتفاق بيافتد. براي افزون يك تمام كننده به كلاس كافي است كه متد finalize() را به كلاس خود اضافه كنيد. شكل كلي استفاده از متد finalize() به فرم زير است.

```
protected void finalize()
{
// finalization code here
}
```

توصيفگر protected براي اين مي باشد كه از دسترسي به متد finalize() توسط كدهاي تعريف شده خارج از كلاس جلوگيري كند. توجه كنيد كه متد finalize() قبل از شروع كار اشغال جمع كن جاوا بر روي آن شئ اجرا مي شود. اگر يك شئ اشاره گر به آن گم شود، بر اثر اشتباه در برنامه شما و آن شي از قلمرو خارج شود نمي توان تشخيص داد كه متد finalize() چه زماني اجرا خواهد شد. بنابراين برنامه شما بايد ساير وسايل براي آزاد كردن منابع سيستم مورد استفاده شئ را تدارك ببينند.
برنامه شما نبايد فقط متكي به finalize() باشد. البته جاوا اين پيش بيني را كرده و يك يك سيستم جمع كردن اين اشياء در خود ايجاد كرده.

استفاده از this

در زمانهاي لازم مي باشد كه يك روش به شئي كه آنرا فراخواني مي كند ارجاع كند. براي اين كار جاوا واژه كليدي this را تعريف كرده است. به كمك this مي توان در داخل هر متد به يك ارجاع به شئ جاري داشت. اين به اين معني مي باشده كه this همواره ارجاعي به شئي است كه متد روي آن فراخواني شده است. ميتوانيد از this هر جاي كه ارجاع به يك شئي از نوع كلاس جاري مجاز مي باشد استفاده كرد.
براي فهم بهتر در مورد كار this روايت بعدي كلاس Box را در نظر بگيريد.

```
// A redundant use of this.
Box(double w/ double h/ double d ){
this.width = w;
this.height = h;
this.depth = d;
}
```

اين روايت از سازنده دقيقآ مثل روايت اوليه كار مي كند. استفاده از this به ندرت انجام مي گيرد. در داخل Box() همواره this به شئي فراخوانده شده ارجاع مي كند.

مخفي نمودن متغير نمونه

حتمآ مي دانيد كه اعلان دو متعير محلي با يك نام داخل يك قلمرو يا قلمرو هاي بسته شده، غير مجاز است. به طور جالبي مي توان متغيرهاي محلي شامل پارامترهاي رسمي براي متدها،داشته باشيد كه با اسامي متغيرها نمونه كلاسها مشترك باشند.اما هنگامي كه يك متغير محلي همان اسم متغير نمونه را داشته باشد، متغير محلي متغير نمونه را مخفي مي سازد. براي همين دليل بود كه از depth ,height و weight به عنوان اسامي پارامترهاي سازنده Box() داخل كلاس Box استفاده نكرديم. اگر از آنها به همين روش استفاده مي كرديم، آنگاه width به پارامترهاي رسمي ارجاع مي كرد و متغير نمونه width را مخفي نگه مي داشت.

اگر چه آسان تر مي باشد كه از اسامي متفاوت استفاده كنيم، اما راه حل ديگري براي چنين شرايطي وجود دارد . چون this امكان ارجاي مستقيم به شيئ را به شما مي دهد، مي توانيد با استفاده از آن هر نوع تصادف و يكي شدن اسامي محلي را رفع كنيد. به عنوان مثال ، دراين جا روايت ديگري از Box() وجود دارد كه از depth ,height و weight به عنوان اسامي پارامتر ها استفاده نموده و انگاه از this براي دستيابي به متغير هاي نمونه با همين اسامي استفاده كرده است:

```
// Use this to resolve name-space collisions.
Box(double width/ doublee height/ double depth ){
this.width = width;
this.height = height;
this.depth = depth;
}
```

دقت داشته باشيد كه استفاده از this در چنين متني ممكن است گاهي گيج كننده شود و برخي برنامه نويسان مراقب هستند تا از اسامي متغير هاي محلي و پارمترهاي رسمي كه متغيرهاي نمونه را مخفي مي سازند استفاده نكنند. البته ساير برنامه نويسان متعقدند استفاده اسامي مشرك براي وضوح مفيد بوده و از this براي غلبه بر مخفي سازي متغيرهاي نمونه كمك مي گيرند. انتخاب يكي از دو روش با سليقه شما ارتباط دارد.

استفاده از كلاسهاي انتزاعي (abstract)

در جاوا اين امكان وجود دارد كه يك ابر كلاس تعريف نموده كه ساختار يك انتزاع معين را بدون يك پياده سازي كامل از هر روشي اعلان نمايد.اين به اين معني است كه مي خواهيد يك ابر كلاس ايجاد كنيد كه فقط يك شكل عمومي شده تعريف كند كه توسط كليه زير كلاسها به اشتراك گذاشته خواهد شد و پر كردن جزئيات اين شكل عمومي

به عهده هر يك از زير كلاسها واگذار مي شود. وشيوه براي وقوع اين شرايط زماني است كه يك ابر كالس تواناي ايجاد يك پياده سازي با معني براي يك متد را نداشته باشد.
تعريف متد area() خيلي ساده يك نگهدارنده مكان است. اين متد روش مساحت انواع شي را محاسبه نكرده ونمايش نمي دهد. هنگام ايجاد كتابخانه هاي خاص كلاس خود اين غير معمول نيست كه اگر يك متد هيچ تعريف با معني در متن ابر كلاس خود نداشته باشد. جاوا براي حل از متد انتزاعي (abstract method) استفاده مي كند. مي توانيد توسط زير كلاسها و با مشخص كردن اصلاح كننده نوع abstract متدهاي معني را لغو كنيد.گاهي به اين متدها جوابگويي زير كلاس (subclass responsibility) مي گويند زيرا آنها هيچ پياده سازي مشخص شده اي در ابر كلاس ندارند. بنا براين يك زير كلاس بايد انها را لغو نمايد چون نمي تواند به سادگي روايت تعريف شده در ابر كلاس را استفاده نمايد.
براي اعلام يك متد انتزاي از شكل عمومي زير استفاده مي كنيم.

```
abstract type name( parameter-list);
```

همان طور كه ملاحظه مي كنيد در اين جا بدنه متد اعلان نشده . هر كلاسي كه يك يا چند متد انتزاعي دارد بايد به عنوان انتزاعي اعلام گردد. براي اعلان يك كلاس به عنوان انتزاعي از واژه كليدي abstract در جلوي واژه كليدي class در ابتداي تعريف كلاس مي كنيم. برا ي يك كلاس انتزاعي هيچ شئي نم توتن ايجاد نمود. يعني يك كلاس انتزاعي نبايد بطور مستقيم با با عملگر new نمونه سازي شود. اين به خاطر اين مي باشد كه يك كلاس انتزاعي بطور كامل تعريف نشده است. همچنين نمي توانيد سازنده گان انتزاعي يا متدهاي ايستاي انتزاعي اعلان كنيد.
هر زير كلاس از يك كلاس انتزاعي بايد يا كليه متدهاي انتزاعي موجود در ابر كلاس را پياده سازي كند و يا خودش به عنوان يك abstract اعلان شود. در اينجا مثال ساده اي از يك كلاس با يك روش انتزاعي مشاهد مي كنيد كه بعد از آن يك زير كلاس قرار گرفته كه آن متد را پياده سازي مي كند.

```
// A Simple demonstration of abstract.
abstract class A {
abstract void callme();
// concrete methods are still allowed in abstract classes
void callmetoo (){
System.out.println("This is a concrete method.");
} }
class B extends A {
void callme (){
System.out.println("B's implementetion of callme.");
} }
class AbstractDemo {
public static void main(String args[] ){
B b = new B();
b.callme();
b.callmetoo();
}
}
```

توجه دشته باشيد كه هيچ شيئ از A در برنامه اعلان نشده است. همان طوري كه قبلآ گفته شد امكان نمونه سازي يك كلاس انتزاعي وجود ندارد. نكته اي كه بايد توجه داشته باشيد اين كه كلاس A يك متد واقعي با نام callmtoo() راپياده سازي مي كند.

اين امر كاملاً قابل قبول است. كلاسهاي انتزاعي مي توانند تا زماني كه تناسب را حفظ نمايند، در بر گيرنده پياده سازي ها باشند.
اگر چه نمي توان از كلاسهاي انتزاعي براي نمونه سازي اشياء استفاه كرد اما مي توان از آنها براي ايجاد ارجاعات براي اشياء استفاه كرد زيرا روش جاوا از طريق استفاده از ارجاعات ابر كلاس پياده سازي خواهد شد. بنابراين بايد امكان ايجاد يك ارجاع به يك كلاس انتزاعي وجود داشته باشد بطوري كه با استفاده از آن ارجاع به يك شيئ زير كلاس اشاره نمود. شما در كلاس بعدي اين موضوع را مشاهده خواهيد كرد. با استفاده از از يك كلاس انتزاعي ميتوانيد كلاس Figure() را توسعه دهيد. چون مفهوم با معناي براي مساحت يك شكل دو بعدي تعريف نشده وجود ندارد، روايت بعدي اين برنامه area() را به عنوان يك انتزاع داخل كلاس Figure اعلان مي كند. البته اين بدان معني است كه كليه كلاسهاي مشتق شده از Figure بايد متد area() را لغو كنند.

```
// Using abstract methods and classes.
abstract class Figure {
double dim1;
double dim2;
Figure(double a/ double b ){
dim1 = a;
dim2 = b;
}
// area is now an abstract method
abstruct double area();
}
class Rectangle extends Figure {
Rectangle(duoble a,double b ){
super(a,b);
}
// override area for rectangle
double area (){
System.out.println("Inside Area for Rectangle.");
return dim1 * dim2;
}
}
class Triangle extends Figure {
Triangle(double a,double b ){
super(a,B);
}
// override area for right triangle
double area (){
System.out.println("Inside Area for Teriangle.");
return dim1 * dim2 / 2;
}
}
class AbstractAreas {
public static void main(String args[] ){
// Figure f = new Figure(10/ 10); // illegal now
```

```
Rectangle r = new Rectanlge(9/ 5);
Triangle t = new Triangle(10/ 8);
Figure figref; // this is OK/ no object is created
figref = r;
System.out.println("Area is " + figref.area)();
figref = t;
System.out.println("Area is " + figref.area)();
}
 }
```

همان طوري كه توضيح درون متد main() نشان مي دهد، ديگر امكان اعلان اشيائ از نوع Figure وجود ندارد چون اكنون بصورت انتزاعي است كه كليه زير كلاسهاي Figure بايد متد area() را لغو كنند. براي اثبات اين امر سعي كنيد كه يك زير كلاس ايجاد نمايد كه متد area() را لغو نمي كند.حتماً يك خطاي زمان كامپايل دريافت خواهيد كرد . اگر چه امكان ايجاد يك شي از نوع Figure وجود ندارد اما مي توانيد يم كتغير ارجاع از نوع Figure ايجاد نمايد. متغير figref به عنوان ارجاعي به Figure اعلان شده و بدان معني مي باشد كه با استفاده از آن مي توان به يك شي از هر كلاس مشتق شده از Figure ارجاع كرد.

مفهوم static

شرايطي وجود دارد كه مايليد يك عضو كلاس را طوري تعريف كنيد كه مسبقل از هر شيئ آن كلاس مورد استفاده قرار گيرد. بطور معمول يك عضو كلاس بايد فقط همراه يك شيئ از همان كلاس مورد دسترسي قرار گيرد .اما اين امكان وجود دارد كه عضوي را ايجاد كنيم كه توسط خودش استفاده شود، بدون اينكه به يك نمونه مشخص ارجاع كنيم. براي ايجاد چنين عضوي ، قبل از آن واژه كليدي static را قرار دهيد. زماني يك عضو به عنوان static اعلام شود مي توان قبل از ايجاد هر شئ از آن كلاس ، و بدون ارجاعي به هيچ يك از اشياء آن را مورد استفاده قرار دهيد. مي توان متدها را هم به عنوان static اعلان نمايد.
رايج ترين مثال براي يك عضو static همان متد main() است.به اين دليل اين كار انجام مي شود كه چون بايد قبل از وجود هر نوع شئ فراخواني شود . متغيرهاي نمونه اعلان شده به عنوان static ضرورتاً متغيرهاي سراسري هستند . هنگامي كه اشياء آن كلاس اعلان مي شوند هيچ كپي از متغير static ساخته نمي شود. به جاي آن كليه نمونه هاي آن كلاس همان متغير هاي staticرا به اشتراك مي گذارند. متدهاي اعلان شده به عنوان static داراي چندين محدوديت هستند. آنها فقط ساير متدهاي static را فراخواني مي كند و فقط به دادهاي static دسترسي دارند. آنها به هيچ وجه امكان ارجاع به this, super را ندارند. اگر لازم است محاسباتي انجام دهيد تا متغيرهاي static را مقدار دهي اوليه نمايد، مي توانيد يك بلوك static اعلان نمايد كه فقط يك بار آن هم زماني كه كلاس براي اولين بار بارگذاري مي شود، اجرا گردد. مثال بعدي كلاسي را نشان مي دهد كه يك متد static برخي متدهاي static و يك بلوك مقداري دهي اوليه static دارد :

```
// Demonstrate static variables/ methods/ and blocks.
class UseStatic {
static int a = 3;
static int b;
static void meth(int x ){
System.out.println("x = " + x);
```

```
System.out.println("a = " + a);
System.out.println("b = " + b);
}
static {
System.out.println("Static block initialized.");
b = a * 4;
}
public static void main(String args[] ){
meth(42);
}
}
```

به محض اين كه كلاس usestatic بارگذاري مي شود ، كليه دستورات static اجرا مي شوند. ابتدا مقدار a برابر با 3 مي شود، سپس بلوك static اجرا شده (يك پيام چاپ خواهد شد) و در نهاتي مقدار a*4 يا عدد 12 در b به عنوان مقدار اوليه نهاده مي شود. سپس متد main() فراخواني مي شودكه متد math() را فراخواني خواهد كرد و مقدار 42 را به x خواهد گذاشت.سه دستور pritnln() به دو متغير static يعني a ,b و همچنين به متغير محلي x ارجاع مي كنند. همان طور كه قبلآ گفتيم ارجاع به هر يك از متغير هاي نمونه داخل يك متد static غير مجاز مي باشد.
خروجي برنامه به شكل زير خواهد بود:

```
Static block initialized.
x = 42
a = 3
b = 12
```

متدها و متغير هاي اعلان شده با static را مي توان در خارج از كلاسي كه تعريف شده اند مستقل از هر نوع شيئي مورد استفاده قرار داد. براي انجام اين كار كافي است كه نام كلاس را با يك عملگر نقطه بعد از مشخص نماييد.
بطور مثال مي توانيد اگر بخواهيد يك متد static را از خارج از كلاس مربوطه فراخواني كنيد، بااستفاده از شكل عمومي زير اين كار را انجام دهيد:

```
classname.method()
```

در اينجا classname نام كلاسي است كه متد static در آن اعلان شده است. همان طوري كه مي توانيد ببينيد، اين شكل بندي مشابه همان است كه براي فراخواني متدهاي غير static از طريق متغيرهاي ارجاع شيئ انجام مي گرفت . يك متغير static را نيز مي توان با همان روش با استفاده از از عملگر نقطه اي روي نام كلاس مورد دسترسي قرار داد. اين روشي است كه كه جاوا به وسيله ان يك روايت كنترل شده از توابع سراسري و متغيرهاي سراسري را پياده سازي مي كند. در اين جا مثالي وجود دارد، در داخل متد main() متد classname() و متغير b كه static هستند در خارج از كلاسهاي خود مورد دسترسي قرار مي گيرند.

```
class StaticDemo {
static int a = 42;
static int b = 99;
static void callme (){
System.out.println("a = " + a);
}
}
```

```
class StaticByName {
public static void main(String args[] ){
StaticDemo.callme();
System.out.println("b + " + StaticDemo.b);
}
}
```

استفاده از اشياء به عنوان پارامتر

تا كنون فقط از انواع ساده به عنوان پارامترهاي متدها استفاده كرديم. اما هم صحيح و هم معمول است كه اشياء را نيز به متدها پاس كنيم.
مثال زير را در نظر بگيريد:

```
// Objects may be passed to methods.
class Test {
int a/ b;
Test(int , int j ){
a= i;
b = j;
}
// return true if o is equal to the invoking object
boolean equals(Test o ){
if(o.a == a && o.b == b )return true;
else return false;
}
 }
class PassOb {
public static void main(String args[] ){
Test ob1 = new Test(100, 22);
Test ob2 = new Test(100, 22);
Test ob3 = new Test-(1,- 1);
System.out.println("ob1 == ob2 :" + ob1.equals(ob2));
System.out.println("ob1 == ob3 :" + ob1.equals(ob3));
}
 }
```

برنامه اين خروجي را توليد خواهد كرد.

```
ob1 == ob2 :true
ob1 == ob3 :false
```

در مثال متد equals() در داخل Test دو شئ را از نظر كيفيت مقايسه نموده و نتيجه را برمي گرداند. يعني اين متد شئ فراخواننده را با شئ كه پاس شده مقايسه كرده و اگر محتواي آن دو يك باشد آنگاه true را بر مي گرداند. در غير اين صورت false را برمي گرداند. پارامتر o در متدequals() مشخص كننده Test به عنوان نوع آن مي باشد. اگرچه Test يك نوع كلاس ايجادشده توسط برنامه است، اما به عنوان انواع توكار جاوا و به همان روش مورد استفاده قرار گرفته است. يك از رايجترين كاربردهاي پارامترهاي شيئ مربوط به سازنده گان است. غالباً ممكن است كه بخواهيد يك شيئ جديد را بسازيد طوري كه اين شئ در ابتدا نظير يك شي موجود باشد. براي انجام اين كار بايد يك تابع سازنده تعريف نماييد كه شي از كلاس خود را به عنوان يك پارامتر انتخاب مي كند.

بطور مثال روايت بعدي از Box()به يك شيئ امكان داده تا اغاز گر ديگري باشد.

```
// Here/ Box allows one object to initialize another.
class Box {
double width;
double height;
double depth;
// construct clone of an object
Box(Box ob ){ // pass object to constructor
width = ob.width;
height = ob.height;
depth = ob.depth;
}
// constructor used when all dimensions specified
Box(double w, double h, double d ){
width = w;
height = h;
depth = d;
}
// constructor used when no dimensions specified
Box (){
width =- 1; // use- 1 to indicate
height =- 1; // an uninitialized
depth =- 1; // box
}
// constructor used when cube is created
Box(double len ){
width = height = depth
}
// compute and return volume
double volume (){
return width * height * depth;
} }
class OverloadCons2 {
public static void main(String args[] ){
// create boxes using the various constructors
Box mybox1 = new Box(10, 20, 15);
Box mybox2 = new Box();
Box mycube = new Box(7);
Box myclone = new Box(mybox1);
double vol;
// get volume of first box
vol = mybox1.volume();
System.out.println("Volume of mybox1 is " + vol);
// get volume of second box
vol = mybox2.volume();
System.out.println("Volume of mybox2 is " + vol);
// get volume of cube
```

```
vol = mycube.volume();
System.out.println("Volume of cube is " + vol);
// get volume of clone
vol = myclone.volume();
System.out.println("Volume of clone is " + vol);
}
 }
```

بعدها خواهيد ديد كه وقتي شروع به ايجاد كلاسهاي خود مي نماييد، معمولآ براي اينكه اجازه دهيم تا اشياء بصورتي موثر و آسان ساخته شوند، لازم است اشكال هاي سازنده را فراهم آوريم.

استفاده از instanceof

در زبانهاي برنامه نويسي امروزه اين امكان وجود دارد كه بتوان بعد كامپايل و در زمان اجرا نوي شئ را تشخص داد. به طور مثال يك نخ(thread) كه داخل آن اشياء مختلف توليد كرده و يك نخ كه اينها را پردازش مي كند. در شرايطي براي پردازنده نخ مفيد مي باشد كه نوع هريك از اشيائي را كه قصد پردازش دارد بداند. دليل ديگري كه دانستن نوع شي را در زمان اجرا مهم مي كند تبديل نوع يا casting مي باشد. در جاوا يك تبديل غيرمجاز سبب بروز خطاي زمان اجرا مي شود و اكثر تبديلات غير مجاز در زمان كامپايل قابل رفع هستند. اما casting شامل سلسه مراتب كلاس مي باشد كه مي تواند تبديلات غير مجاز توليد كند و فقط در زمان اجرا قابل كشف مي باشند. براي مثال يك ابر كلاس A مي تواند به زير كلاس B تبديل كنيم و يك شي از نوع C را به نوع A تبديل كنيم. اما مجاز نيستيم يك شي B را به نوع C يا بالعكس تيديل كنيم. از آنجاي كه يك شي از نوع A مي تواند به اشيائي از نوع B يا C ارجاع كند در زمان اجرا براي فهميدن شي ارجاع شده مي توان قبل از تبديل تستي را انجام داد. به كمك عملگر زمان اجراي جاوا instanceof مي توان اين كنترل را انجام داد.
فرم عمومي instanceof به شكل زير مي باشد.

object instanceof type

يك نمونه از كلاس به كمك object و نوع كلاس با type مشخص مي شود. اگر object از نوع مشخص شده باشد آنگاه عملگر instanceof مقدار true بر مي گرداند. در غير اين صورت false بر مي گرداند. به اين ترتيب instanceof كمكي است براي اينكه برنامه شما اطلاعات نوع در باره يك شي را در زمان اجرا بدست آورد.
برنامه زير نشان دهنده استفاده از instanceof است.

```
// Demonstrate instanceof operator.
class A {
int i/ j;
}
class B {
int i/ j;
}
class C extends A {
int k;
}
class D extends A {
int k;
}
```

```
class InstanceOf {
public static void main(String args[] ){
A a = new A();
B b = new B();
C c = new C();
D d = new D();
if(a instancof A)
System.out.println("a is instance of A");
if(b instancof B)
System.out.println("b is instance of B");
if(c instancof C)
System.out.println("c is instance of C");
if(c instancof A)
System.out.println("c is instance of A");
if(a instancof C)
System.out.println("a is instance of C");
System.out.println();
// compare types of derived types
A ob;
ob = d; // A reference to d
System.out.println("ob new refers to d");
if(ob instancof D)
System.out.println("ob is instance of D");
System.out.println();
ob = c; // A reference to c
System.out.println("ob new refers to c");
if(ob instancof D)
System.out.println("ob is instance of D");
else
System.out.println("ob is instance of D");
if(ob instancof A)
System.out.println("ob is instance of A");
System.out.println();
// all object can be cast to Object
if(a instancof object)
System.out.println("a may be cast to Object");
if(b instancof object)
System.out.println("b may be cast to Object");
if(c instancof object)
System.out.println("c may be cast to Object");
if(d instancof object)
System.out.println("d may be cast to Object");
}
 }
```

خروجي حاصل از اجراي اين برنامه به شكل زير مي باشد:

```
a is instance of A
b is instance of B
```

```
c is instance of C
c can be cast to A

ob now refers to d
ob is instance of D

ob now refers to c
ob cannot be cast to D
ob can be cast to A

a may be cast to Object
b may be cast to Object
c may be cast to Object
d may be cast to Object
```

کلاس Singletone

در برخي مواقع با كلاسهاي برخورد مي كنيم كه لزوماً بايد يك و فقط يك متغير از آنها تعريف شود. بطور مثال يك عامل يا شي كه به يك منبع قابل اشتراك دسترسي دارد اما هيچ چيزي نمي تواند شي را از تعريف متغير ديگري از آن باز دارد پس چه كاري مي توان انجام داد. الگوي تك برگ براي انجام اين كار است .

الگوي تك برگي با گرفتن وظيفه ايجاد و قطع دسترسي به متغير در خود شي طرح را محدود مي كند. چنين كاري تضمين مي كند كه تنها يك متغير ايجاد شود و دسترسي به آن منفرد باشد. در زير پياده سازي از الگوي تك برگ را مشاهده مي كنيد.

```
/*
a class refrence to the singleton instance
*/
public class Singleton {
private static Singleton instance;
protected Singleton(){}
public static Singleton getInstance(){
if (instance== null) {
instance= new Singleton();
}
return instance;
}
}
```

كلاس Singleton يك متغير static از نوع Singleton دارد كه دسترسي به آن را فقط به روال getInstance() محدود كرده.

متدها در جاوا

زير برنامه ها درجاوا

زير برنامه عضو يك كلاس را با كلمه متد در جاوا شناساي مي كنند و فرم كلي تعريف يك متد به شكل زير مي باشد.

modifiers return-type subroutine-name (parameter-list) {
statements
}

در قسمت statement دستوراتي را كه در داخل متد مورد نظر شما انجام مي شود مي توانيد با توجه به پارامتر هاي ورودي دستورات را وارد كنيد.
در قسمت modifiers كه در شروع تعريف يك متد مي نويسيم مشخصات حقيقي يك متد را تعين ميكند. همان طور كه گفته شد كلمه public به اين معني است كه اين متد خارج از كلاس توسط شي تعريف شده از نوع كلاس مي تواند مورد دسترسي قرار بگيرد.
متد يك تابع مي باشد كه كار خاصي را انجام مي دهد مثل كارهاي محاسباتي پس بايد يك خروجي داشته باشد. نوع خوجي تابع را با return-type كه يكي از انواع داده هاي جاوا مي باشد مشخص مي كنيم.اگر متد شما خروجي نداشته باشد بايد مقدار void را در انجا بنويسيد.

در قسمت parameter-list از متد پارامتر هاي ورودي به متد مورد نظر را مشخص مي كنيم. در نوشتن پارامتر ورودي ابتدا نوع پارامتر ورودي را به فرم type parameter-type تعين كرده و اگر چند پارامتر به عنوان ورودي داشته باشيم آنها را با علامت كاما از هم جدا مي كنيم.
بطور مثال "double x,double y" و دقت كنيد كه نوشتن به فرم "double x,y" اشتباه مي باشد.
در زير چند مثال از متدها را مشاهده مي كنيد.

```
public static void playGame() {
// "public" and "static" are modifiers; "void" is the
// return-type; "playGame" is the subroutine-name;
// the parameter-list is empty
. . . // statements that define what playGame does go here
}

int getNextN(int N) {
// there are no modifiers; "int" in the return-type
// "getNextN" is the subroutine-name; the parameter-list
// includes one parameter whose name is "N" and whose
// type is "int"
. . . // statements that define what getNextN does go here
}

static boolean lessThan(double x, double y) {
// "static" is a modifier; "boolean" is the
// return-type; "lessThan" is the subroutine-name; the
// parameter-list includes two parameters whose names are
// "x" and "y", and the type of each of these parameters
// is "double"
. . . // statements that define what lessThan does go here
}
```

افزودن يك متد به كلاس Box

در اغلب اوقات از متدها براي دسترسي به مقدار متغيرهاي نمونه استفاده مي كنيم. همچنين مي توانيد متدهاي تعريف كنيد كه به صورت داخلي و توسط خود كلاس مورد استفاده قرار گيرند.در مثال Box محاسبه فضاي اشغالي (volume) چيزي است كه توسط كلاس Box در مقايسه با كلاس BoxDemo راحت تر مديريت مي شود.همچنين چون فضاي اشغالي يك box به اندازه آن دارد به نظر مي رسد كه بهتر است كه كلاس Box اين محاسبه را انجام دهد .
در مثال زير يك متد به Box اضافه نموده ايم.

```
// This program includes a method inside the box class.
class Box {
double width;
double height;
double depth;
// display volume of a box
void volume (){
System.out.print("Volume is ");
System.out.println(width * height * depth);
} }
class BoxDemo3 {
public static void main(String args[] ){
Box mybox1 = new Box ();
Box mybox2 = new Box ();
// assign values to mybox1's instance variables
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
/* assign different values to mybox2's
instance variables */
mybox2.width = 3;
mybox2.height = 6;
mybox2.depth = 9;
// display volume of first box
mybox1.volume ();
// display volume of second box
mybox2.volume ();
}
}
```

اين برنامه خروجي زير را توليد مي كند:

```
Volume is 3000
Volume is 162
```

از يك نقطه نظر عمومي يك متد شيوه جاوا براي پياده سازي زير روالها (subroutines) است. زماني كه يك متد از متغير نمونه اي كه توسط كلاس خود تعريف شده استفاده مي كند، اين كار را بطور مستقيم و بدون ارجاع صريح به يك شي و بدون عملگر نقطه اي انجام مي دهد. يك متد همواره نسبت به شيئ از كلاس خود

قرار داده مي شود.زماني كه اين تعبيه اتفاق مي افتد اين شي شناخته شده است. اين به آن معني است كه متغيرهاي depth , height و widthدر داخل متد volume() به طور صريح به كپي هاي در آن شي كه اين متد را فعال مي كند ارجاع مي كنند.

فرض كنيد كه بخش ديگري از برنامه شما قصد فهميدن فضاي اشغالي يك box را دارد و راه بهتر پياده سازي volume() اين است كه آن را وادار كنيم كه فضاي اشغالي box را محاسبه كرده و نتيجه را به فراخواننده بر گرداند. مثال بعدي روايت پيشرفته تر برنامه قبلي است كه اين همين كار را انجام مي دهد.

```
// Now/ volume ()returns the volume of a box.
class Box {
double width;
double height;
double depth;
// compute and return volume
doble volume (){
return width * height * depth;
} }
class BoxDemo4 {
public static void main(String args[] ){
Box mybox1 = new Box ();
Box mybox2 = new Box ();
double vol;
// assign values to mybox1's instance variables
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
/* assign different values to mybox2's
instance variables */
mybox2.width = 3;
mybox2.height = 6;
mybox2.depth = 9;
// get volume of first box
vol = mybox1.volume ();
System.out.println("Volume is " + vol);
// get volume of second box
vol = mybox2.volume ();
System.out.println("Volume is " + vol);
}
}
```

ارسال پارامتر در مثال قبلي ابعاد هر box بايد به طور جداگانه با استفاده از يك سلسله دستورات نظير زير تعين گردد :

```
mybox1.width = 10;
mybox1.height = 20;
mybox1.depth = 15;
```

اين كد اگرچه كار مي كند اما دو اشكال دارد. اول اين كه آن بد تركيب و مستعد خطا مي باشد و دوم اين كه در طراحي خوب برنامه هاي جاوا بايد متغيرهاي نمونه فقط از

طريق متدهاي تعريف شده توسط كلاس خودشان قابل دسترس باشند. در آينده قادر خواهيد بود تا رفتار يك متد را تغير دهيد اما نمي توانيد رفتار يك متغير نمونه بي حفاظ (افشا شده) را تغير دهيد. پس راه بهتر اين است كه ابعاد Box را به يك متد كه وظيفه مقدار دهي را دارد ارسال كنيم.
در برنامه زير اين كار نشان داده شده:

```
// This program uses a parameterized method.
class Box {
double width;
double height;
double depth;
// compute and return volume
double volume (){
return width * height * depth;
}
// sets dimensions of box
void setDim(double w, double h, double d ){
width = w;
height = h;
depth = d;
} }
class BoxDemo5 {
public static void main(String args[] ){
Box mybox1 = new Box ();
Box mybox2 = new Box ();
double vol;
// initialize each box
mybox.setDim(10, 20, 15);
mybox.setDim(3, 6, 9);
// get volume of first box
vol = mybox1.volume ();
System.out.println("Volume is " + vol);
// get volume of second box
vol = mybox2.volume ();
System.out.println("Volume is " + vol);
}
}
```

ارسال ارگامان با مقدار و مرجع

در فراخواني با مقدار (call by value) مقدار يك آرگومان را در پارامتر رسمي متد كپي مي كند. بنابراين تغيرات روي پارامتر در متد اعمال مي شود تاثيري در روي آرگوماني كه روي آن فراخواني شده ندارد. در روش فراخواني با مرجع (call by refernce) ارجاع به يك آرگومان (نه مقدار آن آرگومان) به پارامتر پاس (گذر) داده مي شود. داخل متد از اين ارجاع براي دسترسي به آرگومان واقعي مشخص شده در فراخواني استفاده مي شود و هر گونه تغير در روي پارامتر روي ارگمان تاثير خواهد داشت.
در جاوا وقتي كه يك نوع ساده را به يك متد پاس مي كنيم اين نوع به وسيله مقدارش پاس مي شود. به طور مثال برنامه زير را در نظر بگيريد:

```
// Simple types are passed by value.
class Test {
void meth(int i, int j ){
i *= 2;
j /= 2;
} }
class CallByValue {
public static void main(String args[] ){
Test ob = new Test ();
int a = 15, b = 20;
System.out.println("a and b before call :" + a + " " + b);
ob.meth(a, b);
System.out.println("a and b after call :" ++ a + " " + b);
}
 }
```

خروجي به فرم زير خواهد بود:

```
a and b before call :15 20
a and b after call :15 20
```

به خوبي مشاهده مي كنيد كه عمليات اتفاق افتاده داخل meth() هيچ تاثيري در روي مقادير a و b كه در فراخواني استفاه شده است، نخواهد داشت. در اين جا مقادير آنها به 10 و 30 تغير نمي يابد. بياد بياوريد كه وقتي يك متغير از يك نوع كلاس ايجاد مي كنيد، شما فقط يك ارجاع به شي خلق مي كنيد. بدين ترتيب، وقتي اين ارجاع را به يك متد ارسال مي كنيد پارامتري كه آن را در يافت مي كند به همان شي ارجاع مي كند كه توسط آرگومان به آن ارجاع داده شده بود.اين به اين معني است كه اشياء با استفاده از روش فراخواني مرجع تغيرات روي اشياۀ در داخل متد سبب تغير روي شيئ است كه به عنوان آرگمان استفاده شده است.
مثال زير را در نظر بگيريد:

```
// Objects are passed by reference.
class Test {
int a, b;
Test(int i, int j ){
a = i;
b = j;
}
// pass an object
void meth(Test o ){
o.a *= 2;
o.b /= 2;
} }
class CallByRef {
public static void main(String args[] ){
Test ob = new Test(15, 20);
System.out.println("ob.a and ob.b before call :" + ob.a + " " + ob.b);
ob.meth(ob);
```

```
System.out.println("ob.a and ob.b after call :" ++ ob.a + " " + ob.b);
}
}
```

برنامه فوق خروجي زير را توليد مي كند:

```
ob.a and ob.b before call :15 20
ob.a and ob.b after call :30 10
```

همان طور كه مي بينيد اعمال داخل متد()meth شيئ را كه به عنوان يك آرگومان استفاده شده تحت تاثير قرار داده است.

برگرداندن اشياء

يك متد قادر به اين است هرنوع داده شامل انواع كلاسي كه ايجاد مي كند را برگرداند. به طور مثال در برنامه بعدي متد()incrByTen يك شيئ را بر مي گرداند كه در آن مقدار a ده واحد بزرگتر از مقدار آن در شيئ كه آن را فراخواننده است.

```
// Returning an object.
class Test {
int a;
Test(int i ){
a = i;
}
Test incrByTen (){
Test temp = new Test(a 10);
return temp;
} }
class RetOb {
public static void main(String args[] ){
Test ob1 = new Test(2);
Test ob2;
ob2 = ob1.incrByTen ();
System.out.println("ob1.a :" + ob1.a);
System.out.println("ob2.a :" + ob2.a);
ob2 = ob2.incrByTen ();
System.out.println("ob2.a after second increase :" + ob2.a);
}
}
```

خروجي به شكل زير خواهد بود:

```
ob1.a :2
ob2.a :12
ob2.a after second increase :22
```

هر بار كه متد ()incrByTen فراخوانده مي شود يك شيئ جديد توليد شده و يك ارجاع به آن شيئ جديد به روال فراخواننده برگردان مي شود.از آنجاي كه كليه اشياء به صورت پويا و با استفاده از new تخصيص مي يابند، نگراني راجبع شيئ كه خارج از قلمرو برورد نخواهيد داشت، زيرا در اين متدي كه شيئ در آن ايجاد شده پايان خواهد

گرفت. يك شي مادامي كه از جايي در برنامه شما ارجاعي به آن جود داشته باشد، زنده خواهد ماند.

انباشت متدها

در شي گراي اين امكان وجود دارد كه دو يا چند متد را در داخل يك كلاس را كه همنام مي باشند تعريف كرد. در جاوا به اين امكان انباشتن متدها مي گويند. انباشتن روشها يكي از جنبه هاي هيجان انگيز و سودمند جاوا است و چند شكلي را پياده مي كند.
زماني كه يك متد انباشته شده فراخواني گردد، جاوا از نوع يا شماره آرگومانها بعنوان راهنماي تعين روايت (version) متد انباشته شده اي كه واقعآ فراخواني مي شود، استفاه مي كند.
به اين ترتيب متدهاي انباشته شده بايد در نوع يا شماره پارامترهايشان متفاوت باشند. زماني كه جاوا با يك فراخواني به يك متد انباشته شده مواجه مي شود، خيلي ساده روايتي از روش را اجرا مي كند كه پارامترهاي آن با آرگمانهاي استفاده شده در فراخواني مطابقت داشته باشند.
مثال زير انباشتن متدها را نشان مي دهد.

```
// Demonstrate method overloading.
class OverloadDemo {
void test (){
System.out.println("No parameters");
}
// Overload test for one integer parameter.
void test(int a ){
System.out.println("a :" + a);
}
// Overload test for two integer parameters.
void test(int a, int b ){
System.out.println("a and b :" + a + " " + b);
}
// Overload test for a double parameter.
double test(double a ){
System.out.println("double a :" + a);
return a*a;
} }
class Overload {
public static void main(String args[] ){
OverloadDemo ob = new OverloadDemo ();
double result;
// call all versions of test ()
ob.test ();
ob.test(10);
ob.test(10, 20);
result = ob.test(123.2);
System.out.println("Result of ob.test(123.2 :)" + result);
}
 }
```

خروجي برنامه يه شكل زير خواهد شد:

```
No parameters
a :10
a and b :10 20
double a :123.2
Result of ob.test(123.2 :)15178.2
```

همان طور كه مشاهده مي كنيد متد Test() به چهار شكل مختلف نوشته شده است. اولين روايت هيچ پارامتري را نمي گيرد . در دومين روايت يك پارامتر ورودي با مقدار صحيح و در سومين دو پارامتر با نوع صحيح داريم. در آخرين روايت يك پارامتر از نوع double مي گيرد. برگرداندن مقداري در آخرين روايت هرگز نتيجه حاصل از عمل انباشت متدها نيست چون انواع برگشتي نقشي در تجزيه و تحليل انباشت ندارد.
زماني كه يك متد انباشته شده فراخواني مي شود، جاوا به دنبال تطبيقي بين آرگمانهاي استفاده شده براي فراخواني متد و پارامترها آن متد مي گردد. اما، اين تطابق نبايد لزوماً هميشه صحيح باشد. در برخي شرايط تبديل انواع خودكار جاوا مي تواند نقشي در تجزيه و تحليل انباشت داشته باشد.
براي مثال برنامه بعدي را در نظر بگيريد.

```
// Automatic type conversions apply to overloading.
class OverloadDemo {
void test (){
System.out.println("No parameters");
}
// Overload test for two integer parameters.
void test(int a, int b ){
System.out.println("a and b :" + a + " " + b);
}
// Overload test for a double parameter.
double test(double a ){
System.out.println("Inside test(double )a :" + a);
} }
class Overload {
public static void main(String args[] ){
OverloadDemo ob = new OverloadDemo ();
int i = 88;
ob.test ();
ob.test(10, 20);
ob.test(i); // this will invoke test(double)
ob.test(123.2); // this will invoke test(double)
}
 }
```

اين برنامه خروجي زير را ايجاد مي كند.

```
a and b :10 20
Inside test(double )a :88
Inside test(double )a :123.2
```

در كلاس overloadDemo داخل آن متد test(int) موجود نمي باشد. بنابراين هنگامي كه متد test() همراه با يك عدد صحيح داخل overload فراخواني مي شود، هيچ

تطبيق دهنده اي پيدا نخواهد شد. اما جاوا مي تواند به طور خودكار يك عدد صحيح را به يك double تبديل كند و اين تبديل براي رفع فراخواني مورد استفاده قرار مي گيرد. بنا بر اين بعد از آنكه test(int) پيدا نشد جاوا i را ارتقاء داده و آنگاه test(double) فراخواني مي شود. جاوا فقط در صورتي كه هيچ تطبيق دقيقي پيدا نكند از تبديل خودكار استفاده مي كند.

لغو(جلوگيري) متد

در يك سلسه مراتب از كلاس و زماني كه يك متد در يك زيركلاس همان نام و نوع يك متد در ابر كلاس خود داشته باشد، آنگاه بيان مي شود كه ان متد در زير كلاس ، متد موجود در ابر كلاس را لغو نموده(يا از پيشروي آمتد جلو گيري مي نمايد). وقتي كه يك متد لغو شده از داخل يك زير كلاس فراخواني مي شود، همواره به روايتي از آن متد كه توسط زير كلاس تعريف شده، ارجاع خواهد نمود و روايتي كه ابر كلاس از همان متد تعريف نموده، پنهان خواهد شد.

مورد زير را در نظر بگيريد:

```
// Method overriding.
class A {
int i, j;
A(int a, int b ){
i = a;
j = b;
}
// display i and j
void show (){
System.out.println("i and j :" + i + " " + j);
} }
class B extends A {
int k;
B(int a, int b, int c ){
super(a, b);
k = c;
}
// display k -- this overrides show ()in A
void show (){
System.out.println("k :" + k);
}
}
class Override {
public static void main(String args[] ){
B subOb = new B(1, 2, 3);
subOb.show (); // this calls show ()in B
}
}
```

حاصل ايجاد شده توسط برنامه به شكل زير مي باشد:

```
k:3
```

وقتي show() روي يك شي از نوع B فراخواني مي شود، روايتي از show كه داخل B تعريف شده مورد استفاده قرار مي گيرد، يعني روايت اعلان شده در A را لغو مي كند. اگر مي خواهيد به روايت ابر كلاس يك متد لغو شده دسترسي داشته باشيد، اين كار را با استفاده از super انجام دهيد. براي مثال در اين روايت از B روايت ابر كلاس

show() داخل روايت مربوط به زير كلاس فراخواني خواهد شد. اين امر به كليه متغير هاي نمونه اجازه مي دهد تا به نمايش درآيند.

```
class B extends A {
int k;
B(int a, int b, int c ){
super(a, b);
k = c;
}
void show (){
super.show (); // this calls A's show ()
System.out.println("k :" + k);
}
}
```

اگر اين روايت از A را در برنامه قبلي جايگزين نماييد، خروجي زير را مشاهده مي كنيد :

```
i and j :1 2
k:3
```

در اينجا super.show() روايت كلاس بالاي show() را فراخواني مي كند. لغو متد زماني اتفاق مي افتد كه اسامي و نوع دو متد يكسان باشند. اگر چنين نباشد، آنگاه دو متد خيلي ساده انباشته خواهند شد. به عنوان مثال ، اين روايت اصلاح شده مثال قبلي را در نظر بگيريد :

```
// Methods with differing type signatures are overloaded -- not
// overridden.
class A {
int i, j;
A(int a, int b ){
i = a;
j = b;
}
// display i and j
void show (){
System.out.println("i and j :" + i + " " + j);
} }
// Create a subclass by extending class A.
class B extends A {
int k;
B(int a, int b, int c ){
super(a, b);
k = c;
}
// overload show ()
void show(String msg ){
System.out.println(msg + k);
} }
class Override {
```

```
public static void main(String args[] ){
B subOb = new B(1, 2, 3);
subOb.show("This is k :"); // this calls show ()in B
subOb.show (); // this calls show ()in A
}
}
```

حاصل ايجاد شده توسط برنامه به شكل زير مي باشد:

```
This is k:3
i and j :1 2
```

روايت show() در B يك پارامتر رشته اي مي گيرد. اين عمل سبب متفاوت شدن تايپد به نوع آن از نوع موجود در A شده، كه هيچ پارامتري را نمي گيرد. بنابراين انباشت شدن (يا مخفي شدن اسم) اتفاق نمي افتد.

متدهاي Native

در برخي از مواقع گاهي اتفاق مي افتد كه مي خواهيد از يك زير روال (subroutine) نوشته شده توسط ساير زبانهاي غير از جاوا استفاده كنيد. معمولآ به دلايل در دسترس بودن روالهاي كه لازم داريم به زبانهاي ديگر يا كار با مورد خاص و سيستمي بودن قسمتي از برنامه از زير روالهاي نوشته شده با ساير زبانها استفاده مي كنيم. چنين زير روالهاي به صورت كدهاي قابل اجرا براي CPU و محيط خاصي خاصي كه در آن كار مي كنيد عرضه مي شوند يعني به صورت كدهاي بومي (Native). به عنوان مثال ممكن است كه بخواهيد يك روال كد بومي را فراخواني كنيد تا به زمان اجراي سريعتر برسيد. و يا بخواهي از يك كتابخانه تخصصي شده مثل يك بسته آماري استفاده كنيد.
اما از آنجاي كه برنامه هاي جاوا به كد بايتي كامپايل مي شوند و سپس توسط توسط سيستم در حين اجراي جاوا تفسير خواهند شد، بنا براين به نظر مي رسد فراخواني يك زير روال كد بومي از داخل برنامه جاوا غير ممكن باشد. خوشبختانه، جاوا واژه كليدي native را فراهم ديده تا از آن براي اعلان متدهاي كدهاي بومي استفاه شود. هر بار كه اين متدها اعلان مي شوند مي توانيد آنها را از درون برنامه خود فراخواني كنيد، درست همانند فراخواني هر متد ديگري در جاوا. براي اعلان يك متد بومي اصلاحگر native را قبل از متد قرار دهيد. اما براي آن بدنه اي تعريف نكنيد.
بطور مثال:

```
public native int meth ();
```

هر بار كه يك متد بومي را اعلان مي نمايد، مجبوريد متد بومي نوشته و يكسريي مراحل پيچيده تر را تعقيب كنيد تا آن را به كدهاي جاواي خود پيوند دهيد. به اين نكته توجه داشته باشيد كه مراحل دقيقي كه لازم است طي نمايد ممكن است براي محيطها و روايتهاي گوناگون جاوا متفاوت باشند. همچنين زباني كه مورد استفاده براي پياده سازي روال بومي موثر مي باشد. اين بحث از JDK (version 1.02) و ابزار هاي آن استفاده مي كند و محيط windows NT/95 را فرض مي كند. زباني هم كه براي پياده سازي روال بومي استفاده شده زبان C مي باشد. آسان ترين روش براي درك اين عمل انجام يك كار عملي است.
براي شروع برنامه كوتاه زير را كه از يك متد native به نام test() استفاده مي كند وارد نمايد :

```
// A simple example that uses a native method.
```

```
public class NativeDemo {
int i;
int j;
public static void main(String args[] ){
NativeDemo ob = new NativeDemo ();
ob.i = 10;
ob.j = ob.test (); // call a native method
System.out.println("This is ob.j :" + ob.j);
}
// declare native method
public native int test ();
// load DLL that contains static method
static {
System.loadLibrary("NativeDemo");
}
}
```

اگر به بلوك static دقت كنيد، همانطوري كه قبلاً گفتيم يك بلوك static فقط يك بار اجرا مي شود و آن هم زماني است كه برنامه شما شروع به اجرا مي نمايد (دقيقتر زماني كه آن كلاس براي اولين بار بار گذاري مي شود). در اين حالت ، از بلوك static استفاده شده تا كتابخانه پيوند پويا در زمان اجرا (dynamic link liberary) را كه در برگيرنده پياده سازي روال بومي ()test است بار گذاري نمايد. كتابخانه فوق توسط متد ()LoadLibrary بارگذاري مي شود كه بخشي از كلاس System است و شكل عمومي آن به فرم زير است :

```
Static void LoadLibrary( string filename)
```

پارامتر filename رشته اي است كه نام فايل كتابخانه اي را نگهداري مي كند. براي محيط ويندوز فرض شده كه پسوند dll را داشته باشد. بعد از وارد كردن برنامه آنرا كامپايل كنيد تا NativeDemo.class توليد شود. سپس به كمك java.exe براي ايجاد دو فايل كمك بگيريد. اين دو فايل NativeDemo.1 و NativeDemo.C مي باشند. شما NativeDemo.h را در پياده سازي روال ()test مي گنجانيد. فايل NativeDemo.C يك stub (رابط بين برنامه جاوا و روال بومي) را درخود دارد كه كدهاي ارتباطي در داخل آن موجود مي باشند. براي توليد NativeDemo.h از دستور بعدي javah NativeDemo استفاده كنيد كه اين دستور يك فايل سرآيند به نام NativeDemo.h را ايجاد مي كند. اين فايل همان فايلي است كه بايد در فايل برنامه به زبان C براي روال ()test گنجانده شود.

حاصل توليد شده اين فرمان به شكل زير است :

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include
/* Header for class NativeDemo */
#ifndef_ Included_NativeDemo
#define_ Included_NativeDemo
typedef struct ClassNativeDemo {
long j;
long j;
} ClassNativeDemo;
HandleTo(NativeDemo);
#ifdef __ cplusplus
```

```
extern "C" {
#endif
extern long NativeDemo_test(struct HNativeDemo *);
#ifdef __ cplusplus
}
#endif
#endif
```

چند چيز در اين فايل است كه بايد به آنها دقت نمايد. اول اينكه ساختار ClassNativeDemo دو عضو را در بر مي گيرد : i و j . اين متغير هاي نمونه تعريف شده توسط NativeDemo در فايل جاوا را معين مي كند. دوم اين كه ماكرو HandleTo() نوع ساختار HNativeDemo را ايجاد مي كند، كه براي ارجاع به شيئ كه test() را فرخواني مي كند مورد استفاده قرار مي گيرد. به خط بعدي دقت كنيد كه الگوي براي متد test() ايجاد كرده ايد تعريف مي كند:

```
extern long NativeDemo_test(struct HNativeDemo *);
```

دقت كنيد كه نام تابع NativeDemo-test است .بايد اين را به عنوان نام متد بومي كه پياده سازي مي كنيد بكار بريد. يعني به جاي ايجاد يك روال C موسوم به test() يك تابع موسوم به NativeDemo-test() را ايجاد مي كنيد.پيشوند NativeDemo براي اين اضافه شده تا مشخص كننده متد test() كه بخشي از كلاس NativeDemo مي باشد، باشد. پيشوند گذاري نام متد بومي با نام كلاس فراهم كننده شيوه اي است براي تمايز بين روايتهاي مختلف. به عنوان يك قانون عمومي ، توابع بومي ، يك نام كلاس كه در آن اعلان شده اند را به عنوان يك پيشوند قبول مي كنند.مقدار برگشتي تابع NativeDemo-test() از نوع long است، اگر چه داخل برنامه جاوا كه آن را فراخواني مي كند به عنوان يك int توصيف شده است.
دليل آن اين است كه در جاوا اعداد صحيح مقادير 32 بيتي هستند و در زبان C اندازه مقدار long همان 32 بيت مي باشد.الگوي NativeDemo-test() يك پارامتر از نوع HNativeDemo* را تعريف مي كند. كليه متدهاي بومي حداقل يك پارامتر دارند كه اشاره گري است به شيئ كه متد بومي را فراخواني نموده است . اين پارامتر ضرورتآ شبيه به this است. مي توانيد از اين پارامتر استفاده كرده تا دسترسي به متغيرهاي نمونه شيئ كه متد را فراخواني مي كند، داشته باشد.
بمنظور توليد فايل stub براي NativeDemo.C از اين دستور استفاده كنيد:

```
java-stubs NativeDemo
```

فابل توليد شده به شكل زير است:

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include
/* Stubs for class NativeDemo */
/* SYMBOL :"NativeDemo/test ()I"/ Java_Native_test_stub */
__ declspec(dllexport )stack_item
*Java_NativeDemo_test_stub(stack_item *_P_/struct execenv *_ EE )_{
extern long NativeDemo_test(vide *);
_ P_[0].i = NativeDemo_test_(P_[0].p);
return_ P _+ 1;
}
```

شما باید این فایل را کامپایل نموده و با فایل پیاده سازي خود پیوند دهید. پس از تولید فایلهاي سرآیند و stub ضروري، مي توانید پیاده سازي خود را از ()test بنویسید. از روایتي که در زیر نشان داده ایم، استفاده کنید :

```
/* This file contains the C version of the
test ()method.
*/
#include " stubpreamble.h"
#include "NativeDemo.h"
long NativeDemo_test(struct HNativeDemo *this)
{
printf("This is inside the native method.\n");
printf("this->i :%ld\n"/ unhand(this.-)i);
return( unhand(this-)>i);
}
```

دقت کنید که این فایل در بر گیرنده stubpreamble.h است که شامل اطلاعات رابط سازي (interfacing) است. این فایل توسط کامپایلر جاوا براي شما فراهم مي شود. فایل سرآیند NativeDemo قبلاً توسط java ایجاد شده بود. پس از ایجاد test.c باید آن را و NativeDemo.C را کامپایل کنید. بعد این دو فایل را به عنوان يك dll (کتابخانه پویا) با یکدیگر پیوند دهید. به منظور انجام اینکار با کامپایلر میکروسافت C++/C، از خط فرمان دستور CL/LD NativeDemo.ctest.c استفاده کنید. این فرمان يك فایل تحت عنوان NativeDemo.dll تولید مي کند. هر بار که این کار انجام شود، میتوانیدبرنامه جاوا را اجرا نماید.
انجام این کار خروجي زیر را تولید مي کند:

```
This is inside the native method.
this->i :10
This is ob.j :20
```

در داخل test.c به استفاده از متد ()unhand دقت نماید. این ماکرو توسط جاوا تعریف شده و يك اشاره گر را به نمونه ساختار ClassNativeDemo که در NativeDemo.h تعریف شده و همراهي شيئ است که ()test را فراخواني مي کند، بر مي گرداند. در کل هر گاه نیاز به دسترسي به اعضاء يك کلاس جاوا داشته باید از ()unhand استفاده مي کنید.
یادآوري مي کنیم که استفاه از native بستگي به پیاده سازي ومحیط دارند و باید مستندات موجود در سیستم توسعه جاوا خود را مطالعه نمایید تا جزئیات مربوط به روشهاي بومي را درك کنید.
اگر چه استفاده از متدهاي بومي مزیتهاي دسترسي به روالهاي کتابخانه اي و اجراي سریعتر بعضي از کار ها را فراهم مي کند اما دو مشکل اساسي را در خود نشان مي دهد، کاهش امنیت و کاهش قابلیت حمل.
يك متد بومي ریسك امنیتي را مطرح مي کند. از آنجاي که يك متد بومي کد واقعي ماشین را اجرا مي کند، مي تواند به هر بخش از رایانه میزبان دسترسي داشته باشد . یعني که کد بومي منحصر به محیط اجراي جاوا نیست. این کد امکان حمله ویروسي را هم مي دهد. به همین دلیل ریز برنامه ها نمي توانند از روشهاي بومي استفاده کنند. همچنین بارگذاري dll ها مي تواند محدود شود و بار گذاري آنها منوط به تصدیق مدیر امنیتي باشد.

دومين مشكل قابليت حمل است چون كد بومي داخل يك dll گنجانده شده، بايد روي ماشيني كه در حال اجرايي برنامه جاوا است حاضر باشد. به علاوه چون هر متد بومي بستگي به CPU وسيستم عامل دارد ، هر dll بطور وراثتي غير قابل حمل مي شود. به اين ترتيب، يك برنامه جاوا كه از متدهاي بومي استفاده مي كند، فقط قادر است كه روي يك ماشين كه براي آن يك dll سازگار نصب شده باشد، اجرا شود.

فصل سوم

فصل سوم

JFC and Swing

در برنامه هاي جاوا به منظور ايجاد رابط گر افيكي كاربر (GUI) و اضافه كردن قابليتهاي گرافيكي بالا از Java Foundation Class (JFC) استفاده مي كنيم. كلاسهاي پايه جاوا (JFC) قابليتهاي مختلف براي ايجاد كردن برنامه ها به صورت ويژول و امكان كار كردن با موس فراهم مي كنند. JFC شامل كامپونتهاي Swing، توابع كار با اشياء دوبعدي (Java 2D API)، پشتيباني از كشيدن و انداختن (Drag & Drop) و ساير امكانات ديگر مي باشد.
اصلي ترين قابليت JFC كامپونتهاي Swing است و از اين كامپونتها براي ايجاد و به كارگيري عناصر ويژوال مثل دكمه ها،جعبه متنها، قابها، منوها و ساير گزينه استفاده مي شود. بطور كلي Swing داراي هفده بسته كلاسهاي قابل استفاده است كه آنها را در جدول زير نشان داده ايم و اكثر برنامه هاي ساده از دو بسته Javax.swing و javax.swing.event در كد خود استفاده مي كنند.

javax.accessibility	javax.swing.plaf	javax.swing.text.html
javax.swing	javax.swing.plaf.basic	javax.swing.text.parser
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.rtf
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.tree

javax.swing.event javax.swing.table javax.swing.undo
javax.swing.filechooser javax.swing.text

اولين برنامه Swing

در اين برنامه يك فريم ساده براي نشان دادن يك پيام توليد خواهيم كرد.

```
import javax.swing.*;

public class HelloWorldSwing {
    private static void createAndShowGUI() {
    //Make sure we have nice window decorations.
    JFrame.setDefaultLookAndFeelDecorated(true);

    //Create and set up the window.
    JFrame frame = new JFrame("HelloWorldSwing");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    //Add the ubiquitous "Hello World" label.
    JLabel label = new JLabel("Hello World");
    frame.getContentPane().add(label);

    //Display the window.
    frame.pack();
    frame.setVisible(true);
  }

  public static void main(String[] args) {
    //Schedule a job for the event-dispatching thread:
    //creating and showing this application's GUI.
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        createAndShowGUI();
      }
    });
  }
}
```

پس از اجراي اين برنامه قاب زير را مشاهده خواهيد كرد كه با كمك برچسب يك پيغام Hello World را نشان خواهد داد.

در كدهاي زير نحوه استفاده از كلاس فريم جاوا يا JFrame نوشته نشان داده شده:
با كمك اعلان new JFrame("HelloWorldSwing"); يك فريم با عنواني كه با پارامتر ورودي دريافت مي كند ايجاد كرده و به شي ايجاد شده از آن نوع انتساب مي دهد

```
JFrame.setDefaultLookAndFeelDecorated(true);
JFrame frame = new JFrame("HelloWorldSwing");
...
frame.pack();
frame.setVisible(true);
```

دستور اول نوشته شده براي مشخص كردن شكل و فرم پنجره ها به همراه لبه هاي آنها است. دو دستور آخر فريم را آماده و نشان خواهند داد. براي اضافه كردن برچسب از كلاس برچسب جاوا يعني JLabel كه عنوان برچسب را به هنگام ساخت از ورودي مي گيرد كمك مي گيريم. به منظور اضافه كردن برچسب به فريم ساخته شده از متد getContemtPane()از كه در آن نام عنصر را به داخل پاس مي كنيم. توجه داشته باشيد كه به طور مستقيم نمي توان به فريم عنصر اضافه كرد و از content pane براي اضافه كردن استفاده مي شود. دليل اين امر اين است كه كلاس فريم يك ظرف (با ظرفهاي EJB اشتباه گرفته نشود) شامل اجزائ گرافيكي به جز منو و تعريف پنجره ها است. دستور زير زماني كه دكمه بستن سمت راست در بالا كليك مي شود برنامه ما را از سيستم خارج مي كند.

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

به جاي دستور بالا در ورژنهاي قبلي از دستور زير استفاده مي شده:

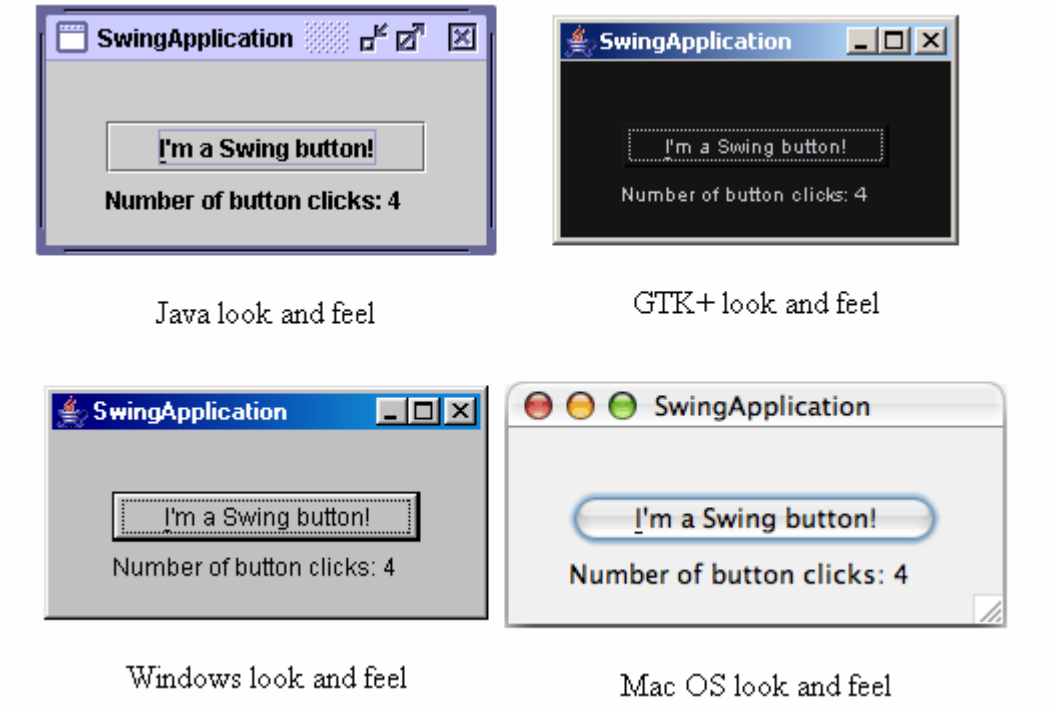
```
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});
```

استفاده از كد زير به جاي فراخواني مستقيم براي اين مي باشد كه اگر در نخ برنامه اشكالي به وجود بيايد به طور خودكار از برنامه خارج خواهد شد.

```
javax.swing.SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        /* create and show the GUI */
    }
});
```

Look and Feel

برنامه هاي ويژوال جاوا مي توانند در سيستمهاي مختلف با توجه به فرم اشكال در آن سيستم به همان روش نشان داده شوند. همچنين مي توانند با يكي از look&feel موجود داخلي جاوا مثل look&feel مخصوص كامپيوترهاي مكينتاش كار كنند. چند عدداز اين look&feel ها در زير نشان داده شده. ميتوان در جاوا look&feel خاص خود را ايجاد كند مثل look&feel مخصوص محصولات Oracle كه اگر محصولات آن را نصب كرده باشيد آنرا خواهيد ديد.



کدهاي زير براي مشخص کردن Look&Feel در برنامه به کار مي رود و در آن متغير رشته اي lookAndFeel براي مشخص کردن Look&Feel که به صورت پيش فرز در سيستم مي باشد است.

```
String lookAndFeel = null;
...
lookAndFeel =
UIManager.getCrossPlatformLookAndFeelClassName();
...
try {
    UIManager.setLookAndFeel(lookAndFeel);
} catch (Exception e) { }
...// Create and show the GUI...
```

شما مي توانيد مقدار دلخواه را در داخل اين متغير قرار دهيد مثل رشته زير که Look&Feel مخصوص ويندوز را نشان مي دهد.

"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"

در زمان اجرا به کمك کدهاي زير مي توانيد Look&Feel را تغير دهيد:

```
UIManager.setLookAndFeel(lnfName);
SwingUtilities.updateComponentTreeUI(frame);
frame.pack();
```

اضافه کردن دکمه

از کلاس JButton براي اضافه کردن دکمه ها و کنترل اعمال آنها استفاده مي کنيم.
نحوه تعريف دکمه به شکل زير است:

```
JButton button = new JButton("I'm a Swing button!");
button.setMnemonic('i');
button.addActionListener(/*...create an action listener...*/);
```

خط اول يك دكمه با متن كه روي آن نوشته مي شود ايجاد مي كند. در خط دوم يك ميانبر براي آن كليد به كمك متد setMnemonic() ايجاد كرده و با استفاده از Alt-i فشرده شدن دكمه را شبيه سازي مي كنيم.
كد هاي بعدي نشان دهنده اضافه كردن اجزاء گرافيكي به ظرف مخصوص پانل پيش از اضافه كردن آن به فريم است.

```
JPanel panel = new JPanel(new GridLayout(0,1));
panel.add(button);
panel.add(label);
panel.setBorder(BorderFactory.createEmptyBorder(...));
```

خط اول يك پانل با كلاس JPanel تعريف مي كند و اين شي يك مدير طرح بندي (layout manager) است كه قابليت تعين اندازه و وضعيت مكاني هر كدام از اجزا را دارد. كد new GridLayout(0,1) ظرف را وادار مي كند كه محتوا هر كدام در يك خط نشان داده شده و در همان اندازه تعين شده باشند. خط آخر براي تعين شكل لبه پنجره است. در مثال تعداد كليك كردن دكمه متد setBorder() فاصله Panel را از چپ، بالا، راست را به تعداد 30 پيكسل و فاصله از پائين را به تعداد 10 پيكسل تنظيم خواهد كرد. در شكل زير فاصله از چپ و راست نشان داده شده.

دستگير كردن دخدادها

هر زماني كه كاربر دكمه موس را فشار مي دهد يا كاركتري تايپ مي كند يك رخداد اتفاق مي افتد و هر شيئ مي تواند از اين اين رخدادها باخبر شود. براي اين كار اين شي بايد يك event listener را تكميل كند. در زي مثال مربوط به كليك كردن دكمه نشان داده شده.

```
public class SwingApplication implements ActionListener {
   ...
      JButton button = new JButton("I'm a Swing button!");
      button.addActionListener(this);
   ....

   public void actionPerformed(ActionEvent e) {
      numClicks++;
      label.setText(labelPrefix + numClicks);
  }}
```

هر دستگير رخداد سه قسمت كد نياز دارد. قسمت اول دستور كامل كردن رابط شنود كننده (listener)، در مثال بالا در خط اول نشان داده شده. قسمت دوم كد براي ثبت شنود كننده براي آن كامپوننت، كه در اين مثال كلاس JButton است. در قسمت آخر كد براي اجرا در زماني كه رخداد مورد نظر اتفاق مي افتد نوشته مي شود. در اين

مثال زماني كه دكمه كليك مي شود يكي به شمارنده مورد نظر مي افزايد و آنر را در برچسب متن نشان مي دهد.
كامپونتهاي Swing انواع رخدادهاي را پشتيباني مي كنند كه در جدول زير ليست آنها را مشاهده مي كنيد.

Act that Results in the Event	Listener Type
زماني كه كاربر كليدي را فشار اده يا منوي را انتخاب ميكند	ActionListener
كاربر يك فريم را مي بندد	WindowListener
اگر كاربر دكمه موس را روي يكي از كامپونتهاي گرافيكي فشار دهد	MouseListener
اگر كاربر موس را روي كامپونتها انتقال دهد	MouseMotionListener
زماني كه كامپونت قابل ديدن شود	ComponentListener
زماني كه كامپونت محل تمركز (focus) صفحه كليد را بدست آورد	FocusListener
زماني كه جدول يا ليست انتخابي تغير كند	ListSelectionListener
زماني كه هر كدام از خصيصه هاي (property) كامپونت گرافيكي تغير كند.	PropertyChangeListener

توجه داشته باشيد كه كدهاي دستگير كردن رخدادها همگي در داخل يك نخ تنها اجرا مي شوند و نام آن event-dispatching thread مي باشد.

مثال تعداد كليك كردن دكمه:

```
//SwingApplication.java is a 1.4 example that requires no other files.
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingApplication implements ActionListener {
    private static String labelPrefix = "Number of button clicks: ";
    private int numClicks = 0;
    final JLabel label = new JLabel(labelPrefix + "0    ");

    //Specify the look and feel to use.  Valid values:
    //null (use the default), "Metal", "System", "Motif", "GTK+"
    final static String LOOKANDFEEL = null;

    public Component createComponents() {
        JButton button = new JButton("I'm a Swing button!");
        button.setMnemonic(KeyEvent.VK_I);
        button.addActionListener(this);
        label.setLabelFor(button);
```

```
        /*
         * An easy way to put space between a top-level container
         * and its contents is to put the contents in a JPanel
         * that has an "empty" border.
         */
        JPanel pane = new JPanel(new GridLayout(0, 1));
        pane.add(button);
        pane.add(label);
        pane.setBorder(BorderFactory.createEmptyBorder(
                                30, //top
                                30, //left
                                10, //bottom
                                30) //right
                                );
        return pane;
    }
    public void actionPerformed(ActionEvent e) {
        numClicks++;
        label.setText(labelPrefix + numClicks);
    }
    private static void initLookAndFeel() {
        String lookAndFeel = null;

        if (LOOKANDFEEL != null) {
            if (LOOKANDFEEL.equals("Metal")) {
                lookAndFeel =
UIManager.getCrossPlatformLookAndFeelClassName();
            } else if (LOOKANDFEEL.equals("System")) {
                lookAndFeel = UIManager.getSystemLookAndFeelClassName();
            } else if (LOOKANDFEEL.equals("Motif")) {
                lookAndFeel = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
            } else if (LOOKANDFEEL.equals("GTK+")) { //new in 1.4.2
                lookAndFeel = "com.sun.java.swing.plaf.gtk.GTKLookAndFeel";
            } else {
                System.err.println("Unexpected value of LOOKANDFEEL specified:
"
                                   + LOOKANDFEEL);
                lookAndFeel =
UIManager.getCrossPlatformLookAndFeelClassName();
            }

            try {
                UIManager.setLookAndFeel(lookAndFeel);
            } catch (ClassNotFoundException e) {
                System.err.println("Couldn't find class for specified look and feel:"
                                   + lookAndFeel);
                System.err.println("Did you include the L&F library in the class
path?");
```

```
            System.err.println("Using the default look and feel.");
        } catch (UnsupportedLookAndFeelException e) {
            System.err.println("Can't use the specified look and feel ("
                               + lookAndFeel
                               + ") on this platform.");
            System.err.println("Using the default look and feel.");
        } catch (Exception e) {
            System.err.println("Couldn't get specified look and feel ("
                               + lookAndFeel
                               + "), for some reason.");
            System.err.println("Using the default look and feel.");
            e.printStackTrace();
        }
    }
}

 private static void createAndShowGUI() {
    //Set the look and feel.
    initLookAndFeel();

    //Make sure we have nice window decorations.
    JFrame.setDefaultLookAndFeelDecorated(true);

    JFrame frame = new JFrame("SwingApplication");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    SwingApplication app = new SwingApplication();
    Component contents = app.createComponents();
    frame.getContentPane().add(contents, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
}

public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createAndShowGUI();
        }
    });
}
}
```

استفاده از برچسبها

به كمك كلاس JLabel شما مي توانيد متنها و تصاويري كه قابل انتخاب نيستند را نمايش داده. اگر مي خواهيد متنها وتصاوير بصورت محاوره اي بوده و داراي حالت باشند از دكمه ها استفاده كنيد.به كمك كدهاي html ميتوانيد برچسبها با رنگها،

قلمها و خطهاي مختلف ايجاد كنيد .اگر متن شما يك رنگ و قلم دارد شما مي توانيد از سربار پردازش html اجتناب كرده و از متدهاي setForeground() يا setFont() استفاده كنيد. در زير كدهاي مربوط به يك مثال را مشاهده مي كنيد كه سه برچسب را نمايش مي دهد و پنجره به سه سطر با ارتفاع يكسان تقسيم شده.

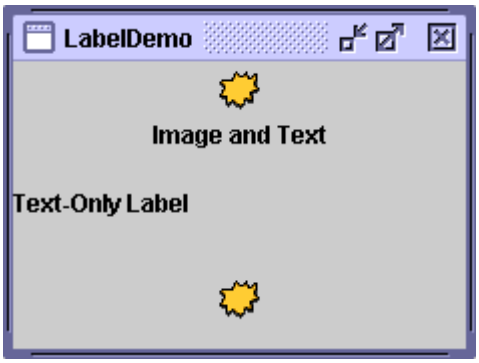
```
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.ImageIcon;
public class LabelDemo extends JPanel {
    public LabelDemo() {
        super(new GridLayout(3,1));  //3 rows, 1 column
        JLabel label1, label2, label3;

        ImageIcon icon = createImageIcon("images/middle.gif",
                                         "a pretty but meaningless splat");
        //Create the first label.
        label1 = new JLabel("Image and Text",
                            icon,
                            JLabel.CENTER);
        //Set the position of its text, relative to its icon:
        label1.setVerticalTextPosition(JLabel.BOTTOM);
        label1.setHorizontalTextPosition(JLabel.CENTER);
       //Create the other labels.
        label2 = new JLabel("Text-Only Label");
        label3 = new JLabel(icon);
        label1.setToolTipText("A label containing both image and text");
        label2.setToolTipText("A label containing only text");
        label3.setToolTipText("A label containing only an image");
        add(label1);
        add(label2);
        add(label3);
    }
    /** Returns an ImageIcon, or null if the path was invalid. */
    protected static ImageIcon createImageIcon(String path,
                                               String description) {
        java.net.URL imgURL = LabelDemo.class.getResource(path);
        if (imgURL != null) {
            return new ImageIcon(imgURL, description);
        } else {
            System.err.println("Couldn't find file: " + path);
            return null;
        }
    }
    private static void createAndShowGUI() {
```

```
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame("LabelDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        LabelDemo newContentPane = new LabelDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);
        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

در اغلب اوقات يك برچسب ساير كامپوننتهاي گرافيكي مثل جعبه متن را شرح مي دهد و شما براي بهبود قدرت دستيابي برنامه خود مي توانيد از متد setLableFor() براي شناساي برچسب آن كامپوننت استفاده كنيد. يك مثال در زير نوشته شده.

```
amountLabel.setLabelFor(amountField);
```

پس از اجراي برنامه خروجي به شكل زير خواهد بود.



مثال تبديل درجه سلسيوس به فارنهايت:

در اين مثال براي تبديل و نشان دادن درجه حرارات سلسيوس به فارنهايت از يك دكمه و جعبه متن به همراه دو برچسب استفاده كرده ايم. كد زير يك جعبه متن در جاوا با عدد 5 براي سازنده ايجاد مي كند.اين عدد تعداد ستونها را در جعبه براي نمايش مناسب آن فونت تنظيم مي كند و تعداد كاراكترها را محود نمي كند.

```
JTextField tempCelsius = null;
...
tempCelsius = new JTextField(5);
```

زماني كه روي دكمه كليك مي شود يا كليد Enter بر روي صفحه كيد فشرده مي شود عمل تبديل انجام شده پس يك شنود كننده اعمال روي دكمه تبديل و جعبه متن تعريف مي كنيم. همانند زير:

```
convertTemp.addActionListener(this);
tempCelsius.addActionListener(this);
...
public void actionPerformed(ActionEvent event) {
        //Parse degrees Celsius as a double and convert to Fahrenheit.
    int tempFahr = (int)((Double.parseDouble(tempCelsius.getText()))
                    * 1.8 + 32);
        fahrenheitLabel.setText(tempFahr + " Fahrenheit");
}
```

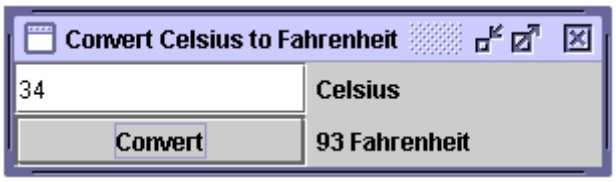
پس از فشار دكمه يا زدن كليد Enter متد getText() از كلاس جعبه متن عدد وارد شده از طرف كاربر را مي دهد.سپس با متد parseDouble() عدد متني را به عدد اعشاري تبديل مي كند.در آخر به كمك متد setText() عدد محاسبه شده را به كاربر نشان مي دهد. در زير كد كامل اين مثال نشان داده شده.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CelsiusConverter implements ActionListener {
    JFrame converterFrame;
    JPanel converterPanel;
    JTextField tempCelsius;
    JLabel celsiusLabel, fahrenheitLabel;
    JButton convertTemp;

    public CelsiusConverter() {
        //Create and set up the window.
        converterFrame = new JFrame("Convert Celsius to Fahrenheit");
        converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        converterFrame.setSize(new Dimension(120, 40));

        converterPanel = new JPanel(new GridLayout(2, 2));

        //Add the widgets.
        addWidgets();

        //Set the default button.
        converterFrame.getRootPane().setDefaultButton(convertTemp);

        //Add the panel to the window.
        converterFrame.getContentPane().add(converterPanel,
BorderLayout.CENTER);

        //Display the window.
        converterFrame.pack();
```

```
        converterFrame.setVisible(true);
    }

    //Create and add the widgets.

    private void addWidgets() {
        //Create widgets.
        tempCelsius = new JTextField(2);
        celsiusLabel = new JLabel("Celsius", SwingConstants.LEFT);
        convertTemp = new JButton("Convert");
        fahrenheitLabel = new JLabel("Fahrenheit", SwingConstants.LEFT);

        //Listen to events from the Convert button.
        convertTemp.addActionListener(this);

        //Add the widgets to the container.
        converterPanel.add(tempCelsius);
        converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);
        converterPanel.add(fahrenheitLabel);

        celsiusLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        fahrenheitLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
    }

    public void actionPerformed(ActionEvent event) {
        //Parse degrees Celsius as a double and convert to Fahrenheit.
        int tempFahr = (int)((Double.parseDouble(tempCelsius.getText()))
                        * 1.8 + 32);
        fahrenheitLabel.setText(tempFahr + " Fahrenheit");
    }

    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        CelsiusConverter converter = new CelsiusConverter();
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
```

}

پس از اجرا فرم به شكل زير خواهد بود.



مثال بهبود يافته مبدل:

در اين مثال از متن رنگي و دكمه پيش فرز به همراه icon استفاده كرده ايم.

اضافه كردن HTML :

چندين روش براي فرمت دهي اجزائ گرافيكي Swing وجود دارد و يكي از اين روشها استفاده از تگهاي html است. دكمه ها،برچسبها ، متنها وساير كامپونتها مي توانند به كمك كدهاي html براي نمايش فرمت دهي شوند.

در مثال مبدل پيشرفته تر به كمك تگها مي توانيم رنگ و اندازه متن خروجي را تعين كنيم. همان طور كه در كد زير مشاهده مي كنيد رنگ متن آبي شده و با 176&# نماد درجه ار نشان مي دهيم.

```
fahrenheitLabel.setText("<html><font Color=blue>" +
                        tempFahr + "&#176 </font> Fahrenheit</html>");
```

اضافه كردن يك icon : بعضي از كامپونتهاي Swing اين قابليت را دارند كه به همراه يك icon نمايش داده شوند. شي icon رابط براي چسبانده icon به كامپونتها و با نام ImageIcon است. فرمت تصوير هاي icon بايد از انواع GIF,JPEG,PNG باشد.كد زير براي گذاشتن iconدر داخل دكمه convertTemp است.

```
ImageIcon convertIcon = createImageIcon("images/convert.gif",
                                        "Convert temperature");
...
convertTemp = new JButton(icon);
```

در متدcreateImageIcon() اولين آرگومان محل و نام فايل icon براي بار گذاري را مشخص كرده و آرگومان دوم شرح براي استفاده از icon است.

تنظيم دكمه پيش فرز: يك دكمه در ظرف سطح بالا مي تواند به عنوان دكمه پيش فرز در نظر گرفته شود. دكمه پيش فرز در نمايش مشخص تر از بقيه دكمه ها است و كليد Enter زده شده صفحه كليد را به طور پيش فرز مي گيرد. براي قرار دادن دكمه convertTemp به عنوان دكمه پشي فرز از كد زير استفاده مي كنيم.

```
converterFrame.getRootPane().setDefaultButton(convertTemp);
```

RootPane در هر ظرف كامپونتهاي گرافيكي براي مديريت جزئيات مثل پانل محتوا و منوها است.

ايجاد متن فرمت شده: در برنامه قبلي مبدل ما در جعبه متن مي توانستيم الفبا را نيز وارد كنيم و اين در زمان اجرا يك استثناء صادر مي كرد. در اين مثال به جاي JTextField از JFormatedTextField كمك گرفته ايم كه يك راه براي قانونمند كردن مجموعه ورودي هاي كاربر است. در كد زير با استفاده از java.text.DecimalFormat مطمعن مي شويم كه ورودي كاربر فقط اعداد را قبول مي كند.

```
//Create the format for the text field and the formatted text field
tempCelsius = new JFormattedTextField(
                        new java.text.DecimalFormat("##0.0#"));
```

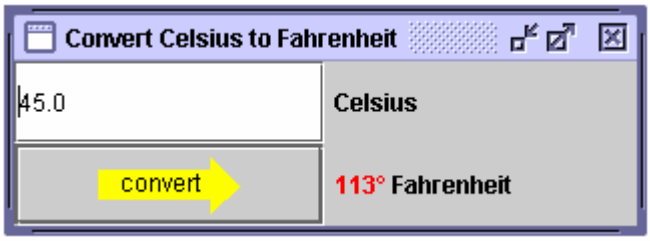
حال کد کامل این مثال را در پائین مشاهده مي نماید.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.NumberFormatter;
import java.text.ParseException;
import java.text.DecimalFormat;
import java.net.URL;

public class CelsiusConverter2 implements ActionListener {
    JFrame converterFrame;
    JPanel converterPanel;
    JFormattedTextField tempCelsius;
    JLabel celsiusLabel, fahrenheitLabel;
    JButton convertTemp;

    public CelsiusConverter2() {
        //Create and set up the window.
        converterFrame = new JFrame("Convert Celsius to Fahrenheit");
        converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        converterFrame.setSize(new Dimension(120, 40));

        //Create and set up the panel.
        converterPanel = new JPanel(new GridLayout(2, 2));

        //Add the widgets.
        addWidgets();

        //Set the default button.
        converterFrame.getRootPane().setDefaultButton(convertTemp);

        //Add the panel to the window.
        converterFrame.getContentPane().add(converterPanel,
BorderLayout.CENTER);

        //Display the window.
        converterFrame.pack();
        converterFrame.setVisible(true);
      }
```

```
    private void addWidgets() {
        //Create the widgets.
        ImageIcon convertIcon = createImageIcon("images/convert.gif",
            "Convert temperature");

        //Create the format for the Celsius text field.
        tempCelsius = new JFormattedTextField(new
DecimalFormat("##0.0#"));

tempCelsius.setFocusLostBehavior(JFormattedTextField.COMMIT_OR_REVERT
);

        //Set and commit the default temperature.
        try {
            tempCelsius.setText("37.0");
            tempCelsius.commitEdit();
        } catch(ParseException e) {
            //Shouldn't get here unless the setText value doesn't agree
            //with the format set above.
            e.printStackTrace();
        }

        celsiusLabel = new JLabel("Celsius", SwingConstants.LEFT);
        convertTemp = new JButton(convertIcon);

        fahrenheitLabel = new JLabel("Fahrenheit", SwingConstants.LEFT);

        celsiusLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        fahrenheitLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));

        //Listen to events from the Convert button and
        //the tempCelsius text field.
        convertTemp.addActionListener(this);
        tempCelsius.addActionListener(this);

        //Add the widgets to the container.
        converterPanel.add(tempCelsius);
        converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);
        converterPanel.add(fahrenheitLabel);
    }

    public void actionPerformed(ActionEvent event) {
        String eventName = event.getActionCommand();
        //Parse degrees Celsius as a double and convert to Fahrenheit.
        int tempFahr = (int)((Double.parseDouble(tempCelsius.getText()))
                        * 1.8 + 32);
```

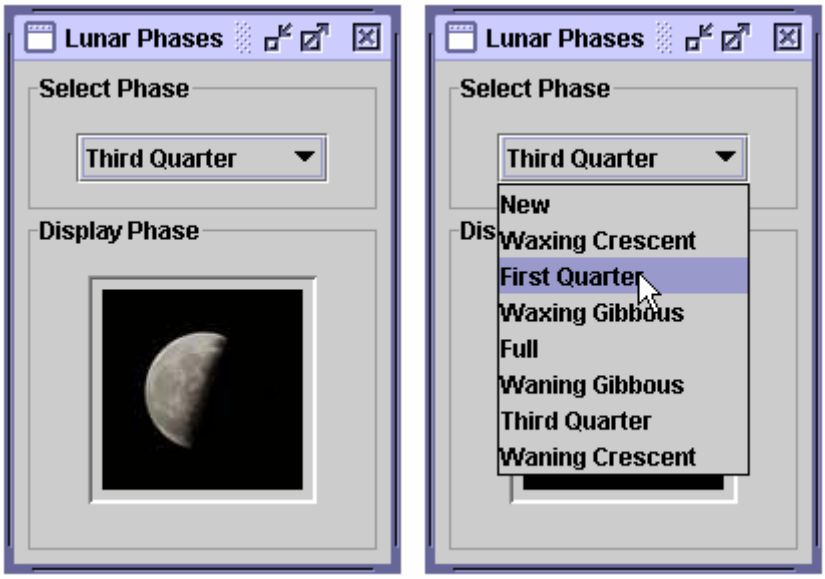
```
        //Set fahrenheitLabel to the new value and set font color
        //based on the temperature.
        if (tempFahr <= 32) {
            fahrenheitLabel.setText("<html><font Color=blue>" +
                    tempFahr + "&#176 </font> Fahrenheit</html>");
        } else if (tempFahr <= 80) {
            fahrenheitLabel.setText("<html><font Color=green>" +
                    tempFahr + "&#176 </font> Fahrenheit</html>");
        } else {
            fahrenheitLabel.setText("<html><font Color=red>" +
                    tempFahr + "&#176 </font> Fahrenheit</html>");
        }
    }

    /** Returns an ImageIcon, or null if the path was invalid. */
    protected static ImageIcon createImageIcon(String path,
                                               String description) {
        java.net.URL imgURL = CelsiusConverter2.class.getResource(path);
        if (imgURL != null) {
            return new ImageIcon(imgURL, description);
        } else {
            System.err.println("Couldn't find file: " + path);
            return null;
        }
    }

    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        CelsiusConverter2 converter = new CelsiusConverter2();
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

بعد از اجرا فرم به شکل زیر ظاهر خواهد شد.

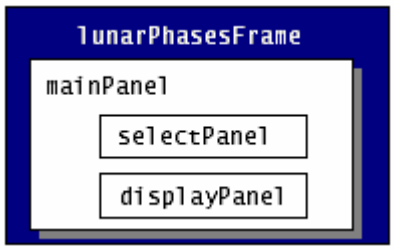


مثال فازهاي ماه

در اين برنامه كاربر فاز هاي مختلف تكميل شده ماه را از combo box انتخاب مي كند و همان طور كه در شكل زير مشاهده ميكنيد آن فاز در قسمت مربوط به نمايش فاز نشان داده مي شود.



اين برنامه همان طور كه در شكل زير مشاهده مي كنيد از سه پانل تشكيل شده.



كدهاي كه در پائين نشان داده در داخل سازنده برنامه سه پانل ايجاد كرده سو اضافه كردن دو زير پانل (پانل انتخاب و پانل نمايش فاز) به پانل اصلي است. كار پانل اصلي با استفاده از مدير طرح بندي (layout manager) قرار دادن درست موقعيت اجزاء زير پانلها است. مدير طرح بند پيش فرز براي JPanel طرح بند FlowLayout است و به طور ساده اجزائ گرافيكي را پشت سر هم مي چيند.

```
//Create the phase selection and display panels.
selectPanel = new JPanel();
```

```
displayPanel = new JPanel();

//Add various widgets to the sub panels.
addWidgets();

//Create the main panel to contain the two sub panels.
mainPanel = new JPanel();
mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.PAGE_AXIS));
mainPanel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));

//Add the select and display panels to the main panel.
mainPanel.add(selectPanel);
mainPanel.add(displayPanel);
```

در كدهاي بالا از مدير طرح بندي BoxLayout براي تنظيم دقيق زير پانل ها استفاه كرده ايم.

استفاده از مدير طرح بندي:

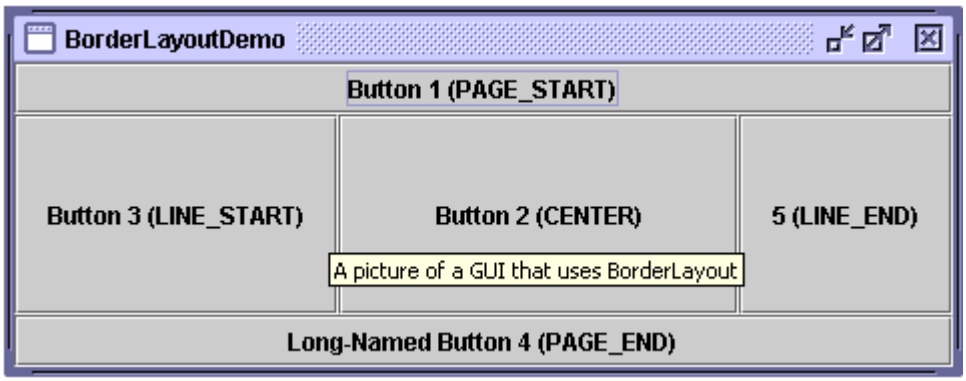
در جاوا از سه نوع مدير طرح بند شامل FlowLayout، BoxLayuot، BorderLayout، GridLayout، GridBagLayout و SpringLayout پشتيباني مي شود. همان طور كه قبلاً گفته شد به طور پيش فرز تمام اشياء از نوع JPlanel به طور پيش فرز از FlowLayout استفاده مي كنند. همين طور ساير نگه دارنده هاي محتوا مثل ظرف اصلي در اشياء JApplet، JDialog و JFrame به صورت پيش فرز از BorderLayout استفاده مي كند. زماني كه شما درباره مدير طرح بندي تفكر مي كنيد يك روش وجود دارد و آن قاعده چگونه گي ايجاد يك JPanel يا اضافه كردن كامپوننت براي يك نگه دارنده محتوا مي باشد. يك روش ديگر متفاوت استفاده از متد setLayout() براي مشخص كردن نحوه كار نگه دارنده محتوا است. براي مثال كد زير ايجاد پانل به كمك BorderLayout را نشان داده:

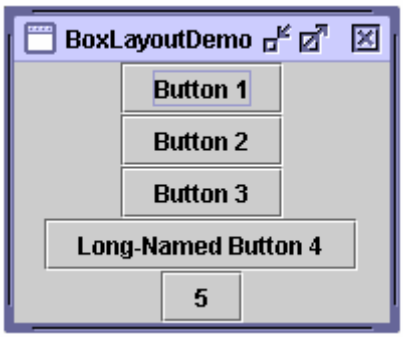
```
JPanel pane = new JPanel(new BorderLayout());
```

مثال زير درباره تنظيم مدير طرح بندي نگه دارنده محتواي پيش فرز است.

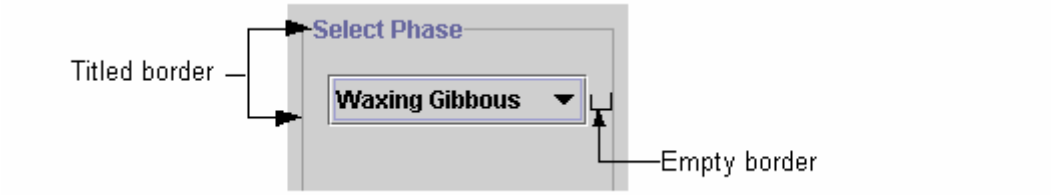
```
Container contentPane = frame.getContentPane();
contentPane.setLayout(new FlowLayout());
```

زماني كه شما به يك پانل يا نگه دارنده محتوا كامپوننت گرافيكي اضافه مي كنيد، شما مي توانيد روش اضافه كردن را جدا از مدير طرح بندي تعين كنيد. در زير سه مثال از انواع مدير طرح بندي شامل BorderLayout، BoxLayout و FlowLayout براي پنج دكمه در يك فريم به صورت پشت سر هم نشان داده ايم:





لبه هاي مركب: در مثالهاي گذشته ما لبه هاي ساده را براي ايجاد يك ميانگير فاصله بين دور اجزاء استفاده كرديم. در اين مثال هر دو زير پانل selectPanel و displayPanel از لبه هاي مركب استفاده كرده و شامل لبه عنوان دار (شامل لبه حاشيه به همراه يك عنوان) و لبه خالي (يك لبه با فضاي اضافي) است. اين دو لبه در شكل زير نشان داده شده.



كد مورد نياز براي selectPanel به فرم زير است. پانل displayPanel به هيمن روش لبه خود را تنظيم مي كند.

```
// Add border around the select panel
selectPanel.setBorder(BorderFactory.createCompoundBorder(
                      BorderFactory.createTitledBorder("Select Phase"),
                      BorderFactory.createEmptyBorder(5,5,5,5)));
```

Combo Box : در اين مثال از جعبه متن تركيبي (combo box) براي نشان دادن يك گروه از انتخابهاي استفاده مي كنيم. بجاي combo box مي توان از جعبه متن براي نوشتن هر كدام استفاده كرد. كدهاي زير در مثال يك combo box غير قابل ويرايش را ايجاد مي كند.

```
JComboBox phaseChoices = null;
...
//Create combo box with lunar phase choices.
String[] phases = { "New", "Waxing Crescent", "First Quarter",
                    "Waxing Gibbous", "Full", "Waning Gibbous",
                    "Third Quarter", "Waning Crescent" };
phaseChoices = new JComboBox(phases);
phaseChoices.setSelectedIndex(START_INDEX);
```

شما به جاي آرايه اي از رشته مي توانيد از آرايه اي از icon ها يا ساختار داده منظم در ايجاد combo box استفاده كنيد. متد setSelectedIndex() مشخص مي كند كه در زمان شروع اجراي برنامه كدام يك از قلمها بايد نمايش داده شود. زماني كه كاربر يك قلم را از combo box انتخاب مي كند يك رخداد اتفاق مي افتد و كد زير در مثال وظيفه ثبت و كامل كردن يك شنود كننده رخداد روي combo box را دارد. زماني كه قلم جديد انتخاب مي شود، تصوير اين قلم نشان داده شده و برچسب به روز رساني مي شود.

```
phaseChoices.addActionListener(this);
...
public void actionPerformed(ActionEvent event) {
    if ("comboBoxChanged".equals(event.getActionCommand())) {
        //Update the icon to display the new phase
        phaseIconLabel.setIcon(images[phaseChoices.getSelectedIndex()]);
    }
}
```

تصاوير چند گانه: در مثال قبلي نحوه اضافه كردن يك تصوير تنها به يك دكمه را مشاهده كرديد. در اين مثال از هشت تصوير استفاده شده و در هر زمان تصوير يكي از آنها به نمايش در مي آيد. همچنين مي توانيم تمام اين تصاوير يكجا بار گذاري كرده يا در زمان لازمه يك از آنها را بار گذاري كنيم. در زير نحوه بار گذاري يكجاي هشت تصوير را ملاحظه مي كنيد.

```
final static int NUM_IMAGES = 8;
final static int START_INDEX = 3;

ImageIcon[] images = new ImageIcon[NUM_IMAGES];
...
//Get the images and put them into an array of ImageIcon.
for (int i = 0; i < NUM_IMAGES; i++) {
    String imageName = "images/image" + i + ".jpg";
    System.out.println("getting image: " + imageName);
    URL iconURL = LunarPhases.class.getResource(imageName);
    ImageIcon icon = new ImageIcon(iconURL);
```

```
    images[i] = icon;
}
```

استفاده از متد setResource() براي جستجوي مسير كلاسهاي برنامه بمنظور پيدا كردن فايلهاي تصوير است. كد كامل برنامه در زير نوشته شده.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.URL;

public class LunarPhases implements ActionListener {
    final static int NUM_IMAGES = 8;
    final static int START_INDEX = 3;
    ImageIcon[] images = new ImageIcon[NUM_IMAGES];
    JPanel mainPanel, selectPanel, displayPanel;
    JComboBox phaseChoices = null;
    JLabel phaseIconLabel = null;

    public LunarPhases() {
        //Create the phase selection and display panels.
        selectPanel = new JPanel();
        displayPanel = new JPanel();

        //Add various widgets to the sub panels.
        addWidgets();
        //Create the main panel to contain the two sub panels.
        mainPanel = new JPanel();
        mainPanel.setLayout(new                    BoxLayout(mainPanel,
BoxLayout.PAGE_AXIS));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        //Add the select and display panels to the main panel.
        mainPanel.add(selectPanel);
        mainPanel.add(displayPanel);
    }

    // Get the images and set up the widgets.
      private void addWidgets() {
        //Get the images and put them into an array of ImageIcons.
        for (int i = 0; i < NUM_IMAGES; i++) {
            images[i] = createImageIcon("/images/image" + i + ".jpg");
        }

        //Create a label for displaying the moon phase images
        phaseIconLabel = new JLabel();
        phaseIconLabel.setHorizontalAlignment(JLabel.CENTER);
        phaseIconLabel.setVerticalAlignment(JLabel.CENTER);
        phaseIconLabel.setVerticalTextPosition(JLabel.CENTER);
```

```
        phaseIconLabel.setHorizontalTextPosition(JLabel.CENTER);
        phaseIconLabel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLoweredBevelBorder(),
            BorderFactory.createEmptyBorder(5,5,5,5)));

        phaseIconLabel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createEmptyBorder(0,0,10,0),
            phaseIconLabel.getBorder()));

        //Create a combo box with lunar phase choices.
        String[] phases = { "New", "Waxing Crescent", "First Quarter",
                            "Waxing Gibbous", "Full", "Waning Gibbous",
                            "Third Quarter", "Waning Crescent" };
        phaseChoices = new JComboBox(phases);
        phaseChoices.setSelectedIndex(START_INDEX);

        //Display the first image.
        phaseIconLabel.setIcon(images[START_INDEX]);
        phaseIconLabel.setText("");

        //Add a border around the select panel.
        selectPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Select Phase"),
            BorderFactory.createEmptyBorder(5,5,5,5)));
        //Add a border around the display panel.
        displayPanel.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createTitledBorder("Display Phase"),
            BorderFactory.createEmptyBorder(5,5,5,5)));
        //Add moon phases combo box to select panel and image label.
        displayPanel.add(phaseIconLabel);
        selectPanel.add(phaseChoices);
        //Listen to events from the combo box.
        phaseChoices.addActionListener(this);
    }

    public void actionPerformed(ActionEvent event) {
        if ("comboBoxChanged".equals(event.getActionCommand())) {
            //Update the icon to display the new phase.
            phaseIconLabel.setIcon(images[phaseChoices.getSelectedIndex()]);
        }
    }

    /** Returns an ImageIcon, or null if the path was invalid. */
    protected static ImageIcon createImageIcon(String path) {
        java.net.URL imageURL = LunarPhases.class.getResource(path);

        if (imageURL == null) {
            System.err.println("Resource not found: "
```

```
                        + path);
            return null;
        } else {
            return new ImageIcon(imageURL);
        }
    }

    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create a new instance of LunarPhases.
        LunarPhases phases = new LunarPhases();

        //Create and set up the window.
        JFrame lunarPhasesFrame=new JFrame("Lunar Phases");
        lunarPhasesFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        lunarPhasesFrame.setContentPane(phases.mainPanel);

        //Display the window.
        lunarPhasesFrame.pack();
        lunarPhasesFrame.setVisible(true);
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

مثال راي دادن:

در اين مثال بيشتر روي دكمه هاي راديوي (radio buttons) و جعبه هاي گفتگو (dialogs) كار خواهيم كرد. در اين مثال كابر مي تواند يك دكمه راديوي را انتخاب كرده و به كمك دكمه راي يك راي را صادر كند. سپس يك جعبه گفتگو با پيام ظاهر شده و اطلاعات را نشان مي دهد.

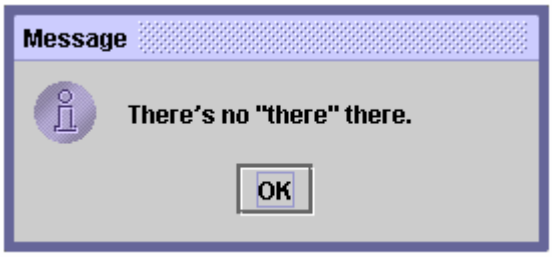
دكمه هاي راديوي: در اين مثال يك شنود كننده رخداد كليك كردن دكمه راي را در ظرف كنترل مي كند و هر زمان كه آن كليك شد برنامه تعين مي كند كه كدام دكمه راديوي كليك شده و دستورات لازم آن را انجام مي دهد. براي هر گروه از دكمه هاي راديوي شما نياز داريد به ايجاد نمونه شئ از ButtonGroup و اضافه كردن هر دكمه راديوي به داخل آن. اين شي دكمه راديوي را كه قبلآ انتخاب شده را زماني كه دكمه راديوي ديگري انتخاب مي كنيم غير فعال مي كند. دكمه هاي راديوي مي توانند دكمه انتخاب شده پيش فرز نداشته باشند.

كدهاي زير دكمه هاي راديوي و شي ButtonGroup را ايجاد كرده و از html براي فرمت دهي متنها استفاده مي كند. متد setActionCommand() با توجه به قلم هر دكمه راديوي متن جعبه گفتگو را مشخص مي كند.

```
final int numButtons = 4;
JRadioButton[] radioButtons = new JRadioButton[numButtons];
final ButtonGroup group = new ButtonGroup();
...
final String defaultMessageCommand = "default";
final String yesNoCommand = "yesno";
final String yeahNahCommand = "yeahnah";
final String yncCommand = "ync";

radioButtons[0] = new JRadioButton("<html>Candidate 1:
    <font color=red>Sparky the Dog</font></html>");
radioButtons[0].setActionCommand(defaultMessageCommand);

radioButtons[1] = new JRadioButton("<html>Candidate 2:
    <font color=green>Shady Sadie</font></html>");
radioButtons[1].setActionCommand(yesNoCommand);

radioButtons[2] = new JRadioButton("<html>Candidate 3:
    <font color=blue>R.I.P. McDaniels</font></html>");
radioButtons[2].setActionCommand(yeahNahCommand);

radioButtons[3] = new JRadioButton("<html>Candidate 4:
    <font color=maroon>Duke the Java<font size=-2><sup>TM</sup>
    </font size> Platform Mascot</font></html>");
radioButtons[3].setActionCommand(yncCommand);

for (int i = 0; i < numButtons; i++) {
   group.add(radioButtons[i]);
}
//Select the first button by default.
radioButtons[0].setSelected(true);
```

جعبه هاي گفتگو: در مثالهاي قبلي ظرف سطح بالاي شما هميشه يك JFrame بود. براي ايجاد جعبه هاي گفتگوي مختلف و استاندارد بايد از كلاس JOptionPane استفاده كرد و اين جعبه هاي گفتگو بصورت مقيد (modal) مي باشند. زماني كه يك جعبه گفتگوي modal در حال نمايش است از وارد كردن اطلاعات كاربر به ساير پنجره هاي برنامه جلوگيري مي شود. كد زير يك مثال كوچك در باره كلاس JOptionPane است و جعبه گفتگوي حاصل از آن نشان داده شده.

```
JOptionPane.showMessageDialog(frame, "There's no \"there\" there.");
```




```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class VoteDialog extends JPanel {
    JLabel label;
    JFrame frame;
    String simpleDialogDesc = "The candidates";
    public VoteDialog(JFrame frame) {
        super(new BorderLayout());
        this.frame = frame;
        JLabel title;
        //Create the components.
        JPanel choicePanel = createSimpleDialogBox();
        System.out.println("passed createSimpleDialogBox");
      title = new JLabel("Click the \"Vote\" button"
                         + " once you have selected a candidate.",
                         JLabel.CENTER);
       label = new JLabel("Vote now!", JLabel.CENTER);
        label.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
choicePanel.setBorder(BorderFactory.createEmptyBorder(20,20,5,20));
        //Lay out the main panel.
        add(title, BorderLayout.NORTH);
        add(label, BorderLayout.SOUTH);
        add(choicePanel, BorderLayout.CENTER);
    }

    void setLabel(String newText) {
        label.setText(newText);
    }
    private JPanel createSimpleDialogBox() {
        final int numButtons = 4;
        JRadioButton[] radioButtons = new JRadioButton[numButtons];
        final ButtonGroup group = new ButtonGroup();
        JButton voteButton = null;
        final String defaultMessageCommand = "default";
        final String yesNoCommand = "yesno";
        final String yeahNahCommand = "yeahnah";
        final String yncCommand = "ync";
```

```
        radioButtons[0] = new JRadioButton(
          "<html>Candidate 1: <font color=red>Sparky the
Dog</font></html>");
        radioButtons[0].setActionCommand(defaultMessageCommand);

        radioButtons[1] = new JRadioButton(
            "<html>Candidate 2: <font color=green>Shady
Sadie</font></html>");
        radioButtons[1].setActionCommand(yesNoCommand);

        radioButtons[2] = new JRadioButton(
            "<html>Candidate 3: <font color=blue>R.I.P.
McDaniels</font></html>");
        radioButtons[2].setActionCommand(yeahNahCommand);

        radioButtons[3] = new JRadioButton(
            "<html>Candidate 4: <font color=maroon>Duke the Java<font
size=-2><sup>TM</sup></font size> Platform Mascot</font></html>");
        radioButtons[3].setActionCommand(yncCommand);

for (int i = 0; i < numButtons; i++) {
            group.add(radioButtons[i]);
        }
        //Select the first button by default.
        radioButtons[0].setSelected(true);

        voteButton = new JButton("Vote");
       voteButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String command = group.getSelection().getActionCommand();

                //ok dialog
                if (command == defaultMessageCommand) {
                    JOptionPane.showMessageDialog(frame,
                      "This candidate is a dog. Invalid vote.");

                //yes/no dialog
                } else if (command == yesNoCommand) {
                    int n = JOptionPane.showConfirmDialog(frame,
                      "This candidate is a convicted felon. \nDo you still want to
vote for her?",
                      "A Follow-up Question",
                      JOptionPane.YES_NO_OPTION);
                    if (n == JOptionPane.YES_OPTION) {
                        setLabel("OK. Keep an eye on your wallet.");
                    } else if (n == JOptionPane.NO_OPTION) {
                        setLabel("Whew! Good choice.");
```

```
                } else {
                    setLabel("It is your civic duty to cast your vote.");
                }
            //yes/no (with customized wording)
            } else if (command == yeahNahCommand) {
                Object[] options = {"Yes, please", "No, thanks"};
                int n = JOptionPane.showOptionDialog(frame,
                  "This candidate is deceased. \nDo you still want to vote for
him?",
                  "A Follow-up Question",
                  JOptionPane.YES_NO_OPTION,
                  JOptionPane.QUESTION_MESSAGE,
                  null,
                  options,
                  options[0]);
                if (n == JOptionPane.YES_OPTION) {
                    setLabel("I hope you don't expect much from your
candidate.");
                } else if (n == JOptionPane.NO_OPTION) {
                    setLabel("Whew! Good choice.");
                } else {
                    setLabel("It is your civic duty to cast your vote.");
                }
            //yes/no/cancel (with customized wording)
            } else if (command == yncCommand) {
                Object[] options = {"Yes!",
                                    "No, I'll pass",
                                    "Well, if I must"};
                int n = JOptionPane.showOptionDialog(frame,
                                "Duke is a cartoon mascot. \nDo you  "
                                + "still want to cast your vote?",
                                "A Follow-up Question",
                                JOptionPane.YES_NO_CANCEL_OPTION,
                                JOptionPane.QUESTION_MESSAGE,null,options,
                                options[2]);
                if (n == JOptionPane.YES_OPTION) {
                    setLabel("Excellent choice.");
                } else if (n == JOptionPane.NO_OPTION) {
                    setLabel("Whatever you say. It's your vote.");
                } else if (n == JOptionPane.CANCEL_OPTION) {
                    setLabel("Well, I'm certainly not going to make you vote.");
                } else {
                    setLabel("It is your civic duty to cast your vote.");
                }
            }
            return;
        }
    });
```

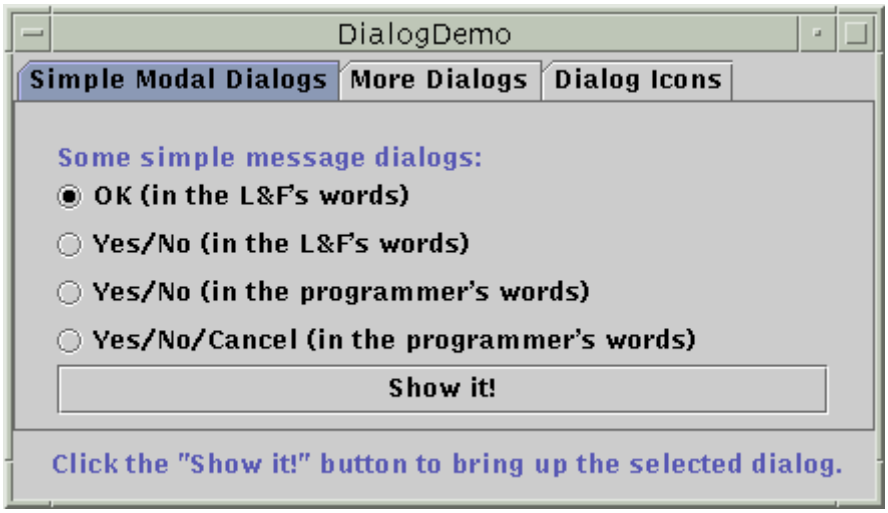
```
        System.out.println("calling createPane");
        return createPane(simpleDialogDesc + ":",
                          radioButtons,
                          voteButton);
    }

    private JPanel createPane(String description,
                              JRadioButton[] radioButtons,
                              JButton showButton) {
        int numChoices = radioButtons.length;
        JPanel box = new JPanel();
        JLabel label = new JLabel(description);

        box.setLayout(new BoxLayout(box, BoxLayout.PAGE_AXIS));
        box.add(label);

        for (int i = 0; i < numChoices; i++) {
            box.add(radioButtons[i]);
        }
        JPanel pane = new JPanel(new BorderLayout());
        pane.add(box, BorderLayout.NORTH);
        pane.add(showButton, BorderLayout.SOUTH);
        System.out.println("returning pane");
        return pane;
    }

    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container contentPane = frame.getContentPane();
        contentPane.setLayout(new GridLayout(1,1));
        contentPane.add(new VoteDialog(frame));

        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

پس از اجرا فرم به صورت زير ظاهر خواهد شد.

نحوه ايجاد جعبه هاي گفتگو

همان طور كه گفته شد براي ايجاد جعبه هاي گفتگوي محاورهاي از كلاس استاندارد JOptionPane استفاده مي كنيم. كلاس ProgerssMonitor همانند جعبه گفتگو وضعيت پيشرفت اعمال را نشان مي دهد. همچنين دو كلاس JColorChooser و JFileChooser جعبه هاي محاوره اي استاندارد براي انتخاب رنگ و فايل هستند. هر جعبه گفتگو وابسته به يك frame است و زمانيكه شئ frame خراب مي شود (از حافظه خارج مي شود) جعبه هاي گفتگوي وابسته به آن هم خراب مي شوند. جعبه هاي گفتگو بصورت مقيد (modal) مي باشند يعني زماني كه يك جعبه گفتگوي modal در حال نمايش است از وارد كردن اطلاعات كاربر به ساير پنجره هاي برنامه جلوگيري مي شود.

كلاس JDialog يك زير كلاس از كلاس java.awt.Dialog بوده و يك جعبه Dialog را به نگه دارنده محتواي ريشه اضافه مي كند و عمل بستن پيش فرز را پشتيباني مي كند. استفاده از كلاس JDialog بسيار شبيه به استفاده از كلاس JFrame بوده و همان قابليتها را دارد. زماني كه از JOptionPane براي جعبه هاي گفتگو استفاده مي كنيد در پيش زمينه همانند اين است كه از JDialogاستفاده مي كنيد. در شكل زير يك Dialog نمونه از يك برنامه را مشاهده مي كنيد.



قابليتهاي JOptionPane

با استفاده از اين كلاس مي توانيد چندين نوع از جعبه هاي گفتوي مختلف و دلخواه را بسازيد و مشخصات آنها را مانند icon هاي جعبه، عنوان و متن جعبه، دكمه ها به طور دلخواه تعين كنيد. همچنين اين امكان نيز وجود دارد كه مكان نمايش آنها را نيز يه طور دلخواه مشخص كنيد. حتي مي توان جعبه گفتگوي option pane را طوري تنظيم كرد كه به جاي گذاشتن آن در JDialog آن را در يك فريم داخلي يا JInteralFrame گذاشت. زمانيكه يك جعبه گفتگو ايجاد مي شود به طور خودكار كد هاي مربوط به Look&Feel براي طرح بندي (layout) مربوط به ظرف اصلي به آن اضافه مي شود. كلاس JOptionPane از چهار icon براي چهار حالت سوال، اطلاعات، اخطار و خطا استفاده مي كند. در شكل زير چهار icon را مشاهده مي كنيد.

ايجاد و نمايش جعبه هاي گفتگوي ساده:

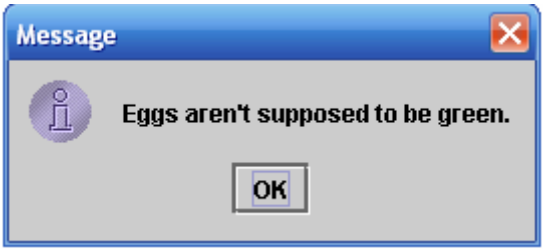
بمنظور ايجاد و نمايش جعبه هاي گفتگو از متد showXxxDialog() در كلاس JOptionPane استفاده كرده و اگر جعبه گفتگو به صورت يك فريم داخلي است يك Interal به آن اضافه مي كنيد همانند showInteralMessageDialog. اگر شما نياز به كنترل جعبه گفتگو در هنگام بستن پنجره داريد و يا مي خواهيد جعبه بصورت modal نباشد شما مي توانيد به صورت مستقيم از معرفي JOptionPane استفاده كرده و آن را به يك نمونه JDialog اضافه كرده و به كمك متد setVisible(true) ظاهر شدن آن را كنترل كنيد.

دو متد مفيد و پر استفاده از showXxxDialog متدهاي showMessageDialog و showOptionDialog هستند. متد اول به صورت ساده اطلاعات را به همراه يك دكمه نشان داده. متد دوم يك جعبه گفتگوي دلخواه را نشان داده كه مي تواند شامل دكمه ها، متنها و اجزائ گرافيكي دلخواه باشد. دو متد باقي مانده كه كمتر از دو متد قيلي استفاده مي شوند متدهاي showConfrimDialog و showInputDialog هستند. متد اول براي سوال كردن از كاربر در باره تائيد چيزي است و دكمه هاي محدودي مثل yes/no دارد. متد دوم بمنظور گرفتن اطلاعات از كاربر به شكلهاي متني يا از combo box به صورت modal است.

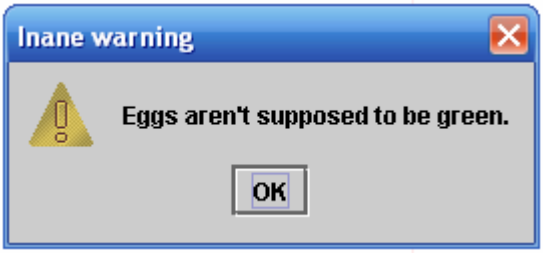
در اغلب این متدها اولین آرگومان نام frame یا ظرف نگهدارنده محتوا است و دومین آرگومان پیغامي که به کاربر نمایش داده خواهد شد است.سومین آرگمان همیشه نوع پیغام را مشخص مي کند و همچنین چهارمین پارامتر icon را براي جعبه گفتگوي مورد نظر تعین مي کند.
حال به چند مثال درباره هر کدام از این متدها به همراه شکلهاي آنها در زیر مي پردازیم.

: showMessageDialog

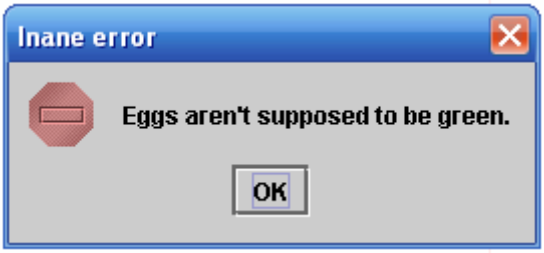
```
//default title and icon
JOptionPane.showMessageDialog(frame,"Eggs aren't supposed to be green.");
```




```
//custom title, warning icon
JOptionPane.showMessageDialog(frame,"Eggs aren't supposed to be green.",
    "Inane warning",
    JOptionPane.WARNING_MESSAGE);
```



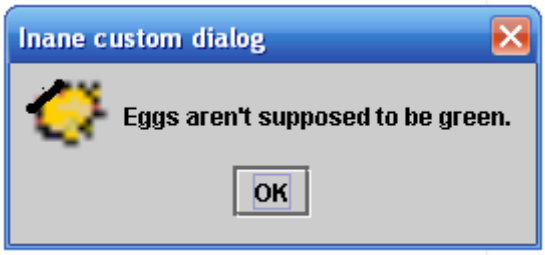

```
//custom title, error icon
JOptionPane.showMessageDialog(frame,"Eggs aren't supposed to be green.",
    "Inane error",
JOptionPane.ERROR_MESSAGE);
```




```
//custom title, no icon
JOptionPane.showMessageDialog(frame,"Eggs aren't supposed to be green.",
    "A plain message",
```

```
    JOptionPane.PLAIN_MESSAGE);
```


```
//custom title, custom icon
JOptionPane.showMessageDialog(frame,"Eggs aren't supposed to be green.",
    "Inane custom dialog",
    JOptionPane.INFORMATION_MESSAGE,icon);
```

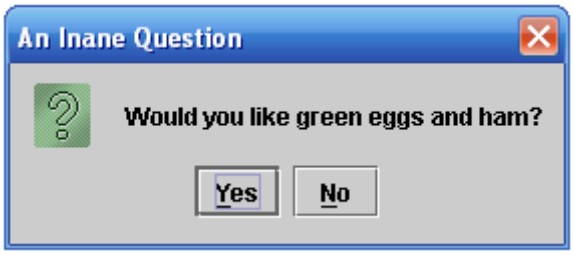


: showOptionDialog

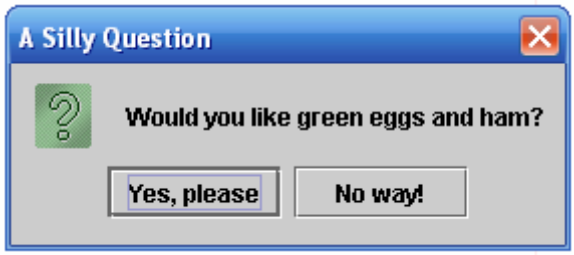
```
//Custom button text
Object[] options = {"Yes, please",
                    "No, thanks",
                    "No eggs, no ham!"};
int n = JOptionPane.showOptionDialog(frame,
    "Would you like some green eggs to go "
    + "with that ham?",
    "A Silly Question",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,
    options,
    options[2]);
```

: JOptionPane

```
//default icon, custom title
int n = JOptionPane.showConfirmDialog(
    frame,"Would you like green eggs and ham?",
    "An Inane Question",JOptionPane.YES_NO_OPTION);
```

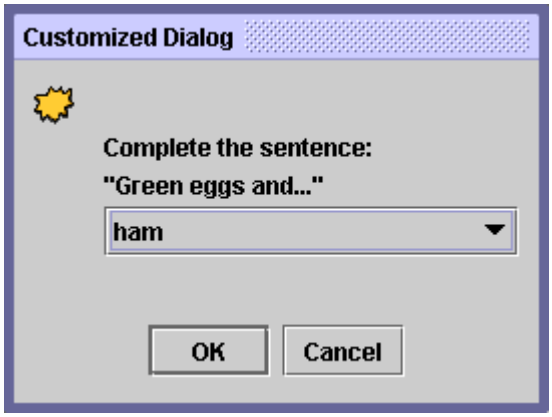



```
Object[] options = {"Yes, please",
                    "No way!"};
int n = JOptionPane.showOptionDialog(frame,
    "Would you like green eggs and ham?",
    "A Silly Question",
    JOptionPane.YES_NO_OPTION,
    JOptionPane.QUESTION_MESSAGE,
    null,     //don't use a custom Icon
    options,  //the titles of buttons
    options[0]); //default button title
```

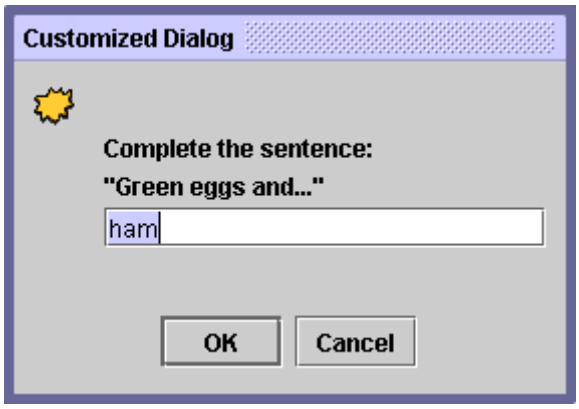


: showInputDialog

```
Object[] possibilities = {"ham", "spam", "yam"};
String s = (String)JOptionPane.showInputDialog(
                    frame,
                    "Complete the sentence:\n"
                    + "\"Green eggs and...\"",
                    "Customized Dialog",
                    JOptionPane.PLAIN_MESSAGE,
                    icon,
                    possibilities,
                    "ham");
```

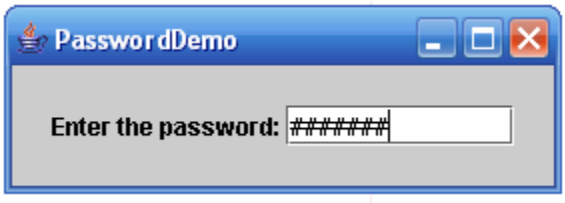


اگر در متد بالا آرايه possibilities راحذف كنيم جعبه گفتگوي ورودي به شكل زير خواهد بود.



گرفتن Password

يكي از زير كلاسهاي كلاس JTextField كلاس JPasswordField مي باشد. از اين كلاس براي گرفتن كلمه هاي عبور در برنامه هاي كه امنيت بالا نياز دارند استفاده مي شود. در هنگام وارد كردن كلمه عبور كاركترهاي كه كاربر وارد مي كند را نشان نمي دهد و به جاي كاركترها يك كاركتر ثابت نشان مي دهد. يك فرم كه در آن از JPasswordFieldاستفاده شده در زير مشاهده مي نمايد.



كدهاي زير نحوه ايجاد و تظيم يك password field را نشان مي دهد.

```
passwordField = new JPasswordField(10);
passwordField.setEchoChar('#');
passwordField.addActionListener(this);
```

عدد 10 در اولين خط نشان دهنده حداكثر طول كامه عبور است. كاركتر # در داخل متد ()setEchoChar كاركتر ظاهر شونده به جاي كلمه عبور را مشخص مي كند.

نحوه استفاده از فريم هاي داخلي و منوها

به كمك كلاس JIntrenalFrameشما مي توانيد يك JFrame را در داخل يك فريم ديگر نمايش دهيد.معمولاً يك فريم داخلي به يك desktop pane اضافه مي شود. خود desktop pane به عنوان يك نگه دارنده محتوا در يك JFrameاستفاده مي شود و يك نمونه از شئ JDesktopPane مي باشد. كلاس JDesktopPane يك زير كلاس از JLayeredPane است و در داخل JLayeredPane يك سري API براي فريم هاي داخلي موجود مي باشد. كار سويچ بين فريمهاي داخلي يا فريم اصلي به كمك اين كلاسها ساده شده و كد هاي زير يك desktop به همراه فريم داخلي آن ايجاد مي كند.

```
desktop = new JDesktopPane();
    createFrame(); //Create first window
    setContentPane(desktop);
    ...
    //Make dragging a little faster but perhaps uglier.
    desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
...
protected void createFrame() {
    MyInternalFrame frame = new MyInternalFrame();
    frame.setVisible(true); //necessary as of 1.3
    desktop.add(frame);
    try {
        frame.setSelected(true);
    } catch (java.beans.PropertyVetoException e) {}
}

...//In the constructor of MyInternalFrame, a JInternalFrame subclass:
static int openFrameCount = 0;
static final int xOffset = 30, yOffset = 30;

public MyInternalFrame() {
    super("Document #" + (++openFrameCount),
          true, //resizable
          true, //closable
          true, //maximizable
          true);//iconifiable
    //...Create the GUI and put it in the window...
    //...Then set the window size or call pack...
    ...
    //Set the window's location.
    setLocation(xOffset*openFrameCount, yOffset*openFrameCount);
}
```

فريم هاي داخلي ظرفهاي سطح بالا نيستند و اين تمايز بين آنها و فريم هاي معمولي است. براي مثال شما يك فريم داخلي را به يك ظرف مثل JDesktopPane اضافه مي كنيد و آنها نمي توانند رخداد هاي مربوط به يك پنجره خارجي را ايجاد كنند. بدليل اين كه فريم هاي داخلي با كدهاي مستقل از زيرساخت (platform) تكميل شده اند

قابليتهاي دارند كه در فريمهاي معمولي اين قابليتها وجود ندارند. يك از اين قابليتها كنترل بيشتر روي روي حالت و تواناي فريم است، همانند دادن شمايل يا icon و بزرگ كردن (maximizing) به صورت برنامه نويسي شده.

قواعد استفاده از Internal Frames

در زير قواعد استفاده از فريم هاي داخلي تك تك بيان شده.

شما بايد اندازه فريم داخلي را تنظيم كنيد: اگر اندازه فريم داخلي را مشخص نكنيد اندازه آن مقدار صفر خواهد گرفت و هرگز قابل ديد نخواهد بود. براي تنظيم اندازه فريم داخلي از يكي سه متد setSize(), pack(), SetBounds() مي توانيد استفاده كنيد.

محل فريم داخلي بايد مشخص گردد: اگر اين كار را انجام ندهيد فريم داخلي در سمت چپ بالاترين نقطه ظرف يعني 0,0 قرار خواهد گرفت.به كمك متد setLocation() يا setBounds() مي توان محل فريم در داخل ظرف را تعين كرد.

براي اضافه كردن يك جزء يا كامپوننت گرافيكي به يك فريم داخلي آنها را بايد به نگه دارنده محتواي فريم هاي داخلي اضافه كنيد: شبيه به اضافه كردن به يك JFrame.

جعبه هاي گفتگو براي فريم داخلي به كمك دو متد JOptionPane() و JInternalFrame() تكميل مي شوند: استفاده از JDialog ناممكن است.

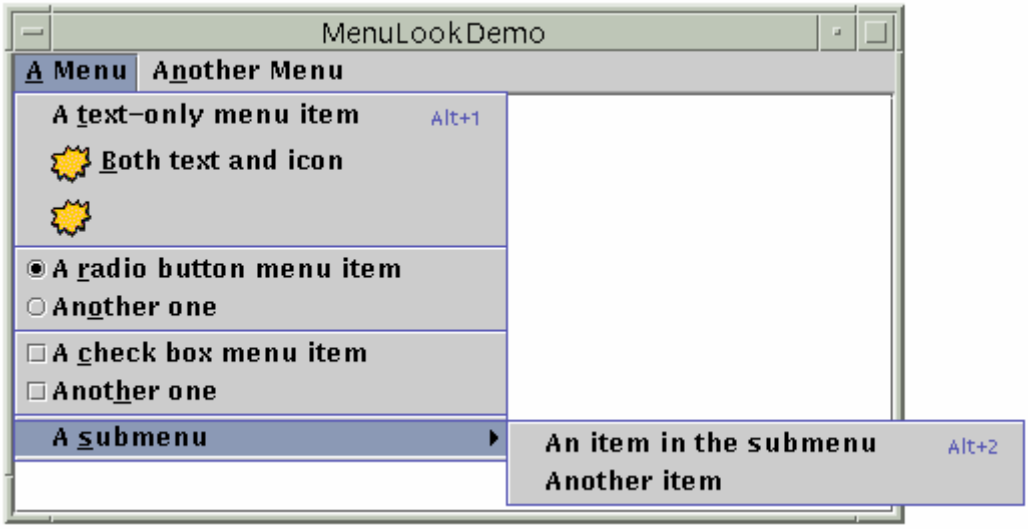
يك فريم داخلي به يك ظرف اضافه مي شود: اگر فريم هاي داخلي رابه داخل يك ظرف مثل JDesktop اضافه نكنيد آنها به نمايش در نمي آيند.
بمنظور ظاهر شدن بايد متد مخصوص اين كار را فراخواني كنيد: شما بايد يكي از دو متد setVisible(true) يا show() را روي فريم داخلي فراخواني كنيد.

زماني كه در داخل يك desktop فريم هاي داخلي زيادي موجود است به نظر مي رسد كه انتقال بين هر كدام از آنها ممكن است كه كند باشد. انتقال به شكل Outline يك از راه ها براي اجتناب از اين امر است. تا زماني كه كه وضعيت جديد در كشيدن (dragging) دكمه موس به وجود نيامده فريم داخلي دوباره سازي نمي شود و به محض ثابت شدن دكمه موس محتويات فريم دوباره سازي مي شود. اگر از اين نوع انتقال استفاده نكنيم تا هر زمان كه دكمه موس حركت مي كند كار دوباره سازي انجام مي شود و اين كار فريمهي داخلي را بسيار كند مي كند.
تنظيم مدل كشيدن به كمك متد setDragMode است و در زير يك نمونه نشان داده شده.

```
desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
```

منوها

با استفاده از منو كاربر مي تواند گزينه هاي زيادي را براي انتخاب داشته و فضاي بسيار كمتري نسبت به ساير روشها اشغال مي كند. يك منو معمولآ در يك نگه دارنده منو يا menu bar ظاهر مي شود. يك menu bar شامل يك يا چند منو بوده و داراي مكان مستقل از زيرساخت در بالاي يك پنجره است، اين به معني است كه اگر در يك سيستمي مكان منو در جاي مشخص بطور مثال سمت راست پنجره در برنامه شما مكان منو همان مكاني است كه شما تعين كرده ايد. در داخل هر منو يك سري قلم (item) از گزينه ها وجود دارد و هر كدام از آنها مي توانند يك ميانبر براي دسترسي داشته باشند. علاوه بر ميانبر اقلام يك منو مي توانند شامل checkbox، radio button، icon، و زير منو باشند. شكل زير يك نمونه از اين منوها است.



بطور مثال در کد زیر نگه دارنده منو شامل يك منو به نام Document دارد. اقلام در این منو new و quit هستند.

```
JMenuBar menuBar = new JMenuBar();
        JMenu menu = new JMenu("Document");
        menuBar.add(menu);
       JMenuItem menuItem = new JMenuItem("New");
        menuItem.setActionCommand("new");
        menu.add(menuItem);
        menuItem = new JMenuItem("Quit");
        menu.add(menuItem);
```

براي تعريف ميانبر به منوي مورد نظر بايد از متد setMnemonic() استفاده كرد. بطور مثال setMnemonic(KeyEvent.VK_N) كليد Alt-n را بعنوان ميانبر تعريف مي كند.

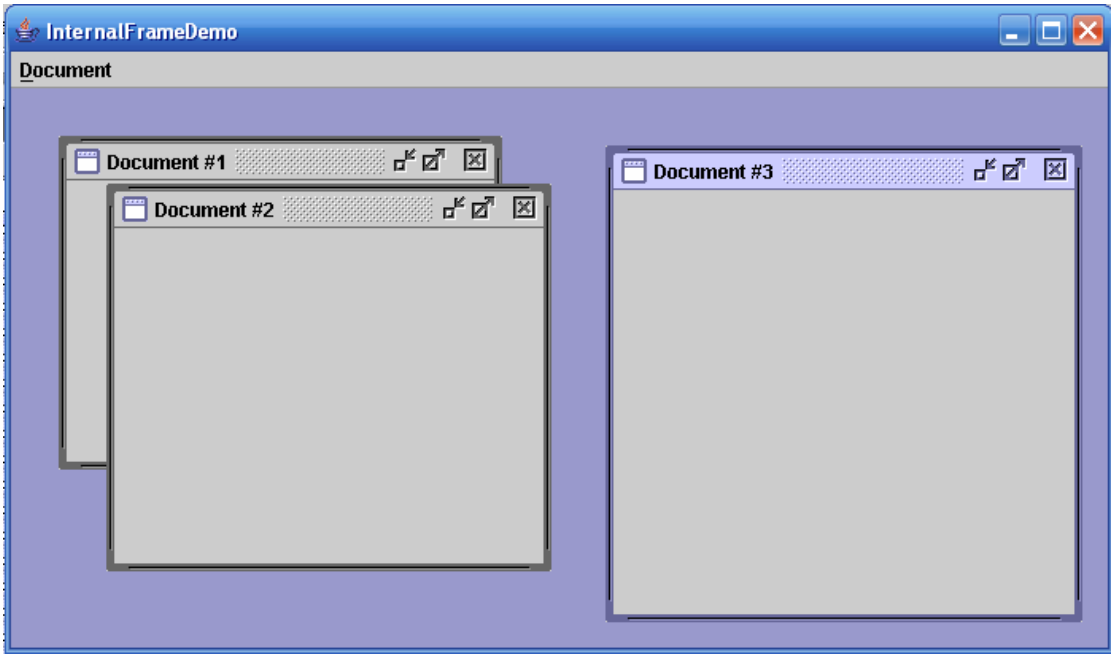
حال كد كامل برنامه را در ادامه ملاحظه مي كنيد.

```
import javax.swing.JInternalFrame;
import javax.swing.JDesktopPane;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JMenuBar;
import javax.swing.JFrame;
import javax.swing.KeyStroke;
import java.awt.event.*;
import java.awt.*;

public class InternalFrameDemo extends JFrame
                             implements ActionListener {
    JDesktopPane desktop;

    public InternalFrameDemo() {
        super("InternalFrameDemo");
```

```
//Make the big window be indented 50 pixels from each edgeof the screen.
    int inset = 50;
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setBounds(inset, inset, screenSize.width  - inset*2,
              screenSize.height - inset*2);

    //Set up the GUI.
    desktop = new JDesktopPane(); //a specialized layered pane
    createFrame(); //create first "window"
    setContentPane(desktop);
    setJMenuBar(createMenuBar());
    //Make dragging a little faster but perhaps uglier.
    desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
}

protected JMenuBar createMenuBar() {
    JMenuBar menuBar = new JMenuBar();
    //Set up the lone menu.
    JMenu menu = new JMenu("Document");
    menu.setMnemonic(KeyEvent.VK_D);
    menuBar.add(menu);
    //Set up the first menu item.
    JMenuItem menuItem = new JMenuItem("New");
    menuItem.setMnemonic(KeyEvent.VK_N);
    menuItem.setAccelerator(KeyStroke.getKeyStroke(
            KeyEvent.VK_N, ActionEvent.ALT_MASK));
    menuItem.setActionCommand("new");
    menuItem.addActionListener(this);
    menu.add(menuItem);

    //Set up the second menu item.
    menuItem = new JMenuItem("Quit");
    menuItem.setMnemonic(KeyEvent.VK_Q);
    menuItem.setAccelerator(KeyStroke.getKeyStroke(
            KeyEvent.VK_Q, ActionEvent.ALT_MASK));
    menuItem.setActionCommand("quit");
    menuItem.addActionListener(this);
    menu.add(menuItem);

    return menuBar;
}
//React to menu selections.
public void actionPerformed(ActionEvent e) {
    if ("new".equals(e.getActionCommand())) { //new
        createFrame();
    } else { //quit
        quit();
    }
```

```
    }

    //Create a new internal frame.
    protected void createFrame() {
        MyInternalFrame frame = new MyInternalFrame();
        frame.setVisible(true); //necessary as of 1.3
        desktop.add(frame);
        try {
            frame.setSelected(true);
        } catch (java.beans.PropertyVetoException e) {}
    }
    //Quit the application.
    protected void quit() {
        System.exit(0);
    }
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        //Create and set up the window.
        InternalFrameDemo frame = new InternalFrameDemo();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Display the window.
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

پس از اجرا برنامه به شکل زیر خواهد بود.



استفاده از برگه ها (Tab)

براي ايجاد برگه ها از كلاس JTabbedPane استفاده مي كنيم و با استفاده از برگه ها مي توان چندين جز (components) گرافيكي را در داخل يك فضاي مشترك قرار داد. در داخل هر برگه اجزاء گرافيك كذاشته مي شود و براي ايجاد هر برگه از متد addTab() استفاده مي شود.

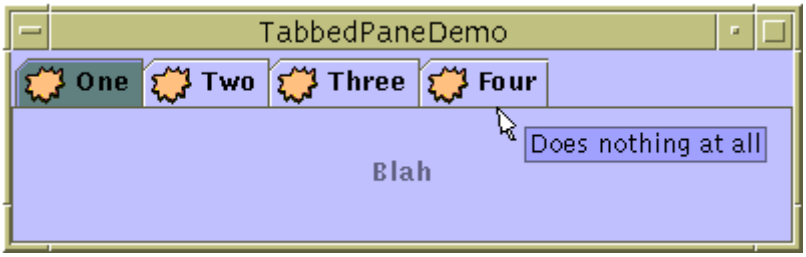
در اين مثال چهار برگه به همراه tool tip و يك ميانبر براي هر كدام ايجاد كرده ايم. در هنگام نمايش برگه اي كه رنگ عنوان آن با ساير برگها تمايز دارد به عنوان برگه كاري محسوب شده و براي تغيير و رفتن به ديگر برگه ها از متد setTabPlacement() استفاده مي كنيم. در مثال توجه داشته باشيد كه نوشتن كدهاي دستگير رخداد لازم نمي باشد و شي JTabbedPane به طور خودكار رخداد هاي صفحه كليد و موس را كنترل مي كند.
متد addTab() هركدام از برگه هاي tabbedPane را با آرگومانهاي عنوان، icon، نام و tool tip ايجاد مي كند. البته هر دو آرگومان icon و متن مي توانند مقدار null داشته باشند. روش ديگر براي ايجاد برگه استفاده از متد insertTab() است. متد setMnemoniAt() ميانبرها را براي هر كدام از برگه ها تعريف كرده و ميانبر همه برگه ها از Alt به همراه شماره برگه استفاده مي كنند. توجه داشته باشيد اندازه همه برگه ها يكسان است و برابر با بزرگترين اندازه برگه در ميان برگه ها است.
كد كامل اين مثال در زير نمايش داده شده.

```
/* TabbedPaneDemo.java is a 1.4 example that requires one additional file:
 *   images/middle.gif.*/
import javax.swing.JTabbedPane;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

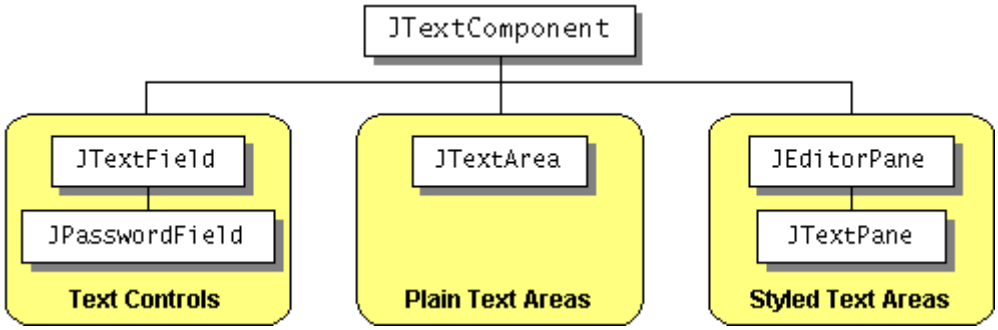
```
import javax.swing.JFrame;
import javax.swing.JComponent;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.KeyEvent;

public class TabbedPaneDemo extends JPanel {
    public TabbedPaneDemo() {
        super(new GridLayout(1, 1));

        JTabbedPane tabbedPane = new JTabbedPane();
        ImageIcon icon = createImageIcon("images/middle.gif");

        JComponent panel1 = makeTextPanel("Panel #1");
        tabbedPane.addTab("Tab 1", icon, panel1,
                          "Does nothing");
        tabbedPane.setMnemonicAt(0, KeyEvent.VK_1);

        JComponent panel2 = makeTextPanel("Panel #2");
        tabbedPane.addTab("Tab 2", icon, panel2,
                          "Does twice as much nothing");
        tabbedPane.setMnemonicAt(1, KeyEvent.VK_2);

        JComponent panel3 = makeTextPanel("Panel #3");
        tabbedPane.addTab("Tab 3", icon, panel3,
                          "Still does nothing");
        tabbedPane.setMnemonicAt(2, KeyEvent.VK_3);

        JComponent panel4 = makeTextPanel(
                "Panel #4 (has a preferred size of 410 x 50).");
          panel4.setPreferredSize(new Dimension(410, 50));
        tabbedPane.addTab("Tab 4", icon, panel4,
                          "Does nothing at all");
        tabbedPane.setMnemonicAt(3, KeyEvent.VK_4);

        //Add the tabbed pane to this panel.
        add(tabbedPane);

        //Uncomment the following line to use scrolling tabs.
        //tabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL_TAB_LAYOUT);
    }
    protected JComponent makeTextPanel(String text) {
        JPanel panel = new JPanel(false);
        JLabel filler = new JLabel(text);
        filler.setHorizontalAlignment(JLabel.CENTER);
        panel.setLayout(new GridLayout(1, 1));
        panel.add(filler);
```

```
        return panel;
    }

    /** Returns an ImageIcon, or null if the path was invalid. */
    protected static ImageIcon createImageIcon(String path) {
        java.net.URL imgURL = TabbedPaneDemo.class.getResource(path);
        if (imgURL != null) {
            return new ImageIcon(imgURL);
        } else {
            System.err.println("Couldn't find file: " + path);
            return null;
        }
    }
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("TabbedPaneDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Create and set up the content pane.
        JComponent newContentPane = new TabbedPaneDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.getContentPane().add(new TabbedPaneDemo(),
                                   BorderLayout.CENTER);
        //Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

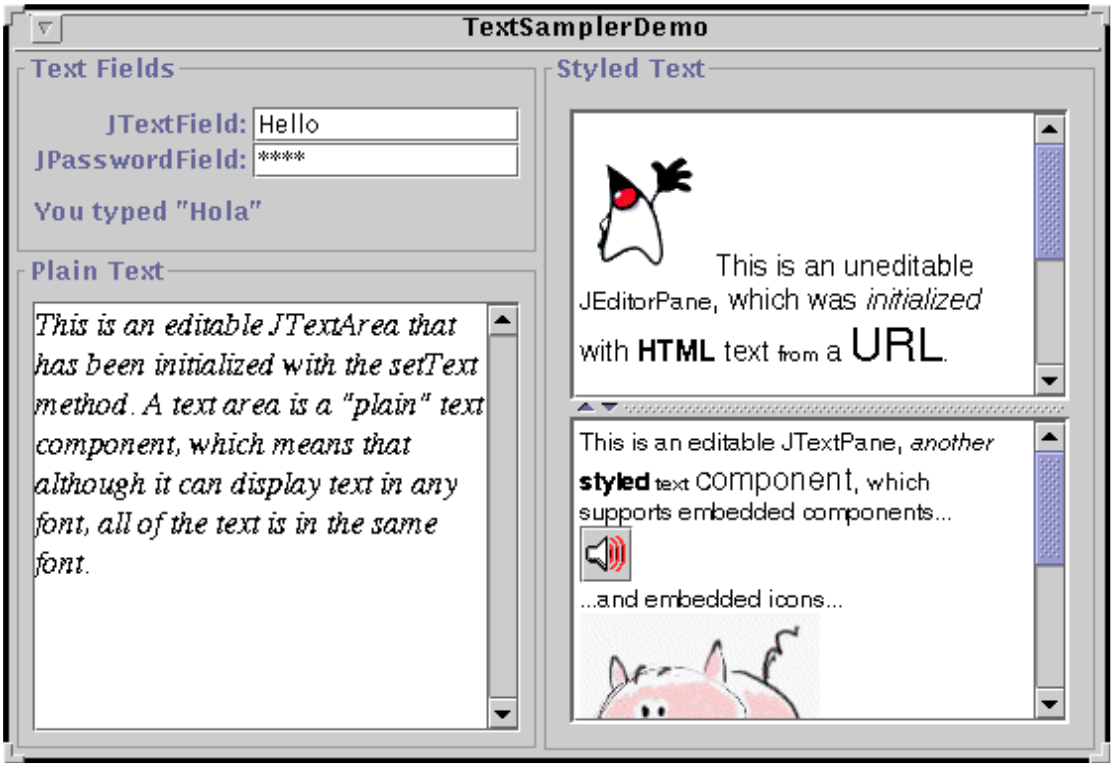
پس از اجرا نتيجه خروجي به شكل زير خواهد بود.



استفاده از كامپونتهاي متن
كامپونتهاي متني براي نمايش و ويرايش متنها كه در برنامه ها لازم هستند ايجاد شده اند و تعداد اين كامپونتها شش عدد مي باشد. همه اين كامپونتها از كلاس JTextComponent به ارث برده شده اند و مي توان آنها را به سه دسته عمده تقسيم كرد. اين سلسله مراتب در شكل زير نشان داده شده.



در تصوير زير نيز يك برنامه كه از هر شش نوع كامپونت متني استفاده كرده به همراه توضيحات نشان داده شده.



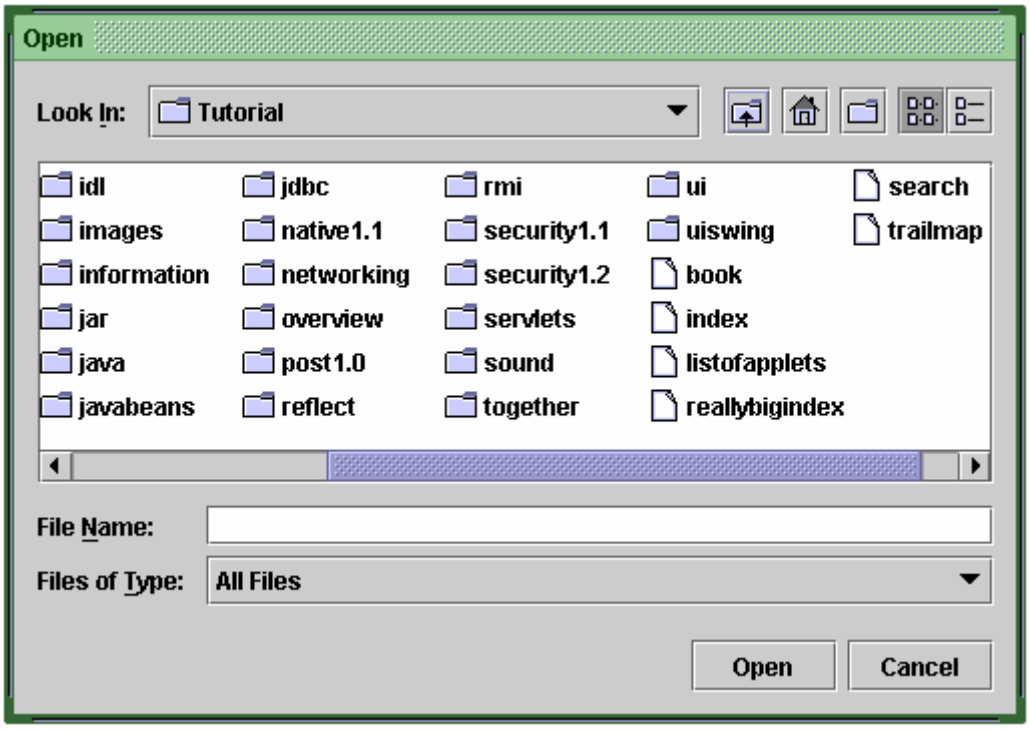
در دسته اول كامپونتها يعني Text Controls ما فقط مي توانيم يك خط را نمايش داده و ويرايش كنيم، شبيه به دكمه ها. استفاده از آنها متني كمتري را نمايش داده و زماني كه متن به اندازه طول رسيد دستگير رخداد آن فراخواني مي شود. اين دسته شامل سه كلاس JFormattedTextField، JPasswordField و JTextField است.

دسته دوم Plain Text Areas است كه در آن مي توان متنهاي بيش از چند خط را نمايش داد و ويرايش كرد. اگرچه text area متن را با هر font نمايش مي دهد اما توجه داشته باشيد كه تمام متن بايد با آن font نشان داده شوند. اين دسته فقط

شامل يك كلاس JTextArea است كه به كاربر اين امكان را مي دهد كه متن فرمت نشده نامحدودي را وارد يا تماشا كند.
دسته سوم Styled Text Areas هستند. در اين دسته مي توان متن را در بيش از يك font نمايش داد و ويرايش كرد. برخي از اين كامپونتها اين اجازه را مي دهند كه تصاوير و ساير كامپونتها را به همراه متن نمايش داد. به دليل قدرت و سازگاري اين كامپونتها براي برنامه ها با متنهاي طرح دار مثل راهنما ها (help) بسيار مفيد هستند. اين دسته شامل دو كلاس JEditorPane و JTextPane به همراه زير كلاسهايشان است.

مثال انتخاب كردن فايلها

انتخاب كنند فايل يك رابط كاربر مناسب بمنظور ارتباط با سيستم فايلها و انتخاب يك فايل يا فهرست مي باشد. براي نمايش انتخاب كننده فايل شما بايد از API مخصوص اين كار يعني JFileChooser كنيد. روش ديگر استفاده از انتخاب كننده فايل قرار دادن نمونه شئ JFileChooser در داخل يك ظرف است. برنامه شما مسئول انجام كار بر روي فايل انتخاب شده است، همانند باز كردن و ذخيره كردن آن فايل. تعين استاندارد جعبه انتخاب كننده فايل به كمك Look&Feel مشخص مي شود. در جاوا Look&Feel جعبه باز كردن فايل وذخيره فايل يكسان مي باشد و فقط در عنوان متن ظاهر شونده در بالاي جعبه و دكمه متمايز مي باشند. در زير تصوير يك جعبه باز كردن فايل با Look&Feel استاندارد را مشاهده مي كنيد.



گرفتن يك جعبه باز كننده فايل نياز به دو خط كد دارد، همانند زير:

```
//Create a file chooser
final JFileChooser fc = new JFileChooser();
...
//In response to a button click:
int returnVal = fc.showOpenDialog(aComponent);
```

آرگومان ورودي به متد showOpenDialog يك كامپوننت گرافيكي والد را براي جعبه باز كردن فايل مشخص مي كند. يك كامپوننت والد فريمي كه جعبه باز كردن فايل در داخل آن مي باشد و وضعيت قرار گرفتن جعبه را مشخص مي كند. بطور استاندارد جعبه در روي كامپوننت والد مي باشد. زماني كه پنجره فريم والد Minmize باشد، اين جعبه نيز نشان داده نخواهد شد و به آن وابسته خواهد بود.
بطور پيش فرز يك انتخاب كننده فايل تمام فايلها در فهرست جاري كاربر (home directory) را نشان مي دهد. شما مي توانيد فهرست اوليه انتخاب كننده فايل را را تغير دهيد. براي اين كار از متد setCurrentDirectory() استفاده مي كنيم.
زماني كه دكمه Open از انتخاب كننده فايل فشرده مي شود متد actionPerformed() از شنود كننده اعمال فراخواني مي شود، همانند زير :

```
public void actionPerformed(ActionEvent e) {
    //Handle open button action.
    if (e.getSource() == openButton) {
        int returnVal = fc.showOpenDialog(FileChooserDemo.this);

        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            //This is where a real application would open the file.
            log.append("Opening: " + file.getName() + "." + newline);
        } else {
            log.append("Open command cancelled by user." + newline);
        }
    } ...
}
```

متدهاي showXxxDialog() يك مقدار عددي صحيح را كه به فايل انتخاب شده كاربر اشاره دارد بر مي گردانند. اين عدد كافي است براي چك كردن اينكه آيا مقدار برگردانده شده با مقدار APPROVE_OPTION (تائيد با زدن دكمه Open) برابر است يا نه و اگر مقدار ديگري باشد (زدن Cancel يا دكمه ديگر) كاري انجام نخواهد شد.
بمنظور گرفتن فايل انتخاب شده (يا فهرست انتخاب شده در انتخاب كردن فهرست با همان كلاس) از متد getSelectedFile() از JFileChooser استفاده مي كنيم. خروجي اين متد يك شئ نمونه از كلاس Fileاست.اين مثال نام يك فايل را گرفته و در پيغام log آن را نشان مي دهد. به كمك متدهاي getPath()، isDirectory() و exists() از شئ File اطلاعاتي درباره فايل گرفته و همچنين ساير متدها مثل delete() و rename() براي تغير در فايل استفاده كنيد. بمنظور خوانده و نوشتن از فايل از كلاسهاي FileReader و FileWriter استفاده كنيد. يك مثال كوچك در اين باره در زير به نمايش درآمده.

```
import java.io.*;

public class Copy {
    public static void main(String[] args) throws IOException {
            File inputFile = new File("farrago.txt");
            File outputFile = new File("outagain.txt");

        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
```

```
        int c;

        while ((c = in.read()) != -1)
          out.write(c);

        in.close();
        out.close();
    }
}
```

این مثال محتواي فايل farrago.txt را به داخل فايل outagain.txt كپي مي كند. در زير كد كامل برنامه را مشاهده مي كنيد.

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.*;

public class FileChooserDemo extends JPanel
                             implements ActionListener {
    static private final String newline = "\n";
    JButton openButton, saveButton;
    JTextArea log;
    JFileChooser fc;

    public FileChooserDemo() {
        super(new BorderLayout());
        //Create the log first, because the action listeners need to refer to it.
        log = new JTextArea(5,20);
        log.setMargin(new Insets(5,5,5,5));
        log.setEditable(false);
        JScrollPane logScrollPane = new JScrollPane(log);

        //Create a file chooser
        fc = new JFileChooser();
        //Create the open button.  We use the image from the JLF
        //Graphics Repository (but we extracted it from the jar).
        openButton = new JButton("Open a File...",
                                 createImageIcon("images/Open16.gif"));
        openButton.addActionListener(this);

        //Create the save button.  We use the image from the JLF
        //Graphics Repository (but we extracted it from the jar).
        saveButton = new JButton("Save a File...",
                                 createImageIcon("images/Save16.gif"));
        saveButton.addActionListener(this);

        //For layout purposes, put the buttons in a separate panel
```

```
    JPanel buttonPanel = new JPanel(); //use FlowLayout
    buttonPanel.add(openButton);
    buttonPanel.add(saveButton);

    //Add the buttons and the log to this panel.
    add(buttonPanel, BorderLayout.PAGE_START);
    add(logScrollPane, BorderLayout.CENTER);
}

public void actionPerformed(ActionEvent e) {

    //Handle open button action.
    if (e.getSource() == openButton) {
        int returnVal = fc.showOpenDialog(FileChooserDemo.this);

        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            //This is where a real application would open the file.
            log.append("Opening: " + file.getName() + "." + newline);
        } else {
            log.append("Open command cancelled by user." + newline);
        }
        log.setCaretPosition(log.getDocument().getLength());

    //Handle save button action.
    } else if (e.getSource() == saveButton) {
        int returnVal = fc.showSaveDialog(FileChooserDemo.this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            //This is where a real application would save the file.
            log.append("Saving: " + file.getName() + "." + newline);
        } else {
            log.append("Save command cancelled by user." + newline);
        }
        log.setCaretPosition(log.getDocument().getLength());
    }
}

/** Returns an ImageIcon, or null if the path was invalid. */
protected static ImageIcon createImageIcon(String path) {
    java.net.URL imgURL = FileChooserDemo.class.getResource(path);
    if (imgURL != null) {
        return new ImageIcon(imgURL);
    } else {
        System.err.println("Couldn't find file: " + path);
        return null;
    }
}
```

```
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);
        JDialog.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("FileChooserDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Create and set up the content pane.
        JComponent newContentPane = new FileChooserDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

بطور پيش فرز انتخاب كننده فايل فقط فايلها را انتخاب مي كند. شما به كمك متد setFileSelectionmode() اين را تغير داده و به انتخاب كننده هر دو فايل و فهرست يا فقط فهرست تبديل كنيد. نمونه اين كار را در زير ملاحظه مي كنيد.

```
fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
fc.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
```

همان طور كه در كد برنامه ملاحظه مي كنيد براي ذخيره فايل از همان كلاس JFileChooser بمنظور ايجاد جعبه ذخير استاندارد استفاده مي شود. استفاده از يك انتخاب كننده فايل يعني fc براي نشان دادن باز كردن و ذخيره فايل همان طور كه در زير ملاحظه مي كنيد دو مزيت دارد. مزيت اول اينكه فهرست جاري باز كننده فايل و ذخيره كننده فايل يكي مي باشند و دوم اينكه شما براي هر دو باز كننده و ذخيره كننده فايل يك انتخاب كننده فايل را تنظيم دلخواه كرده ايد.
پس از اجراي برنامه فرم خروجي به شكل زير خواهد بود.

ايجاد ميله ابزار

بمنظور ايجاد يك ميله ابزار از كلاس JToolBar استفاده مي كنيم. اين كلاس يك ظرف براي نگهداري كامپونتهاي گروهي مثل دكمه ها و icon ها در داخل يك سطر يا ستون است. همانند منوها ميله ابزار يك راه براي دستيابي راحتر به گزينه ها است. بطور پيش فرز كاربر مي تواند يك ميله ابزار را به مكان ديگري با كشيدن (drag) در فريم انتقال دهد. همچنين اين قابليت وجود دارد كه يك ميله ابزار خارج از ظف خود مثل يك فريم قرار بگيرد. كدهاي زير يك ميله ابزار را ايجاد كرده و آن را به ظرف اضافه مي كند.

```
public class ToolBarDemo extends JPanel
                        implements ActionListener {
    ...
    public ToolBarDemo() {
        super(new BorderLayout());
        ...
        JToolBar toolBar = new JToolBar("Still draggable");
        addButtons(toolBar);
        ...
        setPreferredSize(new Dimension(450, 130));
        add(toolBar, BorderLayout.PAGE_START);
        add(scrollPane, BorderLayout.CENTER);
    }
    ...
}
```

وضعيت مكاني ميله ابزار بهمراه ميله لغزان هر دو در پانل به كمك طرح بند لبه (border layout) مشخص شده. مكان ميله ابزار در شروع صفحه (PAGE_START) و مكان ميله لغزنده در مركز (CENTER) مي باشد. به دليل اينكه ميله لغزان در در مركز قرار دارد و كامپوننت گرافيكي ديگري به غير از ميله ابزار در ظرف نمي باشد، ميله ابزار مي تواند به ساير لبه هاي ظرف كشيده شود مثلآ در سمت راست فريم قرار گيرد. همچنين مي توان آنرا به خارج از ظرف كشيد بطوري كه در يك پنجره جدا نمايش داده مي شود و عنوان آن پنجره "Still draggable" مي باشد.

ايجاد دكمه هاي ميله ابزار

دكمه ها در داخل ميله ابزار از نوع JButton بوده و درداخل آنها از تصوير استفاده مي شود. هر تصوير در ابعاد 16در 16 و 24در24 است. اين به اين دليل است كه Look&Feel پيش فرز از آن استفاده مي كند. در زير نحوه اضافه كردن يك دكمه به يك ميله ابزار نشان داده شده.

```
        //third button
        button = new JButton(new ImageIcon("images/right.gif"));
        button.setToolTipText("This is the right button");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayResult("Action for third button");
            }
        });
        toolBar.add(button);
```

کد کامل برنامه را در زیر نشان داده شده.

```
import javax.swing.JToolBar;
import javax.swing.JButton;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.*;

public class ToolBarDemo extends JFrame {
    protected JTextArea textArea;
    protected String newline = "\n";

    public ToolBarDemo() {
        //Do frame stuff.
        super("ToolBarDemo");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        //Create th toolbar.
        JToolBar toolBar = new JToolBar();
        addButtons(toolBar);

        //Create the text area used for output.
        textArea = new JTextArea(5, 30);
        JScrollPane scrollPane = new JScrollPane(textArea);

        //Lay out the content pane.
        JPanel contentPane = new JPanel();
        contentPane.setLayout(new BorderLayout());
        contentPane.setPreferredSize(new Dimension(400, 100));
```

```
    contentPane.add(toolBar, BorderLayout.NORTH);
    contentPane.add(scrollPane, BorderLayout.CENTER);
    setContentPane(contentPane);
}

protected void addButtons(JToolBar toolBar) {
    JButton button = null;

    //first button
    button = new JButton(new ImageIcon("images/left.gif"));
    button.setToolTipText("This is the left button");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayResult("Action for first button");
        }
    });
    toolBar.add(button);

    //second button
    button = new JButton(new ImageIcon("images/middle.gif"));
    button.setToolTipText("This is the middle button");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayResult("Action for second button");
        }
    });
    toolBar.add(button);

    //third button
    button = new JButton(new ImageIcon("images/right.gif"));
    button.setToolTipText("This is the right button");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            displayResult("Action for third button");
        }
    });
    toolBar.add(button);
}

protected void displayResult(String actionDescription) {
    textArea.append(actionDescription + newline);
}

public static void main(String[] args) {
    ToolBarDemo frame = new ToolBarDemo();
    frame.pack();
    frame.setVisible(true);
}
```

}

پس از اجرا میله ابزار به شکلهاي زير خواهد بود. در اولين شكل ميله ابزار به سمت راست كشيده شده و در دومين شكل ميله ابزار به بيرون از ظرف خود كشيده شده.

فصل چهارم

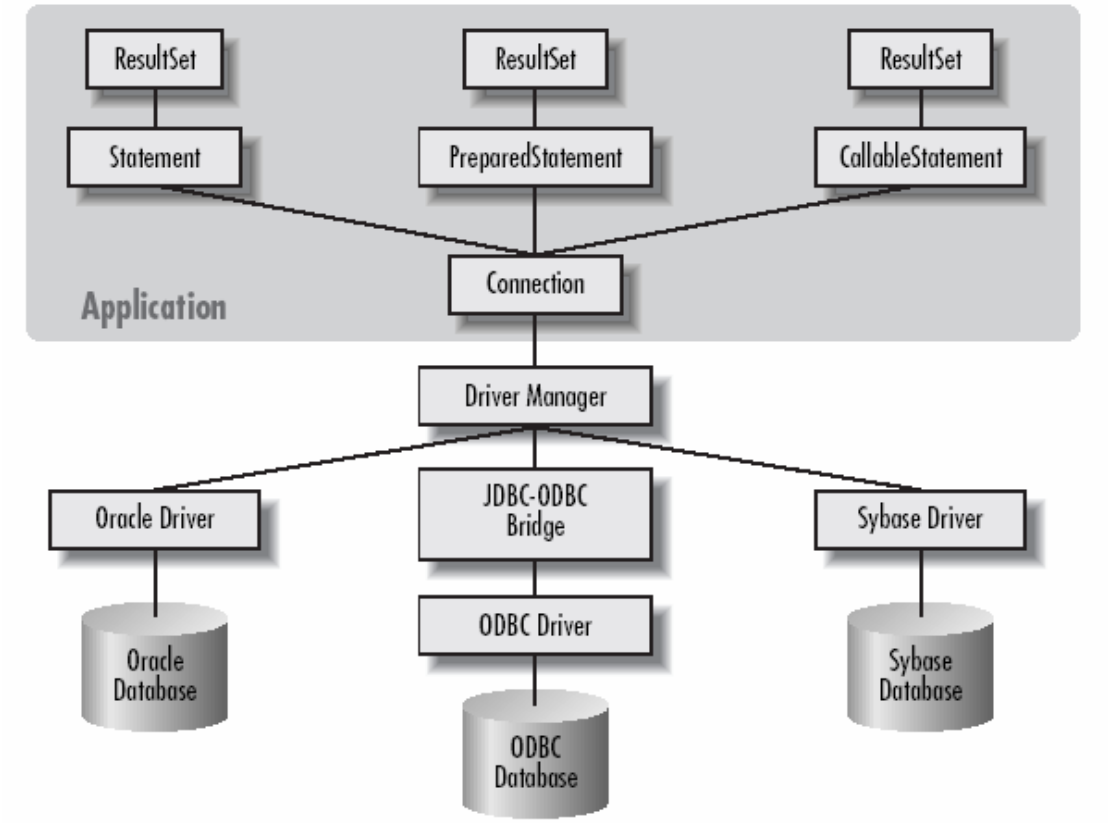
JDBC(Java Data Base Connectivity)

پايگاه داده ربطه اي

يك جدول در پايگاه داده رابطه اي شامل اطلاعاتي در مورد يك موجوديت كه كه كمك يك كليد با مقدار يكتا به سطرهاي آن جدول دسترسي پيدا ميكنيم . اطلاعات داخل جداول ميتوانند به كمك دستورات SQL حذف ، اضافه ، وبه هنگام سازي شوند .
JDBC API كه براي اولين بار در JDK 1.1 براي برقراي ارتباط به پايگاه داده و اجراري دستورا ت SQL استاندارد ارائه شد

JDBC API

JDBC يك سري API مه به شما اجازه اجراي دستورات SQL و گرفتن نتايج از آنها را ميدهد. خود اين API ها مجموعه اي از رابطها وكلاسهاي طراحي شده براي اجراي اعمال روي هر گونه از پايگاه داده رابطهاي موجود است . شكل زير نحوه تعامل JDBC با بانك اطلاعاتي ODBC Connector, Sybase , oracle رانشان ميدهد.



JDBC Drivers

JDBC API در پكيج java.sql و شامل تعدادي كلاسهاي به هم مرتبط است. بيشتر اين كلاسها بصورت رابط (interface) هستند و تكميلي آنها ارائه نشده .
هركدام از بانكهاي اطلاعاتي به كمك يك JDBC Driver ويژه كه در رابط java.sql.Driver ميباشد مورد دستيابي قرار مي گيرند . اين درايورها براي اكثر بانكهاي اطلاعاتي معمول به صورت رايگان موجود هستند . شركت سان JDBC-ODBC به عنوان پل در JDK براي اتصال به منابع داده ODBC استاندارد مثل Microsoft Access ارائه داده.

برقراري ارتباط

اولين قدم استفاده از ODBC Driver برا ي گرفتن اتصال به پايگاه داده است و لود كردن كلاس درايور به ماشين مجازي كه برنامه در آن اجرا ميشود. اين لود كردن درايور را بعداً براي ما همگامي كه مي خواهيم متصل شويم قابل دسترس ميكند .
يك روش راحت براي لود كردن كلاس درايور استفاده از متد Class.forName() است.
براي مثال :

```
Class.forName("sun.jdbc.odbc.JdbcodbcDriver");
```

وقتي كه درايور به حافظه لود ميشود خود را با كلاس java.sql.DriverManager رجيستر مي كند.
گام بعدي سوال كردن از كلاس DriverManager براي باز كردن اتصال به بانك اطلاعاتي است . اتصال به كمك تابع getConnection() از DriverManager كه ورودي آن يك رشته به فرم URL براي اطلاعات لازم بانك اطلاعاتي است .
متد getConnection() يك كلاس ازنوع java.sql.Connection برمي گرداند

```
Connection con=
```

DriverManager.getConnection("jdbc:odbc:somedb","user","password");

يك JDBC URL براي شناساي يك بانك اطلاعاتي است و درايور هاي ويژه نياز به اطلاعات جدا در URL خود دارند.معمولآ JDBC URL با jdbc:subprotoocol:subname شروع مي شوند. براي مثال JDBC Thin Driver براي ORACLE ازURL به فرم jdbc:orcle:thin:@dbhost:port:sid; استفاده مي كند.
براي پل به ODBC شكل كلي URL به فرم jdbc:odbc:datasourcename;odbcoptions است.
در طي زمان اجراي getConnection() كلاس DriverManager از درايور در مورد قبول يا شناساي URL سوال مي كند و اگر جواب بله باشد مدير درايور از آن درايور براي ايجاد Connectionاستفاده مي كند .

```
Connection con = null;

try {
// Load (and therefore register) the JDBC-ODBC Bridge
// Might throw a ClassNotFoundException

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
// Get a connection to the database
// Might throw an SQLException

con = DriverManager.getConnection("jdbc:odbc:somedb", "user", "passwd");
// The rest of the code goes here.
}
catch (ClassNotFoundException e) {

// Handle an error loading the driver
}

catch (SQLException e) {
// Handle an error getting the connection
}
finally {
// Close the Connection to release the database resources immediately.
try {
if (con != null) con.close();
}
catch (SQLException ignored) { }
}
```

اجراي پرس وجو ازبانك اطلاعاتي

استفاده از بانك اطلاعاتي نيازمند نوشتن Queries و راه ساده براي اجراي Query استفاده از كلاس java.sql.Statement است. شي Statement هيچ وقت به طور مستقيم ايجاد نمي شود بلكه برنامه با فراخواني متد create Statement() از شي Connection يك شي Statement ايجاد مي كند.

```
Statement stmt = con.createStatement();
```

وقتي يك Query وقتي ايجاد شد براي گرفتن داده بايد متد executeQuery() از شي Statement فراخواني كرد. اين متد دستور را اجرا كرده و در داخل كلاس jav.sql.ResultSet مي ريزد

```
ResultSet rs = stmt.executeQuery("SELECT * FROM CUSTOMERS");
```

اينگونه ميتوان فرض كرد كه شي ResultSet رديف يا سطرهاي از داده ها در يك زمان از جدول بانك است . براي رفتن به سطر بعدي در ميان سطرها با متد next() از Statement را اجرا كرده و براي گرفتن فيلدهاي سطر بسته به نوع داده از متد سازگار با آن استفاده ميكنيم. متد هاي getString() و getObject بشيتر از ساير متدها معمولاً استفاده مي شوند.

```
while(rs.next()) {

String event = rs.getString("event");

Object count = (Integer) rs.getObject("count");
}
```

همان طور كه ميدانيد ResultSet به كلاس والد خود يعني Statement رابطه دارد و اگر شي Statement بسته شود يعني دستور ديگري اجرا شود به طور اتوماتيك شي ResultSet بسته ميشود .
مثال زير نشان دهنده اتصال به بانك اطلاعاتي Oracle با JDBC Driver و اجراي پرس جوي ساده براي چاپ نام و شماره تلفن باز جدول كارمندان است. فرض كنيد بانك اطلاعاتي داراي جدولي به نام EMPLOYEES با دو فيلد NAME , PHONE است.

```
import java.io.*;
import java.sql.*;

public class GetEmployees {
public static void main(String args[]) {

Connection con = null;
Statement stmt = null;
ResultSet rs = null;
try {

// Load (and therefore register) the Oracle Driver

Class.forName("oracle.jdbc.driver.OracleDriver");

// Get a Connection to the database
con = DriverManager.getConnection(
"jdbc:oracle:thin:dbhost:1528:ORCL", "user", "passwd");

// Create a Statement object
```

```
stmt = con.createStatement();

// Execute an SQL query, get a ResultSet
rs = stmt.executeQuery("SELECT NAME, PHONE FROM EMPLOYEES");
// Display the result set as a list

while(rs.next()) {
System.out.println( rs.getString("name") + "  " + rs.getString("phone"));
}
}
catch(ClassNotFoundException e) {

System.out.println("Couldn't load database driver: " + e.getMessage());
}
catch(SQLException e) {

System.out.println("SQLException caught: " + e.getMessage());
}
finally {
// Always close the database connection.
try {
if (con != null) con.close();
}
catch (SQLException ignored) { }
}
}
}
```

Handle یا دستگیر کردن استثناء های SQL

در کد های بالا با دو بلوك Catch دو استثنا ClassNotFoundException و SQLException را دستگیر(handle) میکنیم .اولي از متد Class.forName وقتي كه درايور JDBCنتواند لود شود و دومي وقتي كه خطاي از هركدام از متدهاي JDBC بر بزند.

SQLException همانند ديگر استثناءها با يك خصوصيت اضافي وآن اينكه ميتواند زنجيره سازي شود .SQLException يك متد به نام getNextException() داردو اجازه ميدهد استثناءهاي اضافي رانيز دستگير كنيم . براي كامل كردن كدقبلي ميتوانيد كد زير را به catche از SQLException اضافه كنيد.

```
catch (SQLException e) {

out.println(e.getMessage());

while((e = e.getNextException()) != null) {

System.out.println(e.getMessage());
}
}
```

این کد ابتدا استثناء اول را نشان داده سپس بقیه استثناءها اگر موجود باشند را نیز نشان میدهد.
برا ي فهميدن تعداد سطر هاي گرفته شده ResultSet ميتوان از ResultSetMetaData به شكل زير استفاده كرد.

```
ResultSetMetaData rsmd = rs.getMetaData();

int numcols = rsmd.getColumnCount();
```

همچنین میتوان با متد getColumnLabel() از ResultSetMetaData نام هر ستون جدول را گرفت. مثال زیر این عمل را نشان میدهد.

```
for (int i = 1; i <= numcols; i++) {

System.out.println("column "+i+"is  :" + rsmd.getColumnLabel(i));
}
```

به منظور دریافت اطلاعات از هر سطر بدون توجه به نام آن از متد getObject() استفاده کرد که بطور خود کار نوع داده ستون را تشخیص داده و معادل با آن را برمیگرداند. با این روش کد نوشته شده شما قابلیت حمل بیشترش خواهد داشت . به كمك getObject() مي توان ستون را به هر نوع حتي به غير از نوع خود دريافت كرد. بطور مثال getObject().toString() .

در جدول زير متدهاي كه مي توانيم به كمك آنها انواع داده در SQL را دريافت نمايم مشخص ميباشد . در ستون اول نام آن داده در SQLاستاندارد و در ستون دوم نوع داده كه بطور خودكار توسط getObject برگردانده ميشود. اگر شما نوع ستونها درجدول SQL خود راميدانيد ميتوانيد بسته به نوع ستون ازمتدهاي سطر سوم استفاده كنيد.

SQL Data Type	Java Type Returned by getObject()	Recommended Alternative to getObject()
CHAR	String	String getString()
VARCHAR	String	String getString()
LONGVARCHAR	String	InputStream getAsciiStream() InputStream getUnicodeStream()
NUMERIC	java.math.BigDecimal	java.math.BigDecimal getBigDecimal()
DECIMAL	java.math.BigDecimal	java.math.BigDecimal getBigDecimal()
BIT	Boolean	boolean getBoolean()
TINYINT	Integer	byte getByte()
SMALLINT	Integer	short getShort()
INTEGER	Integer	int getInt()
BIGINT	Long	long getLong()
REAL	Float	float getFloat()
FLOAT	Double	double getDouble()
DOUBLE	Double	double getDouble()
BINARY	byte[]	byte[] getBytes()
VARBINARY	byte[]	byte[] getBytes()
LONGVARBINARY	byte[]	InputStream getBinaryStream()
DATE	java.sql.Date	java.sql.Date getDate()
TIME	Java.sql.Time	java.sql.Time getTime()
TIMESTAMP	Java.sql.Timestamp	java.sql.Timestamp getTimestamp()

Handle كردن فيلد هاي خالي(null)

Handle كردن فيلد هاي null در JDBC با كمي نيرنگ همراه است (زماني كه محتواي يك فيلد در بانك پوچ يا null است متد گيرنده داده مثل getObject() مقدار null بر مي گرداند) و متد هيچ شيي بر نمي گرداند.
هيچ راهي براي تشخيص null بودن مقدار يك فيلد قبل از خواندن آن وجود ندارد. متد wasNull() در ResultSet با مقدارهاي true , false كه برمي گرداند مشخص ميكند كه آيا مقدار آخرين ستون خوانده شده null مي باشد يا نه . اين به اين معني است كه شما مي بايست ابتدا داده را از ResultSet خوانده و به داخل متغير ريخته و متد wasNull() را فراخواني ميكنيم . در زير اين عمل نشان داده شده است.

```
int age = rs.getInt("age");

  if(!rs.wasNull())

System.out.println("age : "+age);
```

روش ديگر كنترل كردن متد getObject() كه آيا مقدار آن null مي باشد يا نه.

به روز كردن پايگاه داده

در برنامه ها علاوه بر ديدن محتواي جداول نياز به تغيير محتواي آنها نيز داريم . براي اجراي دستورات UPDATE , INSERT , DELETE مي توانيد به غير از ResultSet از متد executeUpdate() از Statement استفاده كنيد. آن تعدادي سطر تغيير داده شده با Statement را بر مي گرداند. بطور مثال :

```
int count =
stmt.executeUpdate("DELETE FROM CUSTOMERS WHERE CUSTOMER_ID =
5");
```

بعد از اجراي دستور SQL كلاسي همانند ResultSet (تعداي سطر) برگرداننده ميشود. به كمك متد execute() از Statement ميتوانيم متوجه شويم كه آيا دستور مورد نظر ما يك يا چند ResultSet() ايجاد كرده يا نه و true به معني ايجاد و false هم به معني عدم ايجاد مي باشد.

```
Boolean b= stmt.execute(sql);
```

متدهاي getResultSet() , getUpdateCount() از Statement براي دسترس به نتايج دستور execute() مي باشند.
در عمل درج در جداول اولين عمل اجراي متد moveToInsertRow() از resultSet مورد نظر است كه امكان درج كردن سطرها را فراهم مي كند. سپس با متدهاي مخصوص هر فيلد كه اول انها با update شروع شده مثل updateString اقدام به ايجاد سطر براي درج ميكنيم . براي درج واقعي دربانك پس از ايجاد سطر درج متد insertRow(); را فراخواني ميكنيم .
از TYPE_SCROLL_SENSITIVE براي نشان دادن(اعمال كردن تغييرات سطر ها در بانك به resultSetمورد استفاده ما) تغيرات دربانك در برنامه كمك ميگيريم.

مثال زير در مورد عمل درج در بانك اطلاعاتي است.

```
import java.sql.*;

 public class InsertRows {

  public static void main(String args[]) {

    String url = "jdbc:mySubprotocol:myDataSource";
    Connection con;
    Statement stmt;
    try {
      Class.forName("myDriver.ClassName");

    } catch(java.lang.ClassNotFoundException e) {
      System.err.print("ClassNotFoundException: ");
      System.err.println(e.getMessage());
    }

    try {
      con = DriverManager.getConnection(url,
            "myLogin", "myPassword");
```

```
stmt = con.createStatement(
        ResultSet.TYPE_SCROLL_SENSITIVE,
        ResultSet.CONCUR_UPDATABLE);

ResultSet uprs = stmt.executeQuery(
          "SELECT * FROM COFFEES");

uprs.moveToInsertRow();

uprs.updateString("COF_NAME", "Kona");
uprs.updateInt("SUP_ID", 150);
uprs.updateFloat("PRICE", 10.99f);
uprs.updateInt("SALES", 0);
uprs.updateInt("TOTAL", 0);

uprs.insertRow();

uprs.updateString("COF_NAME", "Kona_Decaf");
uprs.updateInt("SUP_ID", 150);
uprs.updateFloat("PRICE", 11.99f);
uprs.updateInt("SALES", 0);
uprs.updateInt("TOTAL", 0);

uprs.insertRow();

uprs.beforeFirst();

System.out.println(
  "Table COFFEES after insertion:");
while (uprs.next()) {
  String name = uprs.getString("COF_NAME");
  int id = uprs.getInt("SUP_ID");
  float price = uprs.getFloat("PRICE");
  int sales = uprs.getInt("SALES");
  int total = uprs.getInt("TOTAL");
  System.out.print(
    name + "   " + id + "   " + price);
  System.out.println(
    "   " + sales + "   " + total);
}

uprs.close();
stmt.close();
con.close();

} catch(SQLException ex) {
  System.err.println(
```

```
        "SQLException: " + ex.getMessage());
    }
  }
}
```

در اینجا میتوان کد زیر را نیز هر با در برنامه به کار برد.

```
stmt.executeUpdate("INSERT INTO COFFEES " +
                   "VALUES ('Kona', 150, 10.99, 0, 0)");
```

براي عمل حذف باید از متد deleteRow() وبراي مشخص کردن رکورد خاص(بطور مثال رکورد چهارم) با توجه به ترتیب قرار گیري(معمولآ بر اساس کلید اصلي) با متد absolute(4) برروي آن رفته و عمل حذف را انجام میدهیم.

استفاده از دستورات SQL آماده (Prepared)

شي PreparedStatement شبیه قاعده Statement بوده و در آن مي توانیم دستورات SQL را اجرا کنیم. تفاوت اساسي این است که در PreparedStatement دستور توسط بانك براي سرعت بیشتر پیش کامپایل میشود. زماني که PreparedStatement کامپایل میشود آن میتواند با اشتفاده از پارامتر ها دلخواه تنظیم شود. این قابلیت براي اجراي دستورات کلي SQL در برنامه مفید است.
متدPreparedStatement از شي Connection بوده و ؟ براي ارسال پارامتر در بعدها استفاده میکند.
در کد زیر نحوه استفاده از PreparedStatement نشان داده شده.

```
PreparedStatement pstmt = con.prepareStatement(
"INSERT INTO ORDERS (ORDER_ID, CUSTOMER_ID, TOTAL) VALUES
(?,?,?)");
// Other code
pstmt.clearParameters(); // clear any previous parameter values
pstmt.setInt(1, 2); // set ORDER_ID
pstmt.setInt(2, 4); // set CUSTOMER_ID
pstmt.setDouble(3, 53.43); // set TOTAL
pstmt.executeUpdate(); // execute the stored SQL
```

در نهایت دستور زیر بطور واقعي اجرا میشود.

```
INSERT INTO ORDERS (ORDER_ID, CUSTOMER_ID, TOTAL) VALUES

(2,4,53.43)")
```

متد clearParameters() تمام پارامتر هاي قبلي استفاده شده را حذف میکند. پس از مقدار دهي به فیلدهاي رکورد توسط متدهاي setXXX() به منظور درج واقعي در بانك از متد executeUpdate() استفاده میکنیم.
در درج مقادیر متني در فیلد آن ممکن است بعضي از کاراکتر ها باشند که بطور تصادفي در دستوراتSQL معناي خاصي دارند. به عنوان مثال کارکتر (') در عبارت "John d'Artagan" به عنوان کارکتر کنترلي درنظر گرفته میشود. براي حل این موضوع

دو روش وجود دارد. روش اول گذاشتن كد گريز (escape) كه با (' ') مشخص مي شوند و همچنين در اين روش بايد ساير كاركترها واحتمالات را در نظر گرفت و كار مشكلي مي باشد. روش دوم و بهتر اين كه با تابع setString() از PreparedStatement به طور اتوماتيك تمام كنترل هاي ممكن را روي رشته انجام دهيم و نوشتن با اين روش مستقل از بانك اطلاعاتي وكدهاي مختلف است.
مزيت ديگر اين روش زماني كه دربرنامه هاي وب از بانك اطلاعاتي استفاده ميكنيم از هك شدن جلو گيري مي كند.

```
PreparedStatement pstmt = con.prepareStatement(
"INSERT INTO MUSKETEERS (NAME) VALUES (?)");

pstmt.setString(1, "John d'Artagan");

pstmt.executeUpdate();
```

استفاده مجدد از اشياء بانك اطلاعاتي

در برنامه مختلف همانند برنامه هاي سرويس دهنده وب و سرويس دهنده برنامه بعد از استفاده JDBC Connection و بستن آن براي كاربران ديگر نياز به باز كردن Connection ديگر داريم و اين خود چند ثانيه وقت سرويس دهنده را تلف مي كند.
براي حل مشكل از اشياء مخصوص بانك اطلاعاتي دوباره براي كاربران ديگر استفاده ميكنيم.
اين روش وابسته به نوع برنامه كه استفاده ميكنيم است و بطور مثال در Servelt ها به كمك متد init() از كلاس Servelt ارتباط به بانك را براي استفاده هاي متوالي را برقرار ميكنيم .متد init() وقتي يك شي Servelt براي اولين با ايجاد ميشود فراخواني ميشود .در EJB(Enterprise JavaBean) اينكار رابه كمك متد ejbLoad() از آن Bean انجام مي دهيم. مثال مربوط به استفاده مجدد از بانك اطلاعاتي در فصل Servelt موجود است.

تراكنشها (Transactions)

در بانكهاي رابطه اي مدرن امروزي يك از قابليتهاي مهم پشتيباني از تراكنشها ميباشد و در برنامه هاي سرويس گرا مثل برنامه مالي بانك به صورت online به بيش از يك دستور select در كار با پايگاه داده نياز داريم.فرض كنيد ميخواهيم مقداري پول را از حسابي به حسابي ديگر جابه جا كنيم .
براي اينكار وجود دو حساب چك شده سپس مقدار موجودي حساب اول كه براي انتقال آن مقدار برسي واگر كمتر باشد عمليات متوقف ميشود ودر غير اين صورت ادامه عمليات و بقيه ماجرا.
در اين جا دو راه حل وجود دارد يك گذاشتن تمام اين كنترلها به عهده پايگاه داده رابطه اي كه بطور خود كار تمام اين مراحل راميتواند كنترل كند و در صورت وجود حالت نامعتبر عمليات Rollback را انجام دهد.
راه حل دوم اينكه از داخل برنامه خود بتوانيم عمليات داخل پايگاه داده را كنترل كنيم .
مديريت تراكنش JDBC در شي Connection بوده و به صورت پيش فرز وقتي كه Connection اتصال برقرار ميكند آن در مد auto-commit قرار دارد. اين به اين معني است كه وقتي كه دستور SQL اجرا ميشود يك تراكنش به بانك اطلاعاتي commit(ارسال معتبر) ميشود.بدعي است اگر ارسال commit نباشد تغييرات انجام شده توسط دستور SQL ما نامعتبر بوده وبه حالت قبلي خواهند برگشت. براي كنترل ارسال معتبر يا نامعتبر به كمك متد setAutoCommit(false) بر روي شي Connection

میتوان دستورات را نامعتبر کرد.به منظور چك كردن وضعيت auto-commit متد getAtuoCommit() رابه كار ميبريم.
زماني كه تمام دستورات برنامه و SQL شما بطور كامل اجرا شد براي نهاي كردن تراكنش با فراخواني متد commit() تمام ركورد ها در بانك اطلاعاتي بصورت ماندگار ثبت ميشوند.حال اگر در برنامه، شما با خطاي مواجه شديد به كمك متد rollback() ميتوانيد اطلاعات قبل از تراكنش را برگردانيد.
در مثال زير نحوه استفاده از تراكنش را مشاهده ميكنيد.در بانك اطلاعاتي جدول INVENTORY(شامل ID محصول و قيمت آن)و جدول SHIPPING(شامل ID محصول ، شماره ترتيب و هزينه حمل shipped) را داريم . با متد chargeCard() (در اين جا كد آن نوشته نشده) كارت اعتباري مشتري را كنترل ميكنيم و در داخل آن اگر خطاي به وجود آيد يك استثناء صادر مي كند و موجب ميشود Rollback عمليات Update انجام شده در بانك را به حالت قبل از آن برمي گردانند.

```
import java.io.*;
import java.sql.*;

public class transaction {
public static void main(String args[]) {

Connection con = null;
try {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
con = DriverManager.getConnection("jdbc:odbc:ordersdb", "user", "passwd");
// Turn on transactions
con.setAutoCommit(false);
Statement stmt = con.createStatement();
stmt.executeUpdate(
"UPDATE INVENTORY SET STOCK = (STOCK – 10) WHERE PRODUCTID =
7");
stmt.executeUpdate(
"UPDATE SHIPPING SET SHIPPED = (SHIPPED + 10) WHERE PRODUCTID =
7");
chargeCard(); // method doesn't actually exist...
con.commit();
System.out.println("Order successful! Thanks for your business!");
}
catch (Exception e) {
// Any error is grounds for rollback
try {
con.rollback();
}
catch (SQLException ignored) { }
System.out.println("Order failed. Please contact technical support.");
}
finally {
// Clean up.
```

```
try {
if (con != null) con.close();
}
catch (SQLException ignored) { }
}
}
}
```

سایر امکانات JDBC

در انجا کمي به روالهاي ذخيره شده در بانك اطلاعاتي، نحوه گرفتن داده ها با نوع پيچيده و كار با متنهاي طولاني را بيان خواهيم كرد.

روالهاي ذخيره شده (Stored Procedures)

اكثر بانكهاي اطلاعاتي رابطهاي يك زبان برنامه نويسي داخلي دارند همانند PL/SQL در Oracle و اين امكان را به توسعه دهندهگان بانك اطلاعاتي ميدهند تا روالهاي مورد نياز در برنامه خود را در داخل بانك اطلاعاتي بنويسند و آنها را از برنامه فراخواني كنند.

اين روالها براي انجام اعمال واقعي در بانك اطلاعاتي بسيار مناسب ميباشند و خود بانكهاي اطلاعاتي از آنها بسيار استفاده مي كنند. در كد زير روال نوشته شده در Oracle PL/SQL را مشاهده ميكنيد.

```
CREATE OR REPLACE PROCEDURE sp_interest
(id IN INTEGER bal IN OUT FLOAT) IS
BEGIN
SELECT balance
INTO bal
FROM accounts
WHERE account_id = id;
bal := bal + bal * 0.03;
UPDATE accounts
SET balance = bal
WHERE account_id = id;
END;
```

اين روال با اجراي دستور SQL عمل محاسباتي را بر روي فيلد bal جدول accounts بانك انجام داده و آن را به روز ميكند.
نوشتن اين روالها آسان بوده و مزاياي بسياري دارند. از جمله :

- روالهاي ذخيره شده در بانك اطلاعاتي پيش كامپايل شده و سرعت آنها از dynamic SQL بيشتر مي باشد.
- روالهاي ذخيره در داخل RDBMS اجرا ميشوند و ميتوان با در خواستهاي مختلف جدا آنها را اجرا كرد بدون ترافيك شبكه.
- روالهاي ذخيره در بانك اطلاعاتي يك بار نوشته ميشوند و بارها توسط برنامه هاي كاربردي مختلف به زبانهاي مختلف برنامه نويسي استفاده مي شوند.

- اگر در جدول بانك تغييري ايجاد شود فقط نياز به تغيير كد روالهاي ذخيره شده داريم و لازم نيست كد برنامه ها تغيير كند.

هر كدام از بانكهاي اطلاعاتي براي دستيابي به روالهاي ذخيره شده نحوه خاص خود را دارد. در JDBC يك استاندارد براي دستيابي به روالهاي ذخيره شده در كلاس java.sql.callableStatement وجود دارد.
روش استفاده از روالي كه مقداري بر نميگرداند به فرم { call procedure_name(?,?) } و همچنين روالي كه يك مقدار خروجي برمي گرداند { ? = call procedure_name(?,?) } مي باشد.
استفاده از CallableStatement همانند استفاده از كلاس PreparedStatement مي باشد.

```
CallableStatment cstmt = con.prepareCall("{call sp_interest(?,?)}");
cstmt.registerOutParameter(2, java.sql.Types.FLOAT);
cstmt.setInt(1, accountID);
cstmt.execute();
System.out.println("New Balance: " + cstmt.getFloat(2));
```

در كد بالا با استفاده از متد prepareCall() از شي Connection يك CallableStatement ايجاد مي كند.چون روال مورد نظر ما خروجي دارد با كمك متد registerOutParameter() از CallableStatement نوع خروجي شناسايي مي شود كه از نوع FLOAT (فيلد شماره ۲ كه خروجي است) ميباشد. از setInt() براي دادن مقدار متغيير accountID به پارمتر ورودي(شماره ۱) استفاده مي كنيم.
سرانجام پس از اجراي روال با متد getFloat() خروجي را چاپ مي كنيم. لازم به ذكر است كه استفاده از متدها ي getXXX() همانند استفاده آنها در ResultSet است.

كار با دادهاي متني و باينري

اغلب بانكهاي اطلاعاتي انواع دادها مانند متنها با چندين گيگابايت اندازه و اطلاعات باينري همانند فايلهاي multimedia را پشتيباني مي كنند.
بانكهاي اطلاعاتي مختلف با روشهاي مختلفي اين كار را انجام ميدهند اما با متدهاي JDBC ميتوانيم آنها را در فرمت استاندارد در يافت و ارسال كنيم.
متد getAsciiStream() براي گرفتن داده هاي متني و متد getBineryStream() براي گرفتن اشياء دودوي با اندازه بالا از ResultSet قابل استفاده اند و هر دوي اين متدها يك InputStream بر مي گردانند.
پشتيباني از داده ها بااندازه بالا يكي از مسائل در JDBC است. براي اين كار ابتدا مطمئن شويد كه درايور شما درست كار ميكند و آن را تست كنيد.درايور هاي Oracle JDBC براي داده با اندازه بالا در اين ضمينه مستعد مشكل ميباشد.
در كد زير نحوه خواندن ASCII String نشان داده شده و ما فرض كرده ايم كه Connection , Statement و بقيه قبلآ ايجاد شده اند.

```
try {
ResultSet rs = stmt.executeQuery(
"SELECT TITLE, SENDER, MESSAGE FROM MESSAGES WHERE MESSAGE_ID
= 9");
if (rs.next()) {
System.out.println(rs.getString("title") );
System.out.println("From: " + rs.getString("sender")  );
BufferedReader msgText = new BufferedReader(
new InputStreamReader(rs.getAsciiStream("message")));
while (msgText.ready()) {
System.out.println(msgText.readLine());
```

```
}
}
}
catch (SQLException e) {
// Report it
}
```

در کد بالا فرض کرده ایم که يك جدول حاوي پيغامها(عنوان، فرستنده، پيغام) داريم و در ابتدا عنوان پيغام و فرستنده آن را چاپ کرده و سپس پيغام را با گرفتن InputStream در حلقه while خط به خط چاپ ميکنيم.

در کد تا زماني که از InputStream ميخوانيم نبايد مقدار ساير ستونها را از ResultSet گرفت و اين بسيار مهم است چون هرگونه فراخواني متدهاي getXXX() از ResultSet جريان ورودي(InputStream) را خواهد بست.
داده باينري با getBineryStream() از ResultSet گرفته ميشود و بايد نوع محتواي آن با setContentType("image/gif") مشخص کرد. در اين مثال نوع آن فايل تصوير با پسوند GIF است.

****

```
Connection con;

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(
"SELECT IMAGE FROM PICTURES WHERE PID = 7"));

if (rs.next()) {
BufferedInputStream gifData =
new BufferedInputStream(rs.getBinaryStream("image"));
byte[] buf = new byte[4 * 1024]; // 4K buffer
int len;

while ((len = gifData.read(buf, 0, buf.length)) != -1) {
out.write(buf, 0, len);
}
}
```

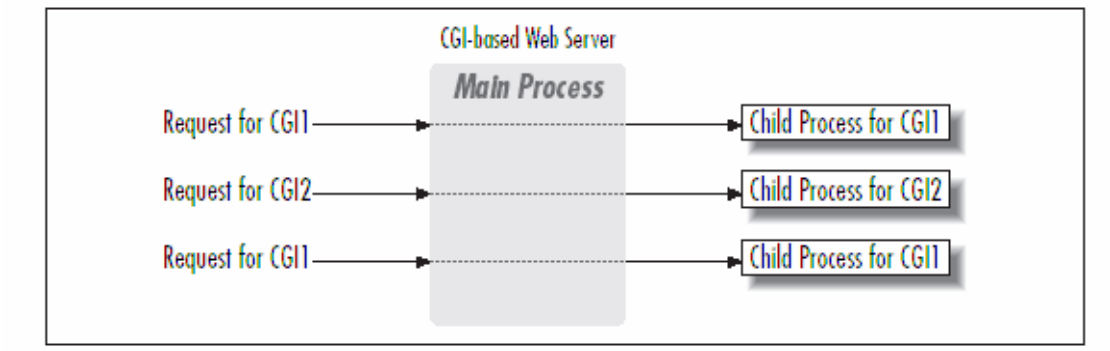
فصل پنجم

Servlet

در بعض از مواقع در سرويس دهنده وب به علت اجراي صفحات در حالت مفسري ما مي توانيم براي افزايش سرعت اجرا از برنامه هاي كامپايل شده استفاده كنيم Servelt كامپوننتهاي كليدي سمت سرويس دهنده جاوا مي باشند كه براي دسترسي به آنها ازروش مشابه در CGI (Common Getaway Interface) استفاده ميشود.

Common Getaway Interface

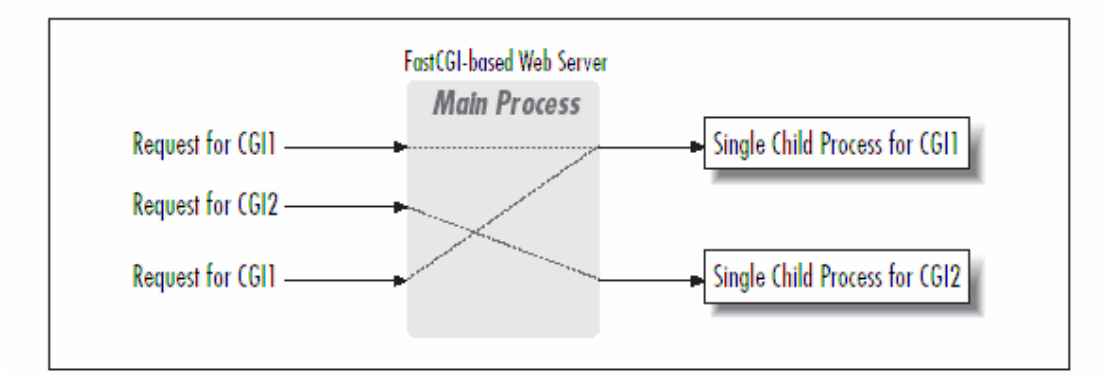
به كمك CGi يك سرويس دهنده وب مي تواند Request فرستاده شده از طرف Client را به يك برنامه خارجي ارجاع دهد و خروجي آن برنامه به Client فرستاده شود . مزيت CGI در اين است كه ميتوان مي توان برنامه هاي وب پويا با برنامه هاي خارج از سرويس دهنده وب را تركيب كرد و سرعت اجرا به ويژه در سرويس هاي وبي كه خاصيت پردازشي و محاسباتي داردند به طور قابل ملاحظهاي بهبود بخشيد.
شكل زير نحوه ارسال درخواست سرويس گيرنده را نشان ميدهد .



برنامه CGI ميتواند با اغلب زبانهاي برنامه نويسي نوشته شود .زبان Perl يكي از زبانهاي رايج در نوشتن CGI مي باشد و قابليت پردازش متن را در خود دارد . همچنين مستقل از سخت افزار ميباشد اما نياز به اجراي مفسر Perl در هر درخواست Client است و مشكل زياد شدن زمان پردازش و منابع اضافي را دارد.
در CGI برنامه CGI با سرويس دهنده وب تعامل لازم را ندارد چون هر دو در فرآيندهاي جدا اجرا ميشوند بطور مثال يك CGI script نمي تواند در يك فايل ثبت وقايع (Log) مربوط به سرويس دهنده وب بنويسد

FastCGI

FastCGI همانند CGI بوده با اين تفاوت كه فرآيند براي هر درخواست سرويس گيرنده را حذف كرده و براي تمام درخواستهاي يك فرآيند ايجاد ميكند اي عمل درشكل زير نشان داده شده.



FastCGI نسبت به CGI سريعتر است اما باز هم نياز به اجراي مفسر Perl براي هر اتصال سرويس گيرنده مي باشد

mod_perl

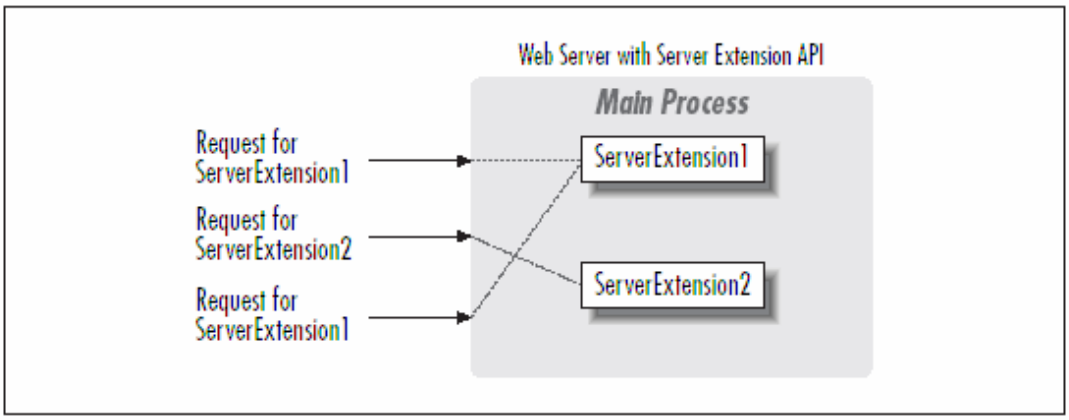
در روش اگر از سرويس دهنده وب Apache استفاده ميكنيد ميتوانيد يك كپي از مفسر Perl را در Apache جاسازي كنيد و به قابليتهاي Perl دسترسي مستقيم داشته باشيد . به كمك mod_perl سرور خود CGI را تفسير كرده و سرعت اجرا بالا تر ميرود .

PerlEx

PerlEx از API هاي محلي سرور وب استفاده ميكند و كارائي بهبود مي يابد ودر سرويس دهنده ويندوز NT يا (Microsoft Internet Information Server) قابل استفاده ميباشد.

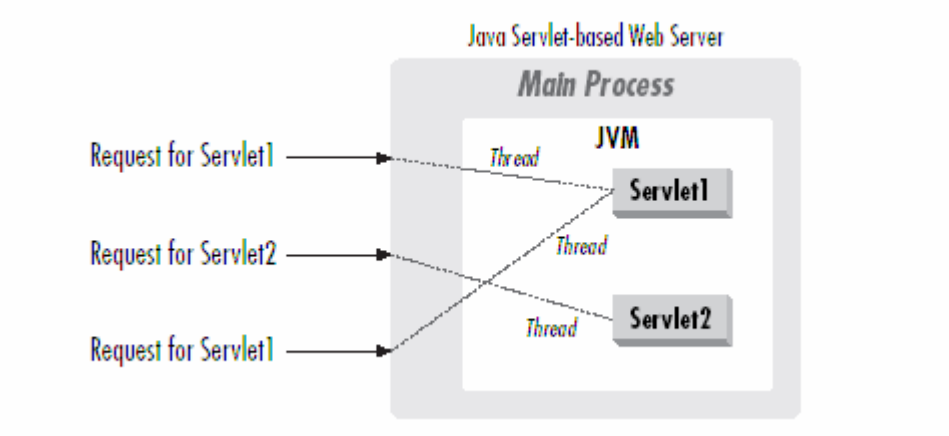
Server Extension APIs

بعضي از شركتها شروع به ايجاد APIهاي با قابليت الحاق به سرويس دهنده وب ميباشند . براي مثال Netscape يكسري API هاي داخلي به نام NSAPI كه اكنون با نام WAI شناخته ميشود . شركت ميكروسافت نيز ISAPI (Internet Service API) براي سرويس دهنده IIS خود ارائه كرد.
با كمك اين API ها مي توان با كدهاي C يا C++ ارتباط برقرار كرده و كد هاي الحاقي باعث بهبود سرعت اجرا و استفاده مناسب از منابع ميشود . در نوع server extension ديگر مثل perl نياز به اجراي يك فرايند براي هر سرويس گيرنده نيست و يك server extension براي نياز ما كافي ميباشد.
همچنين تمام server extensionها در داخل يك فرآيند اجرا ميشوند. اين اعمال در شكل زير كشيده شده .



Java Servelt

Servelt يك كلاس جاوا بصورت Server Extension است كه مي تواند به طور پويا داخل سرور شود و معمولاً در سرور وب جاوا JSP استفاده مي شود.
Servelt همانند Server Extension گفته شده ميباشد با اين تفاوت كه Servelt در داخل JVM بر روي سرور اجرا ميشود (در شكل مشخص شده)
Servelt همانند Applet نياز به پشتيباني جاوا در مرورگر ندارد به دليل اينكه در سمت سرور مي باشد.



در Servelt بجاي استفاده از فرآيند براي اجراي آن از نخ يا Thread استفاده ميشود برعكس CGI كه براي هر سرويس گيرنده فرآيند اجرا مي كرد . هر Servelt براي اجراي خود يك نخ در داخل فرآيند سرور اجرا ميشود و در تعامل نزديك با سرويس دهنده است . مزيت ديگر Servelt قابليت حمل آن است و قابل اجرا در تمام سيستم عامل ها است و بعضي بر اين باروند كه Java Servelt بهترين زير ساخت براي توسعه برنامه هاي وب است .
Servelt جايگزين مناسبي براي CGI است و در انواع سرور ها ميتواند بسط يابد. براي مثال فرض كنيد كه يك سرويس دهنده FTP بر پايه جاوا داريم كه دستورات را با Servelt پشتيباني مي كنند.بمنظور چك كردن فايلها از نظر ويروس و دستورات جديد به راحتي مي توانيم Servelt جديد به سرور اضافه كنيم .

موتور Servelt

يك سروس دهنده براي اجراي Servelt نياز به موتور يا پردازش گر مخصوص به خود دارد ودر سرويس دهنده JSP از آن پشتيباني شده وبطور داخلي از Servelt پشتيباني ميكند . ميتوان براي ساير سرويس دهنده ها نيز از Servelt استفاده كرد وفقط نياز به نصب موتور Servelt دارد. براي سرويس دهندهاي وب Apache , Netscape fastTrak Microsoft IIS , IBM WebSphere, موتور Servelt طراحي شده و ميتوان براي استفاده از Servelt آن را نصب كرد.

مزيت هاي Servelt

قابليت حمل:

به دليل اينكه Servelt به زبان جاوا بوده، خوش تعريف بودن و استفاده از انواع كلاسهاي پايه جاوا قابليت حمل بالاي ميان سيستم عامل ها دارد . يعني شما ميتوانيد Servelt رايكبار نوشته ودر انواع سرويس دهنده ها اجرا كنيد .

قدرت

Servelt ميتواند از API هاي مركزي جاوا مثل API هاي networking , URL , multithreading , image manipulation , data compression , database connectivity , وساير امكانات را remote method invocation , CORBA , object serialization پشتيباني ميكند . همچنين اگر برنامه شما نياز به ايجاد برنامه استفاده كننده از فهرست راهنما (directory service) بر پايه وب داشته باشد JNDI API كمك موثري ميتواند باشد.
Servelt ميتواند با استفاده از كامپونتهاي Enterprise JavaBeans همانند يك Application Server در سرويس دهنده وب استفاده شود.

راندمان و قابليت تحمل

زمانيكه يك Servelt بازگذاري ميشود به طور عام در حافظه سرويس دهنده وب همانند يك شي تنها باقي مانده و در مرحل بعد سرويس دهنده با يك فراخواني ساده همانند فراخواني تبع لز آن استفاده ميكند .برعكس در CGI بايد از فرآيند CGI اين عمل راانجام داد كه پيچيده تر ميباشد .
همچنين Servelt براي فراخواني هاي همزمان و ياچند تاي client فقط براي آنها نخ (thread) جديد اجرا كرده منابع كمتر به هدر ميرود.
Servelt همانند اشياء داخل حافظه هستند كه حالت خود را مديريت كرده و مي توانند منابع خود را مثل اتصال دائم به بانك اطلاعاتي و غيره نگهداري كنند.

ايمني

Servelt از مسائل برنامه نويسي امن پشتيباني درسطوح مختلف پشتيباني مي كند.
چون از زبان جاوا strong type safety به ارث مي برد . بطور مثال زماني كه اقلب مقادير در برنامه CGI شامل اقلام عددي شبيه شماره پورت سرويس دهنده است با آنها شبيه مقدارهاي String در ارسال برخورد ميشود اما در Servelt همانند نوع واقعي خود برخورد شده .
همچنين اشغال جمع كن جاوا (garbage collector) در هنگام احتياج نداشتن به Servelt آنرا به طور خودكار ازحافظه برداشته تا از ارجاع نامعتبر اشاره گرها و اشغال حافظه جلوگيري شود.
Servelt به كمك مكانيزم java exception handling خطاها را دستگير يا handle ميكند تا از االجراي غير قانوني كد جلوگيري شود.

يكپارچگي

Servelt ها بطور كامل با سرويس دهنده جاوا يكپارچه هستند و به Servelt امكان همكاري و هماهنگي با سرور را دارد براي مثال Servelt ميتواند از امكان ترجمه مسير فايلها در دايكتوري مجازي برا ي ثبت وقايع و logها استفاده كند .

قابليت انعطاف و توسعه پذيري

Servletها قابليت توسعه بالاي دارند امروزه كه از كلاسهاي APIهاي بهينه شده HTTP Servelt استفاده ميكنيم در آينده مي توانيم از همان HTTP Servelt بهينه(optimize) شده استفاده كنيم بدون تغيير در كد . شركت سان سالانه تمام APIهاي خود بهتر وجديد تر ميكند و در JSDK ورژن جديد ارائه ميدهد.
Servelt انعطاف بالايي دارد بطور مثال ميتوانبد Servletرا براي ايجاد صفحه وب استفاده كنيد و به يك صفحه ايستا با تگ <SERVELT> آنها را اضافه كنيد. اين كار در شركتهاي استفاده مي شود كه تعداد Servelt زيادي دارد و به كمك اين روش محتواي آنها را فيلتر مي كنند در از كد نويسي اظافي وپيچيده تر شدن جلوگيري شود. به اين عمل Servelt Chain مي گويند.

HTTP Servelt

در اين قسمت با نحوه كار HTTP Servelt ، ايجاد صفحه HTML و دسترسي به آن توسط نام آن اشنا مي شويم.
ميتوان يك HTTP Servelt را در داخل يك صفحه HTML جا سازي كرد بطوري كه متدهاي آن در سمت سرور اجرا ميشوند.براي ايجاد قدرت بيشتر و استفاده از تكنيك فيلتر كردن محتوا اين قابليت وجود دارد كه Servletها به هم ديگر زنجير شوند.
وقتي كه ما Servelt را در داخل صفحه HTML بنويسيم به اين تكنولوژي Java Server Page ميگويند.

درخواست وپاسخ(Request , Response)

ارتباط در اينترنت به صورت است كه سرويس گيرنده با ارسال دخواست به سرويس دهنده وب آماده گي خود را براي دريافت اطلاعات يا سرويس اعلام ميدارد سپس سرويس دهنده با توجه به شرايط به سرويس گيرنده پاسخ ميدهد.
زمانيكه سرويس گيرنده درخواست را ارسال ميكند اولين كار اين است كه در خواست به كمك متدي در سرور پردازش مي شود اين درخواست شامل اطلاعات در باره سرويس گيرنده ميباشد كه به سرور ميگويد كدام كار بايد انجام شود.
اولين خط درخواست شامل آدرس URL صفحه درخواستي و ورژن پرتكول HTTP است.

GET /intro.html HTTP/1.0

درخواست از متد GET براي سوال درباره صفحه intro.html اسفاده ميكند و پس از ارسال درخواست سرويس گيرنده اطلاعات اختياري را ارسال مي كند. اين اطلاعات اختياري ميتواند شامل نام نرم افزار سرويس گيرنده يا مرورگر آن ، انواع فايلهاي كه تواناي نشان دادن آنها را دارد وساير اطلاعات باشد.

User-Agent: Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)
Accept: image/gif, image/jpeg, text/*, */*

سرويس دهنده پس از گرفتن درخواست يك پاسخ ارسال ميكند. اولين خط پاسخ يك خط وضعيت است براي مثال :

HTTP/1.0 200 OK
در خط وضعيت ابتدا ورژن پرتكول HTTP نوشته ميشود و بعد از آن عدد 200 نشانه دريافت درخواست به صورت موفق است. كلمه "OK" به اين معني كه صفحه درخواستي مورد نظر سرويس گيرنده در سرور موجود است .
پس از ارسال خط وضعيت سرويس دهنده اطلاعاتي در مورد نرم افزار و انواع محتوا در سرور به سرويس گيرنده ارسال ميكند.

Date: Saturday, 23-May-2003 03:25:12 GMT
Server: JavaWebServer/1.1.1
MIME-version: 1.0
Content-type: text/html
Content-length: 1029
Last-modified: Thursday, 7-May-2003 12:15:35 GMT
در ابتدا زمان سرويس دهنده و نوع نرم افزار سرور مشخص ميشود.
MIME انوع داده هاي media را كه سرويس دهنده ارسال ميكند را مشخصمي كند.
Content-type معرف شكل فايلي كه درخواست شده است. در انتها تاريخ آخرين تغيير در فايل ذكر ميشود.

GET, POST

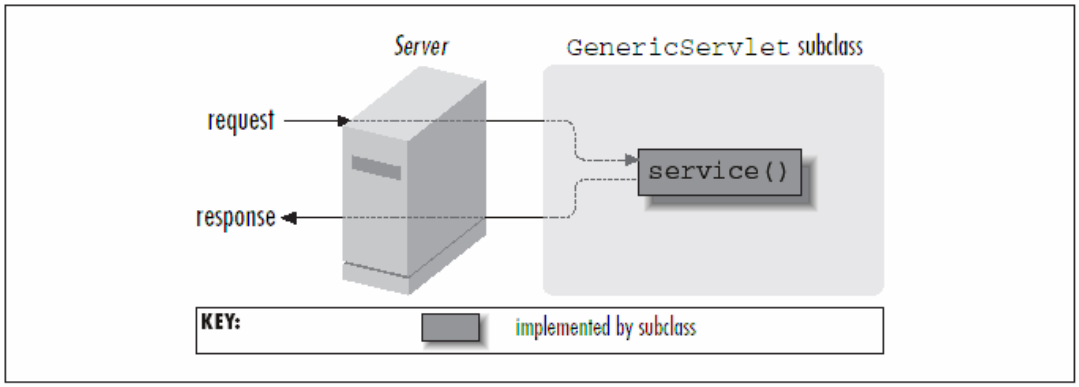
سرويس گيرنده به دو روش مي تواند درخواست را ارسال كند روش اول با متد GET كه براي گرفتن اطلاعات با حجم كم و با امنيت كمتر مثل متن، چارت و يا نتيجه در خواست از يك بانك اطلاعاتي. روش دوم با POST براي ارسال اطلاعات با حجم يا امنيت بالاتر مثل شماره كارت اعتباري ، اطلاعات شخصي و يا اطلاعات كه در داخل بانك اطلاعاتي ذخيره مي شوند.
اطلاعات در GET به صورت يك دنباله از كاركترها در شكل URL ارسال مي شوند و به اين نوع URL ها query string ميگويند. به همين دليل مي باشد كه در GET نمي توانيم اطلاعات با حجم بالا را ارسال كنيم چون حجم URL تا 240 كاركتر ميباشد.
متد POST از تكنولوژي جدا از GET استفاده ميكند و مي توانيم اطلاعات در حد مگا بايت را ارسال كنيم. در POST از ارتباط مستقيم با socket از پرتكول HTTP استفاده كرده و محدوديت ارسال نداريم.

ساير متدها

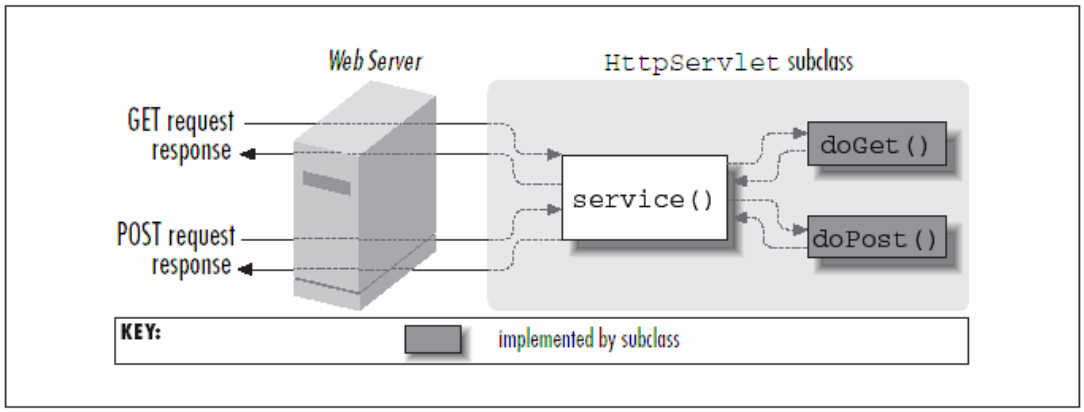
علاو بر GET , POSTچند متد ديگر وجود دارند كه كمتر استفاده ميشوند.متد HEAD كه توسط سرويس گيرنده ارسال ميشود براي زماني كه فقط مي خواهد سرآيند Response را ببيند يا اندازه سند ارسالي راتعين كند و زمان را تغيير دهد و ساير موارد استفاده كند. متد OPTION براي سوال از سرويس دهنده در مورد قابليتهاي اضافي كه پشتيباني مي كند و منابعي كه در آن قابل دسترس ميباشند.
متد TRACEبه اشكال گيري debugging درخواست ارسالي در مورد محتواي آن به سزويس گيرنده كمك ميكند. متد PUT براي قرار دادن يك سند به طور مستقيم در روي سرويس دهنده است و متد DELETE برعكس متد PUT ميباشد.

Servelt API

براي ايجاد HTTP Servelt بايد از API هاي مخصوص اين كار كه در بسته هاي javax.servelt.http, javax.servelt قرار دارند استفاده كنيم .
بسته javax.servelt شامل كلاسهاي براي پشتيباني از Servelt و Servelt مستقل از پرتكول است همچنين براي اضافه كردن قابليتهاي ويژه HTTP بسته javax.servelt.http را به برنامه extend مي كنيم.
تمامي Servelt ها بايد بايد رابط javax.servelt.Servelt را كلاس خود تكميل كنند.اكثر Servelتها آن را با extend كردناز بين كلاس javax.servelt.GenericServelt يا javax.servelt.http.HttpServelt انجام ميدهند.
براي مستقل از پرتكول بودن Servelt بايد از GenericServelt استفاده و براي به كار بردن قابليتهاي HTTP از زير كلاس HttpServelt داخل خود GenericServelt استفاده مي كنيم.
در Servelt همانند Applet ما متد main()نداريم(در برنامه هاي معمولي جاوا بايد حتماً موجود باشد).به جاي آن متد، سرور در هنگام گرفتن Request از سرويس گيرنده متد service()از Servelt را فراخواني ميكند.
يك Servelt در حالت كلي بايد متد service()را خود پياده سازي(override) كند تا درخواست مخصوص سرويس گيرنده را دستگير(handle) كند.
متد service()دو پارامتر قبول ميكند اولي يك شي Request و دومي يك شي Response است.شي Request براي گرفتن درخواست سرويس گيرنده و دادن آن به Servelt و شي Response براي فرستادن پاسخ به سرويس گيرنده مي باشد.
شكل زير اين موضوع را بيان ميكند.



در مقابل HttpServelt متد service()را پياده سازي نمي كند و به جاي آن متد doGet()براي گرفتن درخواستهاي GET و از متد doPost()براي گرفتن درخواستهاي Post استفاده ميكند.
بسته HttpServelt اين قابليت را دارد كه هر دوي اين متدها را پياده سازي(override) كند و مستقل از نوع درخواست شود.
متد service()از HttpServelt تمام متدهاي doXXX()را ميتواند به كار برد و آن را override نمي كند.
در شكل زير اين موضوع نشان داده شده.



علاوه بر كلاسهاي گفته شده در javax.servelt دو كلاس ServeltRequest ,ServeltResponse براي دستيابي Servelt به اشياء Server Response , Server Request و همچنين HttpServeltRequest , HttpServeltResponse در بسته javax.servelt.Http براي دستيابي به HTTP Request ,HTTP Response استفاده ميشوند.
در بسته javax.servelt.Http ميتوانيم از كلاس HttpCookie , HttpSession براي پردازش جلسه كاري و داده هاي ذخيره شده توسط سرور روي سرويس گيرنده استفاده كنيم.

ايجاد صفحه

اكثر HTTP Servelt ها يك صفحه html ايجاد ميكنند همانند اينكه يك CGI Script به اطلاعات دستيابي پيدا كرده و صفحه وب را ايجاد ميكند اما با تفاوت كه در Servelt ميتوانيم همه وظايف را در داخل سرويس دهنده وب انجام دهيم نه بيرون آن.
در اينجا اولين Servelt را كه پيغام ساده Hello World را به سرويس گيرنده ارسال ميكند نشان ميدهيم.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
res.setContentType("text/html");
  PrintWriter out = res.getWriter();
out.println("<HTML>");
out.println("<HEAD><TITLE>Hello World</TITLE></HEAD>");
out.println("<BODY>");
out.println("<BIG>Hello World</BIG>");
out.println("</BODY></HTML>");
}
}
```

اين Servelt از HttpServelt استفاده كرده و متد doGet() را پياده سازي مي كند. هنگامي كه سرويس دهنده وب يك GET Request دريافت مي كند متد doGet را از Servelt فراخواني كرده و آن درخواست را در شي HttpServeltRequest ميگذارد.
براي نوشتن خروجي و پاسخ به سرويس گيرنده Servelt از HttpServeltResponseكمك گرفته ويك جريان خروجي(PrintStream) استاندارد با متد getWriter() از شي ايجاد ميكند.

PrintWriter وظيفه برگرداندن Unicode هاي كاركتر جاوا را به عهده دارد و در اينجا به صورت English ميباشد.
استفاده از setContentType("text/html") براي تنظيم كردن محتواي خروجي و به صورت html از استانداردهاي MIME مي باشد.

اجراي Hello World

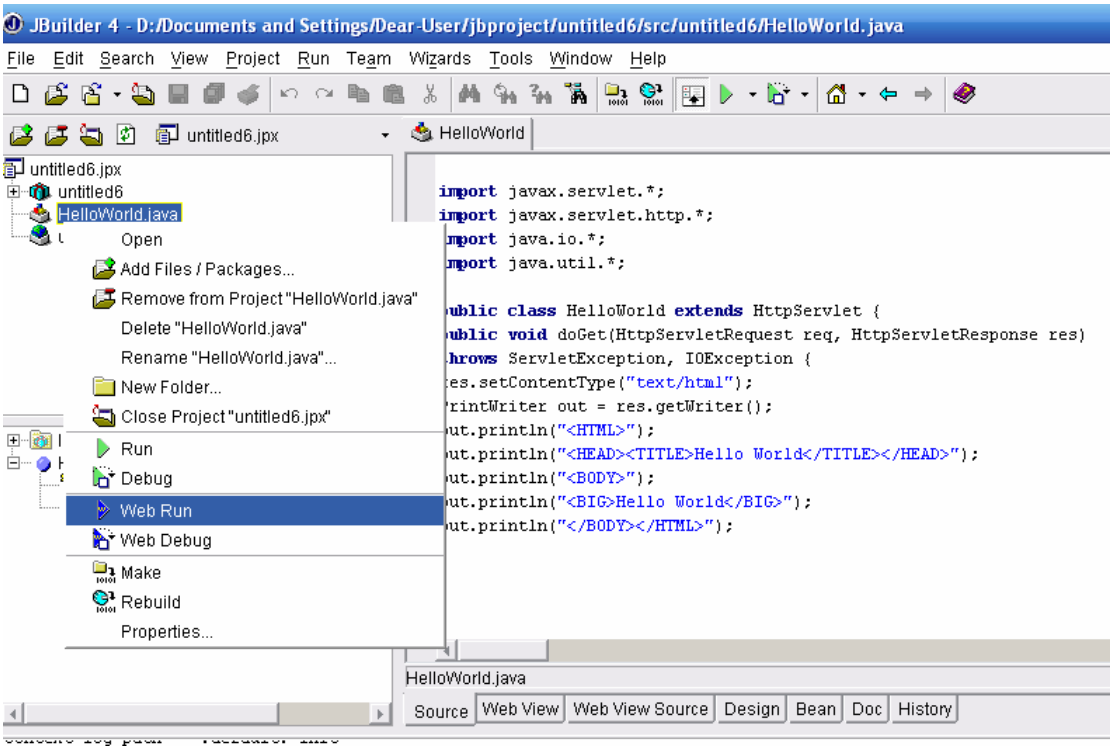
زمانيكه Servelt نوشته شد براي اجراي آن به دو چيز نياز داريم اولي فايل كلاسهاي Servelt API كه برا ي كامپايل لازم هستند و دومي Servelt Engine كه يك سرويس دهنده وب ميباشد.
براي فايل كلاسهاي Servelt API چند راه وجود دارد.

- نصب Java Servelt Development Kit (JSDK) كه در http://java.sun.com/product/servelt موجود است. در آن كد هاي منبع ، Servelt Engine و يك سرويس دهنده وب ساده موجود است.
- نصب يكي از Servelt Engine هاي موجود كه هر كدام فايل كلاسهاي Servelt API را در داخل خود دارد.

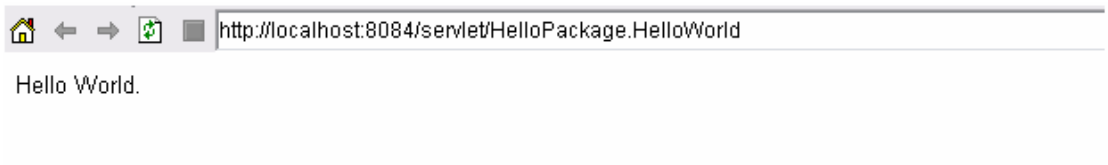
چندين دو جين از Servelt Engine در بازار موجود است كه در قسمت موتور Servelt بيان شدند و ميتوانيد هر كدام از آنها را نصب كنيد.
اگر شما از سرويس دهنده وب جاوا استفاده ميكنيد بايد كد منبع Servelt را در پوشه server_root/Servelt قرارا دهيد(server_root پوشه اي در محل نصب سرويس دهنده است) و اين استاندارد محل فايل كلاسهاي Servelt است. پس گذاشتن كد منبع در مكان درست بايد آنر كامپايل كنيد.
به منظور كامپايل از كامپايلر استاندارد جاوا javac(يا با استفاده از IDE هاي جاوا همانند jbuilder) اين كار را انجام ميدهيم همچنين مطمئن شويد كه بسته هاي javax.servlet , javax.servlet.http در مسير مربوطه موجود باشند.
بعد از كامپايل بايد سرويس دهنده را راه اندازي كنيد تا Servelt را جرا كنيم.برا ي اينكار فايل httpd script (در ويندوز httpd.exe) در server_root/bin پيدا كرده و آن را اجرا كنيد و پيكره بندي پيش فرز در سرور روي پورت 8080 مي باشد.

برا ي اجراي servlet چندين راه وجود دارد ويكي از آنها دستيابي به كمك URL است براي اينكار http://server:8080/servlet/HelloWorld را در مرورگر خود تايپ كنيد.

اگر از IDE معمول مثل JBuilder استفاده ميكنيد كافي است بروي فايل جاوا ايجاد شده كليك راست كرده و گزينه web runرا همانند شكل زير انتخاب كنيد.



اگر Servlet قسمتي از يك بسته(package) باشد براي اجرا بايد URL را به فرم server_root/Servlet/package/name تايپ كنيد.بطور مثال :
http://server:8080/servlet/package.name.HelloWorld

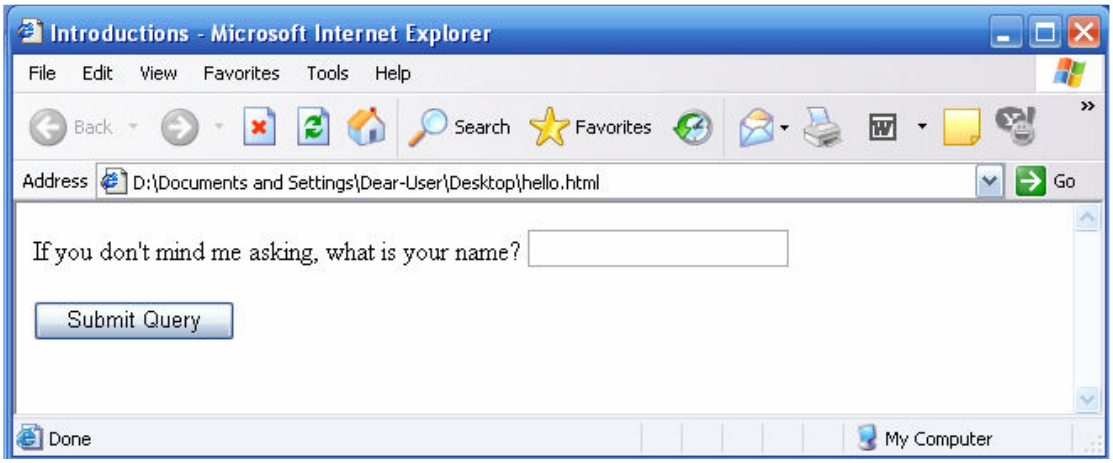


روش ديگر ارجاع به Servlet به كمك نام رجيستر شده مي باشد و مي تواند نام كلاس را روي خود نداشته باشد. در Java Web Server براي رجيستر كردن Servlet بايد از JavaServer Administration Tool كمك گرفت. Administration Tool كه Applet مديريت سرور جاوا است و در پورت 9090 از سرور قابل دسترس است همانند http://server:9090/ سپس در مديريت Web Service قسمت Servlet انتخاب كنيد و پس از آن Add New Servlet. در اين قسمت ميتوانيد نام ويژه اي را براي كلاس Servlet انتخاب كنيد. بطور مثال اگر ما نام "hi" را براي HelloWorld به كمك URL http://server:8080/servelt/hi به دسترسي داريم.
روش سوم دسترسي به Servlet استفاده از Servlet Alias يا نام مستعار كه شبيه به ديگر URLها مي باشد و تفاوت در اين مي باشد كه سرويس دهنده با URL ويژه اي آن Servlet را اجرا ميكند. به طور مثال براي HelloWorld Servlet از http://server:8080/hello.html استفاده مي كنيم. استفاده از alias اين امكان را مي دهد كه استفاده از Servlet را در سايت مخفي كنيم و يك Servlet به جاي يك صفحه وب جايگزين شود.
براي ايجاد نام مستعار در در مديريت Web Service قسمت setup را انتخاب كنيد و سپس Servlet Aliases و در آخر از Add Alias .

گرفتن داده از فرمها

اولين Servlet يعني HelloWorld چندان جالب نبود و حال مي خواهيم يك Servlet ايجاد كنيم كه از يك فرم html كه كد آن در زير موجود است نام كاربر را گرفته و در جواب به وي سلام دهد.

```
<HTML>
<HEAD>
<TITLE>Introductions</TITLE>
</HEAD>
<BODY>
<FORM METHOD=GET ACTION="/servlet/Hello">
If you don't mind me asking, what is your name?
<INPUT TYPE=TEXT NAME="name"><P>
<INPUT TYPE=SUBMIT>
</FORM>
</BODY>
</HTML>
```



هنگامي كه نام توسط كاربر ارسال مي شود آن به Hello Servlet فرستاده مي شود(به دليل اينكه در قسمت ACTION فرم ورودي نام آن را روي سرور وارد كرده ايم). در اينجا ما متد GET در قسمت METHOD براي ارسال استفاده كرده ايم و براي مثال اگرشما نام mehdi را وارد كنيد URL درخواست به فرم
http://server:8080/servlet/hello?name=mehdi خواهد بود. در Servlet مورد نظر به كمك شي HttpServletRequest به داده هاي ارسال شده از صفحه html دستيابي پيدا ميكنيم. در زير كد Hello Servlet نشان داده شده.

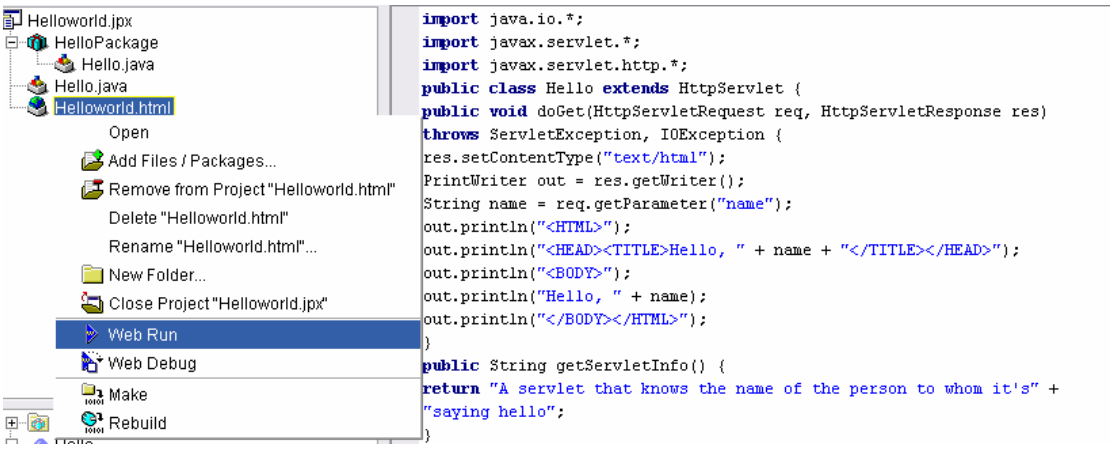
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Hello extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
```

```
throws ServletException, IOException {
res.setContentType("text/html");
PrintWriter out = res.getWriter();
String name = req.getParameter("name");
out.println("<HTML>");
out.println("<HEAD><TITLE>Hello, " + name + "</TITLE></HEAD>");
out.println("<BODY>");
out.println("Hello, " + name);
out.println("</BODY></HTML>");
}
public String getServletInfo() {
return "A servlet that knows the name of the person to whom it's" +
"saying hello";
}
}
```

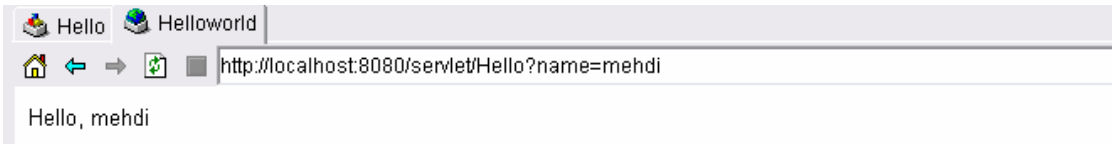
تنها تفاوت اين Servlet با قبلي در اين است كه از متد getParameter("name") از شي Request استفاده كرده ايم . به كمك اين متد به پارامتر ارسال شده از سرويس گيرنده توسط QueryString دستيابي پيدا مي كنيم واگر سرويس گيرنده پارامتر را خالي ارسال كند اين متد null برمي گرداند.

متد getServletInfo() توسط Servlet براي برگرداندن اطلاعات درباره خود Servlet به سرويس دهنده وب مي باشد. اين اطلاعات مي تواند راجبع كاربرد ، نويسنده ، ورژن وساير اطلاعت مربوط به يك Servlet باشد.

حال براي اجراي برنام هموردنظر در محيط JBuilder روي فايل html ساخته شده كليك راست كرده و گزينه web run را انتخاب كنيد.



سپس درقسمت نام نام دلخواه ار وارد كنيد و روي دكمه كليك كنيد. نتيجه به شكل زير خواهد بود.



گرفتن درخواستها به فرم POST

در Servlet قبلي براي گرفتن درخواست GET از متد doGet() استفاده كرديم حال براي گرفتن درخواست POST از متد doPost() بايد استفاده كنيم. تنها تغييرات لازم اين است كه متد doPost() به فرم زير را به Servlet قبلي اضافه كنيم.

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {

doGet(req, res);

}
```

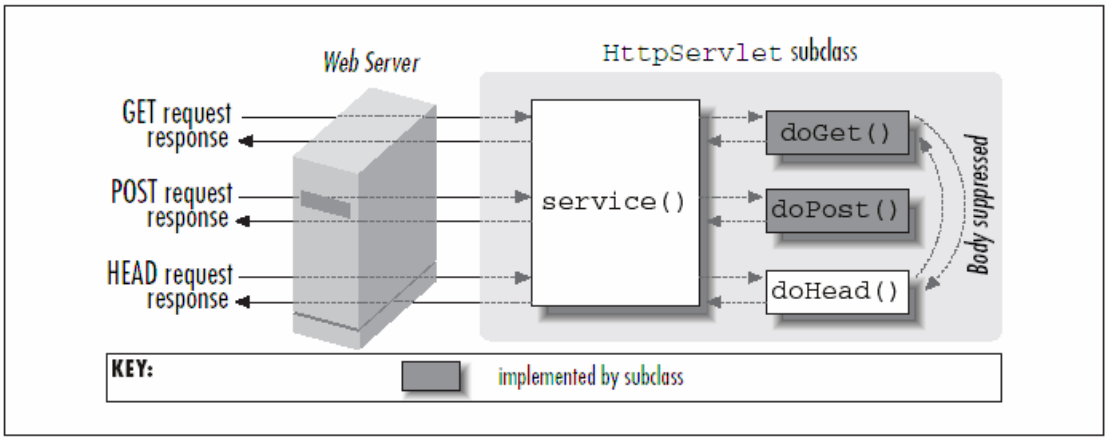
همچنين در فايل html متد را به POST تغيير دهيم.

```
<FORM METHOD=POST ACTION="/servlet/Hello">
```

در حالت كلي بهتر است كه در Servlet هر دو نوع متد doPost() , doGet() را براي قبول كردن هر دو نوع درخواست پياده سازي كنيم. مزيت doPost() در اين است كه قادر به دريافت داده ورودي با اندازه بالا است.
اگر متد doPost() در Servlet پياده سازي نشده باشد و سرويس گيرنده با اين متد ارسال كند از طرف سرويس دهنده يك پيغام خطا به سرويس گيرنده ارسال مي شود.

گرفتن درخواست به فرم HEAD

همان طور كه قبلاً گفته شدمتد HEAD كه توسط سرويس گيرنده ارسال ميشود براي زماني كه فقط مي خواهد سرآيند Response را ببيند يا اندازه سند ارسالي راتعين كند و زمان را تغيير دهد و ساير موارد استفاده كند. مثل دو متد قبلي در اينجا متد doHead() وجود ندارد و براي اينكار از متد doGet() استفاده ميكنيم.
متد service() از HttpServlet درخواست از نوع HEAD را شناساي كرده و به طور ويژه با برخورد ميكند. در زمان رسيدن در خواست از نوع HEAD شي HttpServletResponse آن را دريافت كرده و آنرا به متد doGet() ارسال مي كند.متد doGet() آنرا به صورت عادي پردازش كرده و اما فقط قسمت سرآيند(header) به سرويس گيرنده ارسال مي شود. اين به اين معني است كه قسمت بدنه Response به دور انداخته مي شود.
شكل زير اين جريان را نشان ميدهد.



اگر چه اين استراتژي ساده و راحت است اما شما ميتوانيد كارائي را بهبود دهيد با پيدا كردن درخواست HEAD در داخل متد doGet() .اين كار صرفه جوي را به همراه دارد چون وقت براي خروجي كه لازم نيست فرستاده شود صرف نمي شود.

مثال زير همان Hello Servlet با استراتژي گفته شده است و از متد getMethod() در پياده سازي كمك گرفته.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Hello extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
// Set the Content-Type header
res.setContentType("text/html");
// Return early if this is a HEAD
if (req.getMethod().equals("HEAD")) return;
// Proceed otherwise
PrintWriter out = res.getWriter();
String name = req.getParameter("name");
out.println("<HTML>");
out.println("<HEAD><TITLE>Hello, " + name + "</TITLE></HEAD>");
out.println("<BODY>");
out.println("Hello, " + name);
out.println("</BODY></HTML>");
}
  }
```

توجه كنيد كه هميشه Content-Type سرآيند تنظيم مي شود حتي وقتي كه درخواست HEAD Request مي باشد. Content-Type نيز در سرآيند به سرويس گيرنده ارسال ميشود.

اطمينان پيدا كنيد كه در آخر ارسال پاسخ به HEAD از دستور return استفاده مي كنيد و متد System.exit(0) را فراخواني نكنيد چون باعث خروج از سرويس دهنده وب خواهيد شد.

ضمينه كردن در سمت سرور

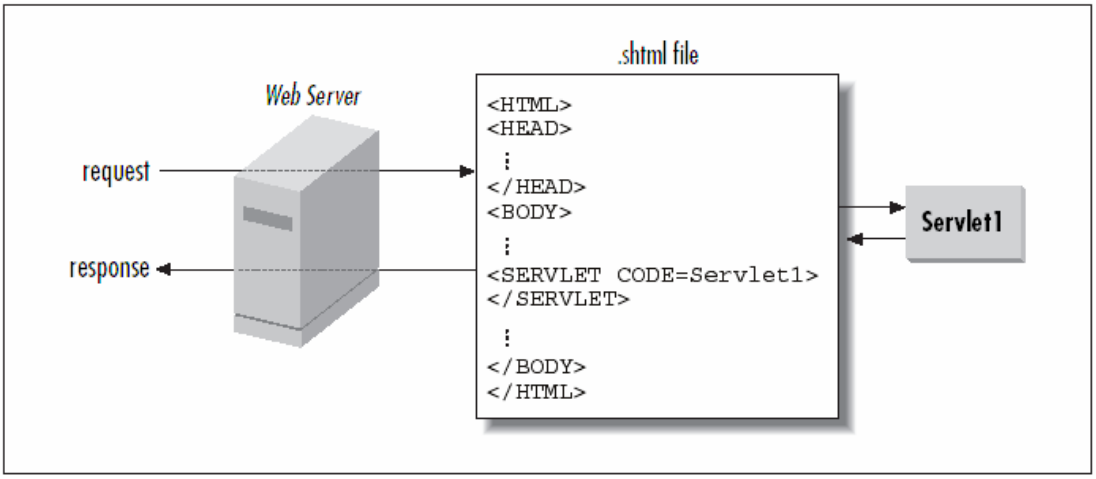
تمامي Servlet هاي كه در قبل مشاهده كرديد بطور كامل صفحه html را ايجاد مي كردند و تمام آن به عهده Servlet بود. Servlet مي تواند در داخل يك صفحه html قرار گيرد كه به آن server-side include(SSI) مي گويند.

اكثر سرويس دهنده ها اينگونه نوشتن را پشتيباني ميكنند و قبل از ارسال صفحه آنرا پردازش كرده وخروجي به سرويس گيرنده ارسال ميشود. اين روش همانند استفاده از Applet است.

```
<SERVLET CODE=ServletName CODEBASE=http://server:port/dir
initParam1=initValue1 initParam2=initValue2>
<PARAM NAME=param1 VALUE=value1>
<PARAM NAME=param2 VALUE=value2>
If you see this text, it means that the web server
providing this page does not support the SERVLET tag.
</SERVLET>
```

در تگ <SERVLET> مشخصه CODE براي نام كلاس Servlet يا نام رجيستر شده آ ن است.CODEBASE كه اختياري مي باشد مكان لود شدن Servlet را نشان ميدهد.

هر پارامتر براي ارسال شدن به Servlet بايد از تگ <PARAM> استفاده كند و Servlet براي دريافت پارامتر ها از متد getParameter() در شي ServletRequest استفاده مي كند. هر متغير ميتواند در انتهاي تگ Servlet مقدار اوليه داشته باشد.
اگر سرويس دهنده از SSI پشتيباني كند تگ Servlet را پردازش كرده واگر از SSI پشتيباني نكند تگ Servlet را به سرويس گيرنده ارسال نخواهد كرد(فقط محتواي داخل آن ارسال مي شود. به طور پيش فرز سرويس دهنده وب جاوا صفحات با پسوند shtml را با تگ Servlet پردازش مي كند(در شكل زير) و سرويس گيرنده تمام چيزهاي بين <SERVLET> </SERVLET> را نخواهد ديد بر عكس Applet.



نوشتن server-side Include

ضمينه كردن Servlet در صفه وب زماني مفيد است كه صفحه در اصل به صورت ايستا باشد اما شامل قسمتي پويا باشد. براي مثال فرض كنيد كه ما چندين صفحه داريم كه بايد زمان جاري را در آن نشان دهيم و اين خود يك چالش است چون بايد در آن ناحيه اي كه سرويس گيرنده قرار دارد زمان را محاسبه كنيم.
به كمك ضمينه كردن در سمت سرويس دهنده و نوشتن راهنماي SSI كه در داخل صفحه html كد هاي جاوا مخصوص اين كار را فراخواني مي كند ميتوان اين كار را انجام داد.
كد زير را در داخل يك فايل با پسوند shtml ذخيره كنيد.

```
<HTML>
<HEAD><TITLE>Times!</TITLE></HEAD>
<BODY>
The current time here is:
<SERVLET CODE=CurrentTime>
</SERVLET>
<P>
The current time in London is:
<SERVLET CODE=CurrentTime>
<PARAM NAME=zone VALUE=GMT>
</SERVLET>
<P>
And the current time in New York is:
<SERVLET CODE=CurrentTime>
```

```
<PARAM NAME=zone VALUE=EST>
</SERVLET>
<P>
</BODY>
</HTML>
```

نام Servlet که از صفحه shtml فراخواني مي شود CurrentTime بوده و براي نشان دادن زمان در منطقه اي که به صورت پارامتر به آن ارسال مي شود است.کد Servlet در زير نوشته شده.

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class CurrentTime extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
PrintWriter out = res.getWriter();
Date date = new Date();
DateFormat df = DateFormat.getInstance();
String zone = req.getParameter("zone");
if (zone != null) {
TimeZone tz = TimeZone.getTimeZone(zone);
df.setTimeZone(tz);
}
out.println(df.format(date));
}
}
```

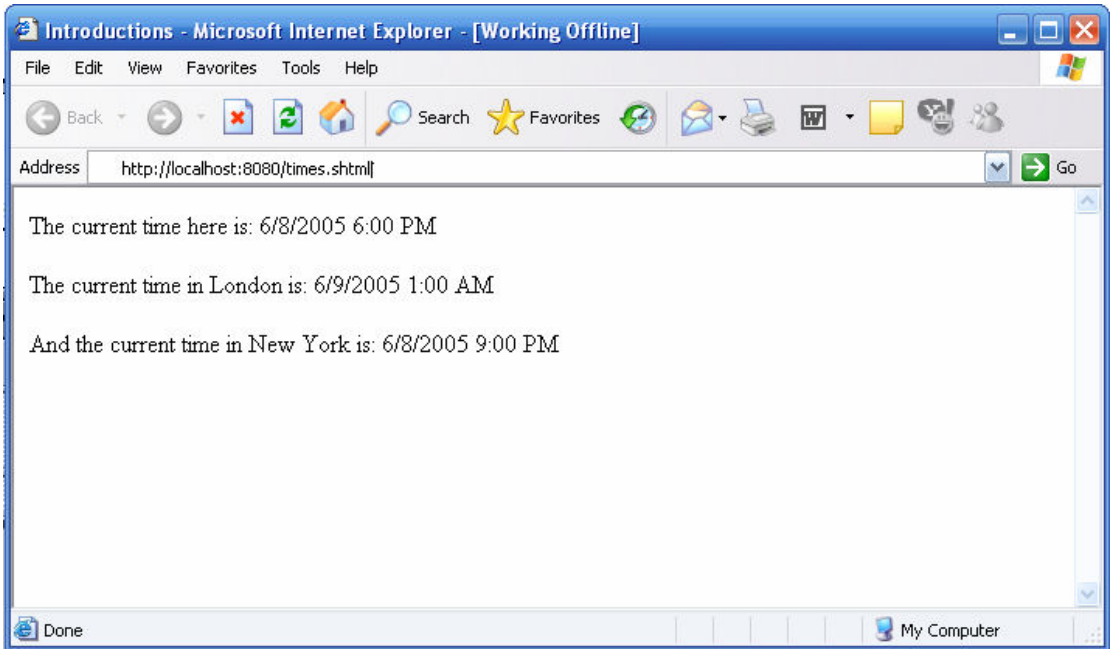
CurrentTime Servlet شبيه به Hello Servlet است و تفاوت چنداني بين آنها وجود ندارد. در CurrentTime Servlet همانند Servlet هاي قبلي از متد doGet() استفاده مي كند. البته Servlet جاسازي شده در صفحه وب محدود ميباشد چون همانند Request از طرف سرويس دهنده قابليت سرآيند HTTP را ندارد.

براي نشان دادن زمان از كلاس Date , DateFormat استفاده مي كنيم. براي تنظيم ناحيه بايد از متد setTimeZone() در شي Dataformat كمك بگيريم.

براي مشخص كردن پارامتر هاي كه توسط متد getParameter() از HttpServletRequest گرفته مي شود در داخل فايل shtml از تگ <PARAM> استفاده مي كنيم.

اگر پارامتري به Servlet ارسال نشود به طور پيش فرز زمان سيستم به خروجي ارسال خواهد شد.

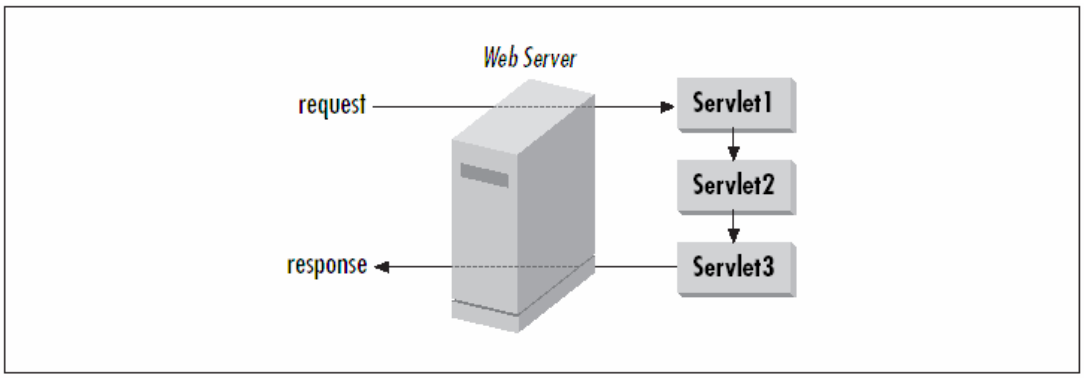
خروجي به شكل زير خواهد بود.



زنجيره سازي و فيلتر كردن Servlet

همكاري Servletها براي ايجاد محتواي صفحات وب را زنجير سازي(Servlet Chaining) مي گويند

در بيشتر سرويس دهنده ها درخواست سرويس گيرنده به كمك دنباله اي از Servlet ها پاسخ داده ميشود. درخواست client ابتدا به Servlet اول در زنجير فرستاده مي شود و پاسخ گرفته شده از آخرين Servlet در زنجير به مرورگر سرويس گيرنده فرستاده مي شود. در ميان هر كدام از Servlet ها خروجي به Servlet بهدي به عنوان ورودي ارسال مي شود. اين عمل در شكل زير نشان داده شده.



دو روش براي به راه انداختن زنجيره وجود دارد. در روش اول به طور صريح URL هاي واقعي زنجير را براي سرور مشخص كرده و در روش دوم براي سرور مشخص مي كنيد كه تمام خروجي Servletها به داخل يك Servlet ويژه رفته و درآخر كار خروجي آن Servlet به سرويس گيرنده ارسال شود.

زماني كه يك Servlet يك محتوا را به ديگر نوع محتوا تبديل مي كند به اين عمل فيلتر(filtering) كردن مي گويند.

زنجيره سازي كردن تفكر شما درباره محتواي واقعي وب را تغيير ميدهد. در ايجا چيزهاي وجود دارند كه شما با آن را انجام دهيد.

-تغير سريع محتواي صفحات
بطور مثال براي بهبود سايت وب خود تمام تگهاي <BLINK> صفحه وب را در سرويس دهنده كه در مثال بعدي نشان داده شده پشتيباني كنيد. در واقع شما با كساني كه نمي توانند با زبان انگليسي صحبت كنند به طور پويا صفحات را براي سرويس گيرنده به زبان خود ترجمه كنيد. همچنين اگر نمي خواهيد متنهاي داخل صفحات وب را هر كاربري بخواند و بعضي متنها براي افراد خاص ظاهر شود. بطور مثال نام پروژه اي مخفي كه هنوز نام آن فاش نشده.
شما مي توانيد كلمات(متنها) تازه را در سايت خود بالا ببريد مثلآ يك مجله خبري online و در اينجا نياز به يك يك Servlet داريد كه نام بالاي هزار شركت را را بطور اتوماتيك پيدا كرده و در صفحه خانگي سايت براي آنها لينك ايجاد كند.
-ايجاد هسته براي محتوا و نمايش آن در فرمت ويژه
بطور مثال شما ميتوانيد تگهاي دلخواه را در صفحات وب جاي داده و Servletمخصوص كه اين جايگذاري با html انجام دهد.فرض كنيد كه يك تگ <SQL> مخصوص گرفتن محتوا با query از بانك اطلاعاتي بوده و نتيجه به جدولي در صفحه html مي فرسد. در عمل اين كار شبيه به استفاده از تگ <SERVLET> است.
پشتيباني از دادهاي كه كمتر مورد استفاده اند
براي مثال شما مي توانيد از تصاويري كه فرمت آنها كمتر پشتيباني مي شود به كمك يك فيلتر آنها را به انواع استاندارد مثل GIF,JPEG تبديل كنيد.

ممكن است شما از خود سوال كنيد كه چرا از زنجيره سازي استفاده كنيد وقتي كه مي توانيد به جاي آن از script براي ويرايش فايلها در مكان ويژه استفاده كنيد و Servlet به خاطر گرفتن درخواست يك سربار اضافي دارد.
در زير سه مزيت زنجيره سازي بيان شده:

- زنجيره سازي Servlet به راحتي قابل برگشت پذيري دارد زماني كه كاربران معترض هستند به استبداد شما در گذاشتن <BLINK> ، و شما مي توانيد سريعآ تغييرات را برگردانيد و قشقرق را آرام كنيد.
- اين امكات وجود دارد كه بطور اتوماتيك محتوا را ايجاد كرده و اطمينان پيدا كنيد كه محدوديت سازي برقرار است. بطور مثال تگها ي ويژه جايگذاري شده و تصاوير ارسالي PostScript به تصوير هاي معمولي تبديل مي شوند.
- آنها محتواي براي آينده را ايجاد مي كنند و نياز نيست هر بار براي اضافه كردن محتواي جديد script اجرا كنيد.

ايجاد يك Servlet Chain

اين مثال تمام تگ هاي <BLINK> را از صفحه html برميدارد. اگر شما با اين تگ آشناي نداريد بايد بگويم كه وقتي متني در بين اين تگ قرار ميگيرد اين متن بصورت درخشان و چشمك زن نشان داده مي شود.مطمعن باشيد كه استفاده از آن مفيد است، امتحان كنيد. اين مثله در بسياري از نوشته ها گفته شده و يكي از جوكها در مورد html است.
در مثال زير با استفاده از زنجيره سازي تمام تگهاي <BLINK> را از صفحات ايستاي سايت خذف مي كند. تمامي اين كارها بطور اتوماتيك است وصفحاتي كه در آينده اضافه مي شوند را نيز در بر مي گيرد. اين Servlet متدهاي getReader(),getContenType() را كامل ميكند.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Deblink extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
```

```
  throws ServletException, IOException {
  String contentType = req.getContentType(); // get the incoming type
  if (contentType == null) return; // nothing incoming, nothing to do
  res.setContentType(contentType); // set outgoing type to be incoming type
  PrintWriter out = res.getWriter();
  BufferedReader in = req.getReader();
  String line = null;
  while ((line = in.readLine()) != null) {
  line = replace(line, "<BLINK>", "");
  line = replace(line, "</BLINK>", "");
  out.println(line);
  }
  }
  public void doPost(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException {
  doGet(req, res);
  }
  private String replace(String line, String oldString, String newString) {
  int index = 0;
  while ((index = line.indexOf(oldString, index)) >= 0) {
// Replace the old string with the new string (inefficiently)
  line = line.substring(0, index) +
  newString +
  line.substring(index + oldString.length());
  index += newString.length();
  }
  return line;
  }
}
```

در اينجا از دو متد doget(),doPost() براي گرفتن هر دو نوع درخواست استفاده كرده ايم. در داخل متد doGet() ابتدا يك print writer گرفته سپس نوع محتواي دادهاي دريافت شده با متد getContentType() مشخص مي شود و از آن براي تنظيم كردن محتواي خروجي استفاده ميشود.
اگر متد getContentType() مقدار null برگرداند اين به اين معني است كه داده اي براي deblink كردن نرسيده و فقط return مي نويسيم. بمنظور خوانده داده از درخواست رسيده از شي BufferedReader با فراخواندن متد req.getReader() كمك مي گيريم.
Servlet صفحه html خروجي از Servlet قبلي از زنجير را خط به خط خوانده و تمام <BLINK> يا </BLINK>با متد replace() برداشته و آنرا به سرويس گيرنده برمي گرداند(يا به Servlet بعدي در زنجير ارسال مي كند).
ورژن قوي تر از اين Servlet بايد سرآيند HTTP رسيده را به همان شكل به سرويس گيرنده ارسال كند.

اجرا كردن Deblink Servlet

اگر از وب سرور جاوا استفاده مي كند در قدم اول بايد سرويس دهنده را از زنجيره اي بودن اين Servlet آگاه سازيد .براي اين كار به قسمت مديريت Web Service رفته و setup را انتخاب كنيد سپس site را انتخاب كنيد ودر آخر در قسمت option گزينه Servlet Chainnig را فعال كنيد.

همان طور كه قبلاً گفتيم كه دو راه براي واكشي Servlet زنجيره اي وجود دارد. يك زنجير مي تواند بطور ويژه براي درخواست واقعي از طرف سرويس گيرنده تعريف شود يا درخواست ايجاد شده از طرف يك Servlet ديگر.ما از هر دو تكنولوژي براي Deblink استفاده مي كنيم
ابتدا ما مي خواهيم به طور صريح تمام فايلها با الگوي *.htm را مشخص كنيم و تطابق نام آنها براي ارسال شدن به file Servlet كه از Deblink Servlet پيروي مي كند. file Servlet از Servlet هاي هسته سرويس دهنده وب جاوا مي باشد كه براي بازيابي فايلها استفاده ميشود و در حالت عادي فايلهاي html را بر مي گرداند.قبل از اينكه فايل html به سرويس گيرنده ارسال شود آنرا توسط file Servlet به Deblink Servlet ارسال مي كنيم.
به عقب برگرديم در مديريت Web Service به قسمت setup رفته و Servlet Aliases را انتخاب كنيد. در ايجا خواهيد ديد كه كدام Servlet ها با نامي جدا از نام خود فراخواني خواهند شد(در شكل زير).

اين نگاشت همان ديد را دارد كه سرويس دهنده وب جاوا براي Servlet هاي هسته خود دارد.مثلاً براي *.shtml ، ssinclude Servlet و براي /Servlet ، invoker Servlet را فراخواني مي كند. Invoker Servlet وظيفه پيدا كردن Servlet مورد نظر و فرخواني آن را دارد. حال ما در Alias *.html نام Deblink را قرار ميدهيم و پس از تغيير تمام فايها با پسوند html. زماني كه درخواست شدند ابتدا file Servlet آنها را بازيابي كرده و به Deblink Servlet راسال ميكند.
حال يك فايل به نام blinky.html در server_root/public_html ايجاد كرده و در داخل آن از تگ <BLINK> استفاده كنيد. سپس با URL http://server:8080/blinky.html آنرا اجرا كنيد و اگر همه چيز درست باشد تمام تگهاي <BLINK> حذف شده ميباشند.

Loophole

این تکنیك که راه گریز نام دارد و این طور استنباط میشود که تمام کدهاي HTML از فایلها با پسوند html. مي آیند و این استنباط را تغییر مي دهد. کدهاي HTML مي توانند از فایلهاي مختلف کار کنیم.پس ما نیاز به سومین تکنیك براي زنجیره کردن نیاز داریم که یك در اصل یك مدخل ایجاد میکند.
قصد ما این است که تمام فایلهاي text/html قادر به ارسال شدن به Deblink Servlet باشند. Administration Tool سرویس دهنده جاوا امکان این کار را به صورت گرافیکي ندارد وباید یك فایل Properties را ویرایش کنیم.این فایل در server_root/properties/server/javawebserver/mimeservlets.properties پیدا میشود.
این فایل شامل رهنمود به فرم زیر است.

Java-internal/parsed-html=ssinclude

این رهنمود به این اشاره دارد که تمام پاسخها با سرآیند Content-type از Java-internal/parsed-html به ssinclude(server side include) Servlet ارسال خواهد شد.
براي چه این کار لازم خواهد بود؟ بدون آن ssinclude Servlet فقط فایلها ایستا با پسوند shtml را پشتیباني خواهد کرد.
این همان راه گریز است که به طور خودکار صفحات شامل تگ <SERVLET> ایجاد کرده و پشتیباني مي کند.
براي مشخص کردن ارسال محتوا بصورت text/html به Deblink باید رهنمود زیر را اضافه کنیم.

Text/html=Deblink

بعد از اعمال تغییرات باید سرویس دهنده را دوباره راه اندازي کرده و HelloWorld Servlet با تگ <BLINK> امتحان نماید.

Java Server Page

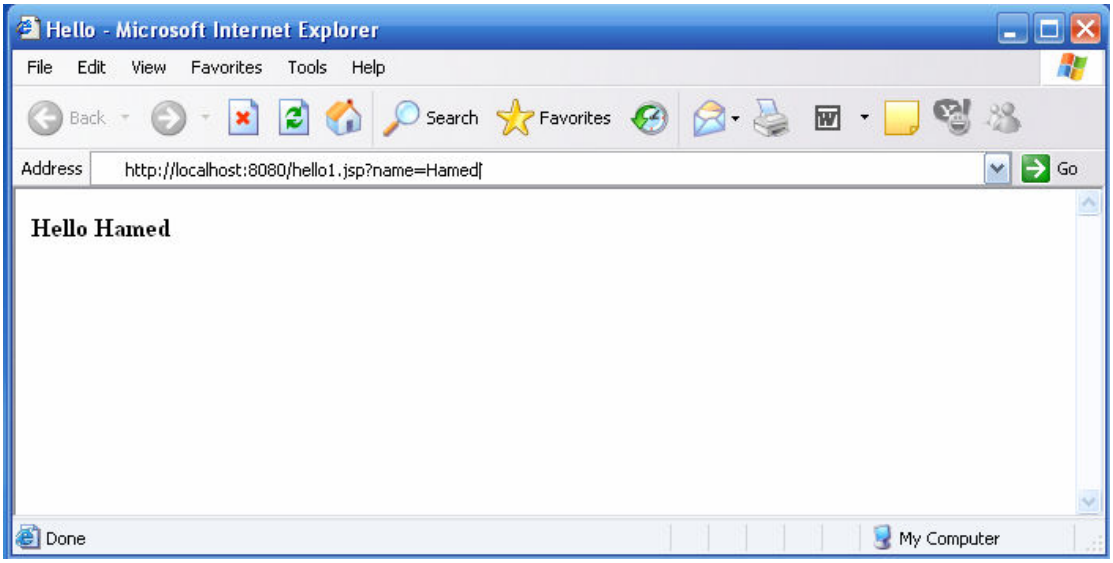
بعد از ظهور Servlet شرکت sun یك راه جدید براي استفاده از Servlet با نام Java Server Page(JSP) ارائه داد و تواناي ها و قابیتهاي همانند شبه به Active Server Page(ASP) شرکت میکروسافت دارد.
JSP از بسیاري لحاظ شبه به server side include عمل کرده و تفاوت اصلي این است که به جاي جاي دادن تگ <SERVLET> در صفحه html آنرا به طور جزئي در کد Servlet قرار مي دهیم.
همانند زنجیره سازي و .
JSP همانند زنجیره سازي و server side include نیازي به به تغیر Servlet API ندارد، اما نیاز به پشتیباني ویژه از طرف سرویس دهنده دارد و این قابلیت در سرویس دهندهاي ورژن 1.1.1 به بعد موجود است.

استفاده از JSP

تکنولوزي JSP به ما این امکان را مي دهد که کد هي Servlet را در داخل یك صفحه ایستا HTML قرار دهیم. هر بلوك از کد Servlet در تگ <% %> محصور شده(با نام scriptlet شناساي میشوند) و براي آسوده گي از چهار متغییر از قبل تعریف شده استفاده میکند.
Request
یك Servlet Request که از نوع شي HttpServletRequset مي باشد.
Response
یك Servlet Response که از نوع شي HttpServletResponset مي باشد.
Out

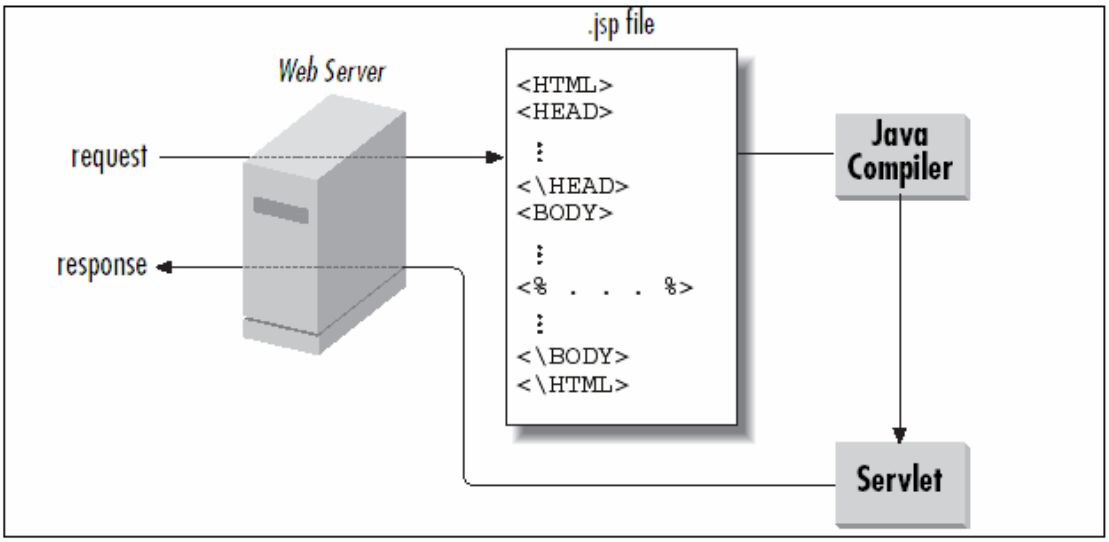
يك output writer كه از نوع شي PrintWriter مي باشد.
in
يك input writer كه از نوع شي BufferedReader مي باشد.
در مثال بعدي يك صفحه JSP كه پيغام Hello را نشان مي دهد با كمي تغيير نسبت به كد هاي قبلي كامل كرده ايم. لازم به ذكر است كه متغيير هاي Request,out از قبل تعريف شده مي باشند.
براي اجراي آنرا با نام hello1 و با پسوند jsp در فولدر ريشه سرور يعني server_root/public_html ذخيره كنيد. براي دستيابي به اين صفحه آدرس http://server:port/hello1.jsp را وارد كنيد. نحوه اجرا اين صفحه در شكل نشان داده شده.

```
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
<%
if (request.getParameter("name") == null) {
out.println("Hello World");
}
else {
out.println("Hello, " + request.getParameter("name"));
}
%>
</H1>
</BODY>
</HTML>
```



بقيه كار

نحوه كار JSP به اين شكل مي باشد كه سرويس دهنده به طور خود كار Servlet براي ساختن محتوا ايجاد، كامپايل، لود و اجرا مي كند. نحوه اين كار درشكل زير نشان داده شده.



اين طور مي توان فكر كرد كه يكServlet در پيش ضمينه اجرا ميشود و نام آن را Servelt زحمت كش گذاشته اند و صفحه html را با فراخواني متد out.println() كامل ميكند.
ممكن است Servlet زحمت كش براي hello1.jsp به شكل زير باشد.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class _hello1_xjsp extends HttpServlet {
public void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
BufferedReader in = request.getReader();
out.println("<HTML>");
out.println("<HEAD><TITLE>Hello</TITLE></HEAD>");
out.println("<BODY>");
out.println("<H1>");
if (request.getParameter("name") == null) {
out.println("Hello World");
}
else {
out.println("Hello, " + request.getParameter("name"));
}
out.println("</H1>");
out.println("</BODY></HTML>");
}
}
```

در هنگام دستيابي به صفحه JSP مدت زماني براي ايجاد پاسخ صرف مي شود و اين لازم است تا Servlet پيش ضمينه را ايجاد و كامپايل كند.به طور حتم در خواستهاي بعدي براي آن صفحه JSP داراي سرعت بيشتري خواهند بود چون سرويس دهنده فقط يك بار Servlet پيش ضمينه آن را كامپايل مي كند آن هم در بار اول. براي هر تغيير نياز به كامپايل دوباره داريم و هر گونه خطا در ضمن كامپايل به سرويس گيرنده ارسال خواهد شد.

عبارت و رهنمودها (expression ,directives)

در scriptlet داخل تگ <% %> مي توانيم از عبارت ها و رهنمودها استفاده كنيم. مثلآ به جاي استفاده از out.println() براي نوشتن در خروجي كافي است تگ را با علامت مساوي استفاده كنيم همانند <%= foo%> كه مقدار متغيير در خروجي ديده خواهد شد.
رهنمودهاي JSP به فرم <%@ %> بوده و اجازه مي هند كه Servlet پيش زمينه را كنترل كرده. مثلآ براي تنظيم كردن نوع محتوا ، اضافه كرده بسته هاي برنامه جاوا، تكميل كردن يك رابط، extend كردن يكsuper class وگرفتن درخواستهاي GET,POST .
در تگهاي رهنمود متغيير هاي كليدي به فرم زير مي توانند مقدار بگيرند.

```
<%@ varname = "value" %>
```

در رهنمودها سه متغيير وجود دارد كه شما مي توانيد به آنها مقدار دهيد.

content-type
نوع محتواي صفحه كلي را مشخص ميكند. بطور پيش فرز اين مقدار "text/html" مي باشد.

```
<%@ content-type = "text/plain" %>
```

import
ليست كلاسهاي را كه سرويس دهنده بايد آنها را به برنامه وارد كند واگر بيش از يك كلاس داشته باشيم مي توانيم آنها را با كاما از هم جدا كنيم.

```
<%@ import = "java.io.*,java.util.Hashtable" %>
```

extend
مشخص كننده ابر كلاسهاي كه Servlet بايد آنها extend كند.

```
<%@ extend = "CustomHttpServletSuperclass" %>
```
ابر كلاس پيش فرز HttpServlet ميباشد.

Implements
مشخص كننده ليست رابطهاي است كه Servlet بايد آنرا تكميل كند. رابطهاي چند تاي با كاما از هم جدا مي شوند.

```
<%@ implements = "Serializable" %>
```

method
مشخص كننده نوع متدي مي باشد كه سرويس گيرنده درخواست را با آن به Servlet ارسال مي كند.به طور پيش فرز همه متدها را پشتيباني مي كند.

```
<@ method = "doPost" %>
```

language
مشخص كننده زبان scripting است و به صورت پيش فرز آن java است. شما مي توانيد ساير زبانهاي مورد قبول را استفاده كنيد.

```
<%@ language = "java" %>
```

مثال زير صفحه Hello را با استفاده از عبارتها و رهنمودها در JSP پياده سازي كرده.

```
<%@ method = "doPost" %>
<HTML>
```

```
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
<% if (request.getParameter("name") == null) { %>
Hello World
<% } else { %>
Hello, <%= request.getParameter("name") %>
<% } %>
</H1>
</BODY>
</HTML>
```

Servlet پيش زمينه اين صفحه JSP به مثال قبلي بسيار نزديك است با اين تفاوت كه متد doPost() را به جاي service() كامل مي كند.

اعلان كردن(Declaration)

در بعضي از مواقع ضروري است كه در يك صفحه JSP متدها و متغييرهاي غير محلي را در Servlet پيش زمينه(زحمت كش) تعريف كنيم.به اين عمل ين عمل اعلان كردن در JSP مي گويند.
يك اعلان كردن با <SCRIPT RUNAT=" server"> شروع شده و با تگ </SCRIPT> خاتمه مي يابد. در ميان اين تگ شما مي توانيد كد هر Servlet را در آن ضمينه (include) كنيد كه در داخل آن از متدهاي آن Servlet استفاده كنيد. در مثال زير از متد getName() كه ايجاد شده استفاده مي شود.

```
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
Hello, <%= getName(request) %>
</H1>
</BODY>
</HTML>
<SCRIPT RUNAT="server">
private static final String DEFAULT_NAME = "World";
private String getName(HttpServletRequest req) {
String name = req.getParameter("name");
if (name == null)
return DEFAULT_NAME;
else
return name;
}
</SCRIPT>
```

Servlet پيش زمينه كه براي اين Servlet ايجاد مي شود، ممكن است به فرم زير باشد.

```
import java.io.*;
import javax.servlet.*;
```

```
import javax.servlet.http.*;
public class _hello3_xjsp extends HttpServlet {
  public void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
BufferedReader in = request.getReader();
out.println("<HTML>");
out.println("<HEAD><TITLE>Hello</TITLE></HEAD>");
out.println("<BODY>");
out.println("<H1>");
out.println("Hello, " + getName(request));
out.println("</H1>");
out.println("</BODY></HTML>");
}
private static final String DEFAULT_NAME = "World";
private String getName(HttpServletRequest req) {
String name = req.getParameter("name");
if (name == null)
return DEFAULT_NAME;
else
return name;
}
  }
```

كامپوننتهاي JavaBeans و JSP

يكي از مزيتهاي و قدرتهاي JSP در اين است كه ميتوانيم در آنها از كامپوننتهاي جاوا يعني JavaBeans استفاده كنيم. اين كامپوننتها كلاسهاي جاوا مي باشند كه قابليت استفاده مجدد داشته و متدها ومتغير هاي آنها به كمك نام ويژه شان قابل دسترس است. به كمك تگ <BEAN> مي توان آنها را به طور مستقيم در صفحه JSP استفاده كرد.يك JavaBeans وظايف خوش تعريف مثل اجرا پرس جو از بانك اطلاعاتي، اتصال به سرويس دهنده mail، مديريت اطلاعات درباره سرويس گيرنده و ساير را مي تواند انجام دهد و نتايج را براي صفحه JSP با متدهاي دسترسي قابل دسترس مي كند.
تفاوت ميان JavaBeans گذاشته شده در داخل يك صفحه JSP و كلاس معمولي Servlet اين است كه سرويس دهنده وب با JavaBeans رفتار متفاوتي دارد. براي مثال سرويس دهنده به طور خود كار متغير هاي نمونه كامپوننت(propertyها) را با پارامتر رسيده از سرويس گيرنده تنظيم ميكند(مقدار را در داخل آنها قرار مي دهد).مثلاً متغير name از JavaBeans كه با متد setName(String name) تنظيم مي شود و سرويس دهنده با فراخواني اين متد آنرا تنظيم مي كند. توجه كنيد كه نياز به متد getParameter() نداريد.
همچنين يك bean مي تواند توسط سرويس دهنده به طور خودكار مديريت شود و براي درخواستهاي ويژه از سرويس گيرنده تخصيص يابد. براي اين كار بايد از جلسه كاري سرويس گيرنده يا Session اسفاده كرد يعني زماني كه سرويس گيرنده متصل مي شود جلسه كاري شروع مي شود و هنگامي كه سرويس گيرنده قصد قطع اتصال را دارد Session پايان مي پذيرد.
يك bean حتي مي تواند در داخل يك Servlet استفاده شود! اگر سرويس دهنده implement يك bean را در داخل رابط javax.servlet.Servlet پيدا كند آنگاه متد service() از bean را براي

هر دخواست سرويس گيرنده فراخواني خواهد كرد، همچنين متد ()init از bean را وقتي كه آن براي اولين بار ايجاد مي شود فراخواني مي كند. سودمندي اين قابليت قابل بحث است اما مي تواند توسط bean براي بررسي چگونه گي آن پيش از گرفتن درخواست سرويس گيرنده استفاده گردد.
فرم استفاده از bean بايد به شكل زير باشد.

<BEAN NAME="*lookup name*" VARNAME="*alternate variable name*"
TYPE="*class or interface name*" INTROSPECT="{yes|no}" BEANNAME="*file name*"
CREATE="{yes|no}" SCOPE="{request|session}">
<PARAM *property1=value1 property2=value2*>
</BEAN>

صفات در تگ <BEAN> به صورت زير تنظيم مي شوند.

NAME

براي مشخص كردن نام bean مورد استفاده در محدوده درخواستهاي سرويس گيرنده و اگر bean با اين نام موجود باشد در اين صفحه استفاده خواهد شد.

VARNAME

مشخص كننده نام متغيير هاي Bean است. صفحه از اين نامها براي ارجاع به bean و فراخواني متد هاي مربوط به property استفاده مي كند.

VARNAME="prefs"

TYPE

نام كلاس bean يا نوع رابط را مشخص مي كند براي مثال:

TYPE="UserPreferenceesBean"

به طور پيش فرز اين مقدار java.lang.Object مي باشد.

INTROSPECT

اگر سرويس دهنده بخواهد ويژه گيهاي bean را با استفاده پارامترهاي ارسال شده از سرويس گيرنده تنظيم كند اين مقدار را بايد برابر با "yes" قرار دهيم و درغير اين صورت با "no" اين كار را انجام مي دهيم.

BEANNAME

به منظور مشخص كردن فايل سريال شده يا فايل كلاس كه شامل bean مورد نظر است و زماني كه براي اولين بار bean ايجاد مي شود سرويس دهنده از آن استفاده مي كند. اين صفت بصورت اختياري مي باشد.

BEANNAME="hellobean.ser"

CREATE

اگر bean هنگام درخواست سرويس گيرنده موجود نباشد آنرا ايجاد كرده و دو مقدار "yes" , "no" را قبول مي كند. اگر مقدار "no" را داشته باشد و نمونه از قبل ايجاد شده در سرور موجود نباشد به سرويس گيرنده پيغام خطا را ارسال خواهد كرد. بطور پيش فرز مقدار "yes" را در خود دارد يعني به طور خودكار bean را ايجاد خواهد كرد.

SCOPE

براي مشخص كردن تخصيص bean ها به درخواستهاي ويژه يا جلسه كار سرويس گيرنده(به طور خودكار تا زماني قابل دسترس باشد كه سرويس گيرنده به سرور متصل است).مقدار هاي قابل قبول دو مقدار request , session مي باشند كه پيش فرز request مي باشد.

پارمترهاي كه به يك bean پاس مي شوند با ليستي از تگهاي <PARAM> در داخل تگ <BEAN> مشخص ميشوند.
مثال زير كه با استفاده از HelloBean پيغام hello را به كاربر مي گويد نحوه استفاده از يك bean در صفحه JSP را نشان مي دهد.

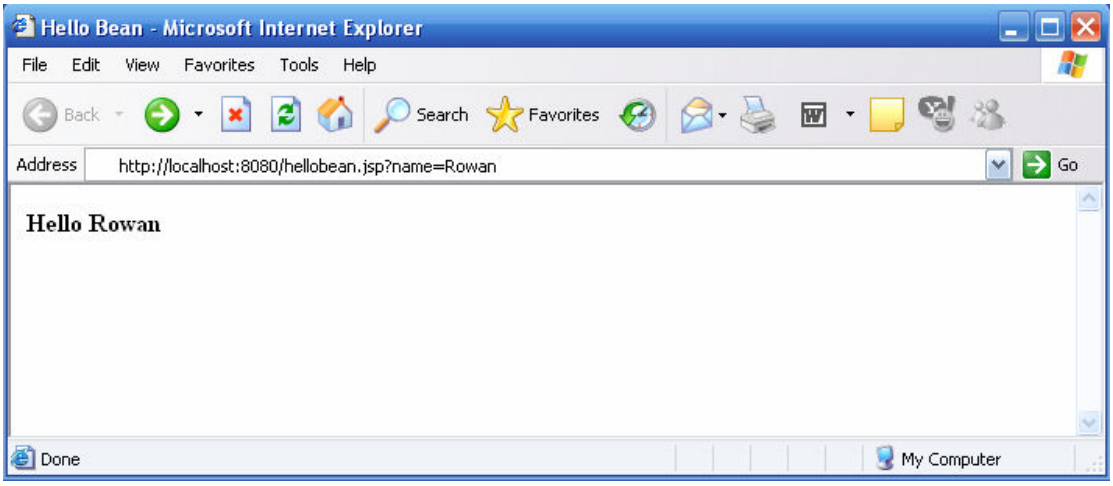
```
<%@ import = "HelloBean" %>
<BEAN NAME="hello" TYPE="HelloBean"
INTROSPECT="yes" CREATE="yes" SCOPE="request">
</BEAN>
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<H1>
Hello, <%= hello.getName() %>
</H1>
</BODY>
</HTML>
```

مشاهده مي كنيد كه استفاده از bean كد را در داخل صفحه JSP كد هاي را كاهش داده و باعث مي شوند كه كد داخل JSP وضوح وقدرت(كامپونتها قدرت بيشتري در ارتباط با ساير برنامه ها دارند) بيشتري داشته باشد.حتي استفاده از API هاي خوش تعريف كه با bean تعامل دارند باعث مي شود كه افرادي كه برنامه نويس حرفه اي نيستند صفحات JSP پيشرفته اي طراحي كنند.
HelloBean كه بايد در مسير كلاسهاي سرويس دهنده قرار گيرد همانند:
server_root/classes

در زير كد HelloBean كامل شده.

```
public class HelloBean {
private String name = "World";
public void setName(String name) {
this.name = name;
}
public String getName() {
return name;
}
}
```

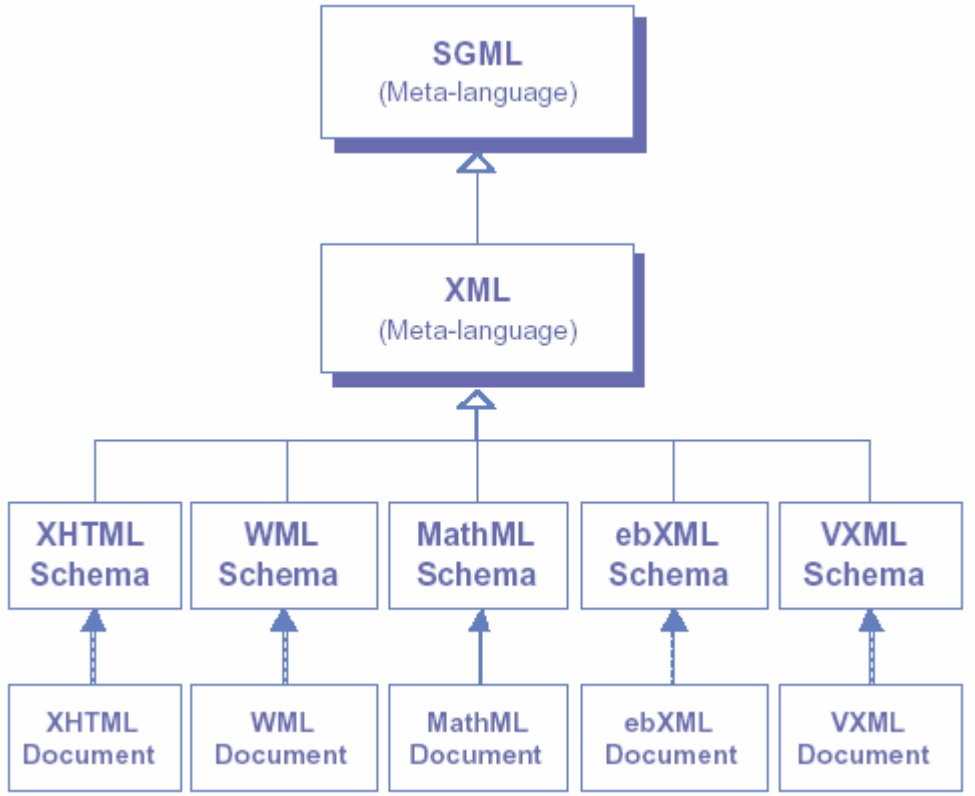
اين bean يك property به نام name داشته كه با متد setName() مقدار مي گيرد و با متد getName() مقدار داخل خود را برمي گرداند. به طور پيش فرز مقدار آن "world" است. براي تست كردن آدرس http://server:port/hellobean.jsp را در مرورگر خود وارد كنيد.


Hello Bean - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/hellobean.jsp?name=Rowan
Go
Hello Rowan
Done
My Computer

فصل ششم

Java XML

زبان eXtensible Markup Language(XML) در عمل يك زبان نمي باشد و يك فرازبان(matalanguage) براي ايجاد ساير زبانها ميباشد. از Xml براي ايجاد ساختار ساير زبانها و مستندات خود تعريف استفاده مي شود. Xml در جنبه اي گوناگون تجاري وصنعتي استانداردهاي فراواني پيدا كرده، همانند Mathematicla Markup Language(MathML) ، Electronic Business XML(ebXML) ، Voice Markup Language(VXML) و ساير نظاير آنها. اين مفاهيم در شكل زير نمايان ميباشد.



Xml شامل نشانه ها(markup)و محتوا مي باشد. نشانه ها شامل يك سري تگها كه محتواي داخل مستندات را شرح ميدهد ميباشند.يك شرح قابل تطابق از داده ها به ما اين امكان را مي دهند كه به راحتي داده را ارسال و در يافت كرده و آنها را به ساير فرمتها تبديل كنيم. به طور مثال صاحبان تجارت و صنايع از xml براي تعويض اطلاعاتي مثل قيمتها، داراي ها، محصولات با ساير شركتها استفاده مي كنند.
استفاده از xmlبسيار گسترده شده به طوري كه رتبه اول را در جابه جاي اطلاعات در اينترنت دارد.

هر سند xml براي خود يك گرامر مخصوص به خود دارد كه مشخص كننده مجموعه قوانين و سازمان آن سند است. بطور مثال يك عنصر به نام قيمت در مستندات ebXML داراي معني بوده اما در مستندات MathXML داراي معني نمي باشد. پس تمام مستندات xml بايد اين گرامر را براي خوانده و ايجاد شدن داشته باشند. xml قابليتهاي براي ايجاد اين گرامرها و تطابق با مستندات رسيده در خود ايجاد كرده.
Xml در خود يك پايه براي زبانهاي سطح بالاتر براي تعويض داده ها در بين كامپوننتهاي نرم افزاري، سيستمهاي enterprise دارد.
امروزه اكثر زبانهاي برنامه نويسي امكاناتي براي تجزيه و ترجمه كردن مستندات xml و ابزار هاي براي ايجاد ،دستكاري انواع اطلاعات بر پايه xml را در خود تعبيه كرده اند.
سازمان دهي بر اساس xml امكان سطح استفاده مجدد بالاي براي كامپوننتها در سيستم هاي توزيع شده فراهم ميكند. براي مثال در جاوا تمام كامپوننتها Deployment Discriptor با فرمت xml براي نصب در يك سرويس دهنده يا برنامه دارند و اين كمك ميكند كه در سيستمهاي مختلف به راحتي از آنها استفاده كرد. اين مستندات مشخصات و ويژه گي هاي كامپوننت را براي استفاده در خود دارند.
چون xml به عنوان يك استاندارد تبادل در صنعت كامپيوتر معرفي شده در بيشتر مواقع تبادل اطلاعات وپيغام رساني از آن استفاده مي كند و اين زماني مهم است كه سيستمهاي كه با هم تفاوت ساختار دارند.

از xml ميتوان به عنوان يك ذخيره كننده داده هاي ماندگار نيز استفاده كرد به طور كه به جاي گذاشتن آنها در بانك هاي اطلاعاتي آنها را در فيالهاي xml قرار ميدهيم.
امروزه اكثر بانكهاي اطلاعاتي امكان import , export داده ها را به صورت xml دارند.

يك سند xml نمونه

براي تشريح بهتر xml و تكنولوژي هاي وابسته يك مثال عملي را ايجاد خواهيم كرد. از اين مثال براي توضيح تكنولوزي هاي متفاوت xml استفاده خواهيم كرد.
در زير سند xml كه براي مشخص كردن كاتالوگ محصولات است را مشاهده ميكنيد.

```
<?xml version="1.0"?>
<product-catalog>
  <product sku="123456" name="The Product">
    <description locale="en_US">
      An excellent product.
    </description>
    <description locale="es_MX">
      Un producto excellente.
    </description>
```

```
    <price locale="en_US" unit="USD">
      99.95
    </price>
    <price locale="es_MX" unit="MXP">
      9999.95
    </price>
  </product>
</product-catalog>
```

در بلا يك كاتالوگ محصول را مشاهده ميكنيد كه فقط يك محصول را نشان مي دهد. اين محصول شامل اطلاعات شرح، قيمت، و شماره SKU است. در اين جا توجه كنيد كه شرح وقيمت اين محصول براي مكانهاي متفاوت جدا مي باشد، مثلآ براي كشور مكزيك قيمت 9999.95 MXP به واحد پول اين كشور است.

رده بندي تكنولوژي هاي XML

استانداردها و تكنولوژي هاي متفاوتي براي xml تا كنون به وجود آمده اند كه هر يك كاربرد خاص خود را دارد. اينها مخصوص جاوا يا زباني خاص نيستند وبراي استفاده كردن ساده و مديرت شده xml ساخته شده اند.
در اين بخش يك مرور خلاصه بر برخي از آنها خواهيم داشت.براي اطلاعات بيشتر در اين زمينه از سايت http://www.zvon.org مي توانيد آنها را در يافت كنيد.

تكنولوژي اعتبار سنجي XML
نتايج يك سند xml به دو روش بازيابي مي شوند. يك روش با استفاده از شماي xml (xml schema) و روش ديگر استفاده از تعريف كننده نوع سند(Document Type Definition). هر كدام از دو روش مشخص كننده قوانين آن سند xml بوده و براي بازيابي از آنها استفاده مي شود و بار سنگيني را از دوش كد هاي برنامه شما برمي دارد.

تعريف كننده نوع سند(Document Type Definition)

اولين وقديمي ترين مكانيزم تعريف زبان Document Type Definition يا به اختصار DTD مي باشد. DTD يك فايل متني شامل مجموعه قوانين در باره ساختار و محتواي يك سند xml بوده و مجموعه عناصري معتبر شامل مشخصات و ترتيب را كه ممكن است در در سند xml ظاهر شود ليست مي كند.
يك DTD سلسله مراتب ساختاري يك سند را ديكته مي كند، آنهاي كه به شدت مهم مي باشند براي اعتبار سنجي ساختار xml. براي مثال عنصر نيمكت يا Couch ممكن است در داخل عنصر LivingRoom معتبر باشد، اما در عنصر BathRoom معتبر نباشد. DTD همچنين مشخصات عناصر مثل عناصر خيلي ويژه ، مقادير قابل قبول و اختياري يا اجباري را تعريف مي كند.

در زير DTD مربوط به مثال كاتالوگ محصولات را مشاهده مي كنيد.

```
<!ELEMENT product-catalog (product+)>
<!ELEMENT product (description+, price+)>
  <!ATTLIST product
   sku ID #REQUIRED
```

```
    name CDATA #REQUIRED  >
<!ELEMENT description (#PCDATA)>
 <!ATTLIST description
    locale CDATA #REQUIRED
  >
<!ELEMENT price (#PCDATA)>
  <!ATTLIST price
    locale CDATA #REQUIRED
    unit CDATA #REQUIRED
  >
```

اين DTD توسط پارسر xml براي اعتبار سنجي سند استفاده مي شود.، ما ميتوانيم يك DTD را در يك فايل جدا ذخيره كرده و اشاره گر به آن را در خود سند xml شبيه به زير ذخيره كنيم.

```
<!DOCTYPE product-catalog SYSTEM "product-catalog.dtd">
```

استفاده از اين دستور پارسر را وادار مي كند كه "product-catalog.dtd" رادر همان فهرست فايل كه سند xml وجود دارد جستجو كرده و از آن استفاده كند.

تعريف كننده الگو(schema)

اگر چه اولين مشخص كننده اعتبار سند DTD بود ولي در توسعه هاي enterprise محدوديتهاي را با خود به همراه داشت .يكي از اين محدوديتها اصلي اين بود كه معتبر بودن خود DTD معلوم نبوده و بنا براين نياز به ابزار هاي براي تجزيه كردن آن بود. يك عيب ديگر اين بود كه DTD ها محدوديت زايدي در ايجاد ساختار ومحتواي مستندات xml داشتند. آنها قادر به حل تصادف نامها در فضاي نامي و رابطه بين عناصر پيچيده نبوده و خاصيت پيمانه اي بودن را پشتيباني نمي كردند. بنابر اين كنسرسيوم W3C تصميم به ايجاد تكنولوژي جايگزين براي آن در نظر گرفت. اين تكنولوژي XML Schema Definition(XSD) نام گرفت.
XSD خود بر پايه xml بوده و استفاده از آن رابطه بين عناصر و فضاي نامي عناصر را پشتيباني ميكند. XSD مشخصات كاملتري از سند xml نسبت به DTD ارائه ميكند. در XSD امكان به ارث بردن ويژه گي ها براي داده ها با نوع پيچيده وجود دارد. هم اكنون بسياري از انواع داده هاي اوليه به صورت خودكار در XSD موجود مي باشند.

در قسمت زير الگوي XSD مربوط به مثال product catalog را مشاهده ميكنيد.

```
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element type="product-catalog"/>

<xsd:complexType name="productCatalog">
  <xsd:element type="productType"
     minOccurs="1"/>
  </xsd:complexType>
```

```
<xsd:complexType name="productType">
  <xsd:element name="description"
      type="xsd:string" minOccurs="1">
    <xsd:attribute name="locale"
      type="xsd:string"/>
   </xsd:element>
   <xsd:element name="price"
      type="xsd:decimal" minOccurs="1">
    <xsd:attribute name="locale"
      type="xsd:string"/>
    <xsd:attribute name="unit"
      type="xsd:string"/>
   </xsd:element>
  <xsd:attribute name="sku"
      type="xsd:decimal"/>
  <xsd:attribute name="name"
      type="xsd:string"/>
 </xsd:complexType>
</xsd:schema>
```

این XSD يك نوع داده پيچيده به نام productType را كه بالاتر از ديگر نوع داده ها قرار دارد تعريف مي كند. داده ها با نوع پيچيده شامل مشخصات و ديگر عناصر از آن داده مي باشند.اين مثال اشكارا سطح ساختار پيچيده را كمتر مي كند تا شما بتوانيد با XSD آنها را تعريف كنيد.
همان طور كه مي بينيد تمام عناصر نام، نوع و حداقل مورد نياز آنها مشخص گرديده و يكي از قدرت XSD تعين نوع و محدوده ورودي است.در اينجا به شما توصيه مي شود كه در تعريف كردن XSD ها ماهر شويد، چون بعدها از آنها در برنامه هاي خود استفاده خواهيد كرد.
اين نكته را بايد بيان كرد كه برخي تجزيه كننده ها(پارسرها) در همه مواقع براي اعتبار سنجي پيشنهاد نمي شوند. اعتبار سنجي مستندات xml در طي توسعه و تست كردن بسيار با اهميت مي باشد و وقتي كه داده ها بين سيستم هاي enterprise به اشتراك گذاشته مي شوند اين امر حياطي مي شود. از اين اطمينان پيدا كنيد كه ارسال و دريافت دادها در فرمت معتبر صورت گيرد.

تكنولوژي تجزيه كردن XML

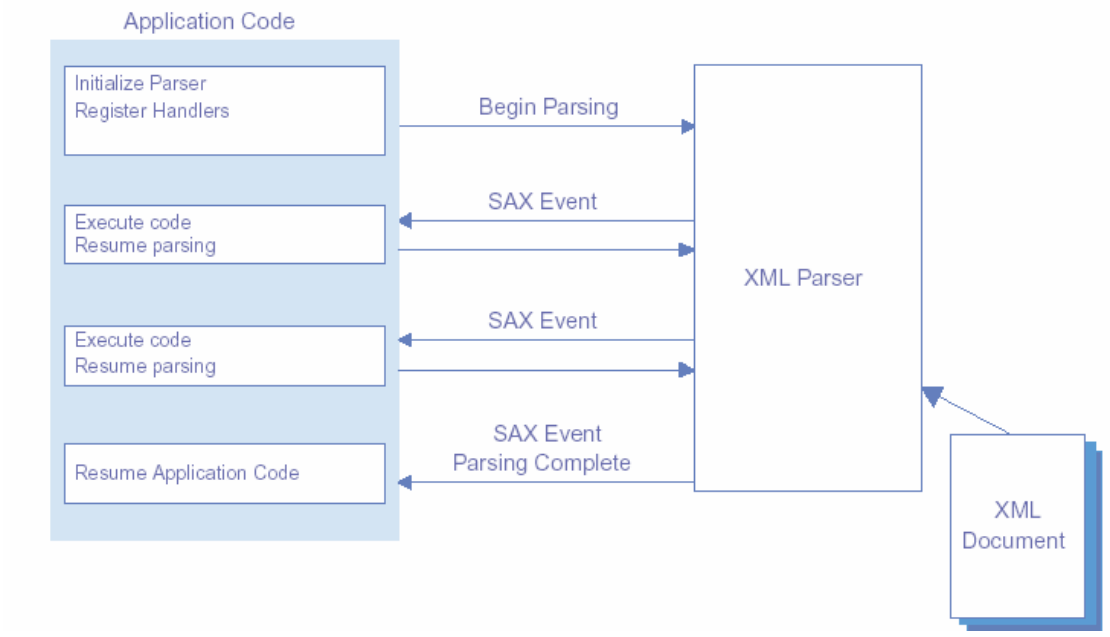
قبل از اينكه يك سند اعتبار سنجي واستفاده شود آن بايد با يكي از نرم افزار هاي پارسر تجزيه شود.اين پارسرها كامپونت نرم افزاري هستند كه قابليت خواندن مستندات xml را داشته و داده داخل اين سند را براي برنامه كاربردي قابل دسترس ميكنند.

پارسرهاي مختلفي تا كنون توسعه داده شده اند مثل Crimson و Xerces از Apache Software Foundation و پارسرهاي از Oracle وIBM .شما براي اطلاعات بيشتر درباره اين پارسرها مي توانيد از سايت http://xml.apache.org اطلاعات بگيريد. اين دو ابزار به صورت open source بوده و به صورت گسترده اي استفاده ميشوند.

SAX

اغلب پارسر ها در دو مد مختلف مي توانند استفاده شوند و بر پايه نيازهاي برنامه شما مي باشد. مدل اول مدل رخداد گرا كه آنرا با Simple API for XML(SAX) مي ناميم. اين پارسر داده منبع xml راخوانده و برنامه سرويس گيرنده را اگر در يك قسمت مشخص از سند با اشكال مواجه شود از آن مطلع مي كند. براي مثال يك رخداد SAX اين است كه وقتي انتهاي سند xml مشخص نباشد يا يك تگ بسته نشده باشد.

براي استفاده از SAX مي بايست دستگر رخداد(event handler) را براي پارسر در زماني كد سند xml را پارس ميكند تكميل كنيد. اين دستگير رخداد اغلب اوقات يك ماشين حالت(يك نمودار كه فقط با گرفتن ورودي هاي خاص مراحل بعدي را شروع مي كند) است كه داده ها را جمع كرده و همچنين در شروع پارس كردن مجموع دادهاي زير مستندات(subdocument) را به صورت مستقل از ديگري دستگير و كنترل مي كند. استفاده از SAX در شكل زير به نمايش در آمده كه در آن از متدهاي سريع پارس كردن و مختص براي مستندات بزرگي كه در يك زمان به حافظه كپي نمي شوند مي باشد.

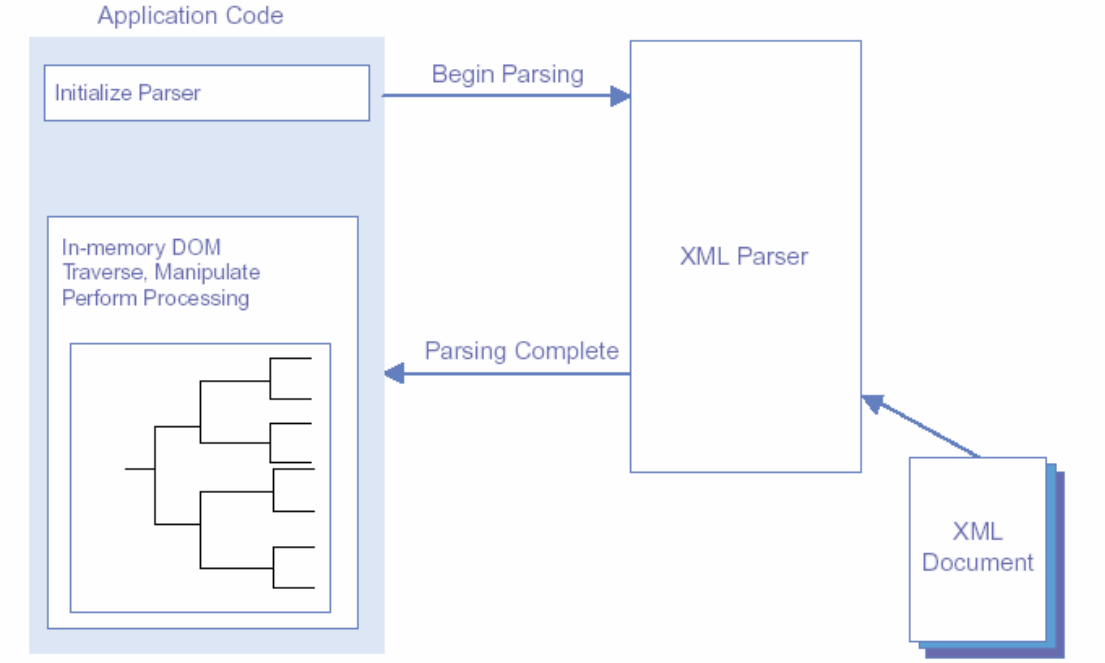


يك از اشكالهاي SAX اين است كه قادر به جستجوي رو به جلو در هنگام پارس كردن اطلاعات نمي باشد. دستگير SAX همانند يك ماشين حالت است كه مي تواند فقط بروي يك قسمت از سند كه هماكنون پارس شده كار كند. عيب ديگر نقص در پيش بيني رابطه بين گره هاي داخل سند است.

DOM

مدل ديگر پارس كردن مستندات xml به جاي SAX مدل Document Object Model(DOM) مي باشد. پارسر به طور كامل سورس سند xml را خوانده و يك ارائه شبيه به ساختار درختي را در حافظه مي سازد.درهنگام استفاده DOM يك اشاره گر

به کل سند به نرم افزار استفاده کننده از آن برمي گرداند. برنامه کاربردي با اين اشاره گر سند xml را دستکاري ، چيدن دوباره گره ها ، اضافه و حذف کردن محتوا را انجام مي دهد. استفاده از DOMدر شکل زير نشان داده شده.

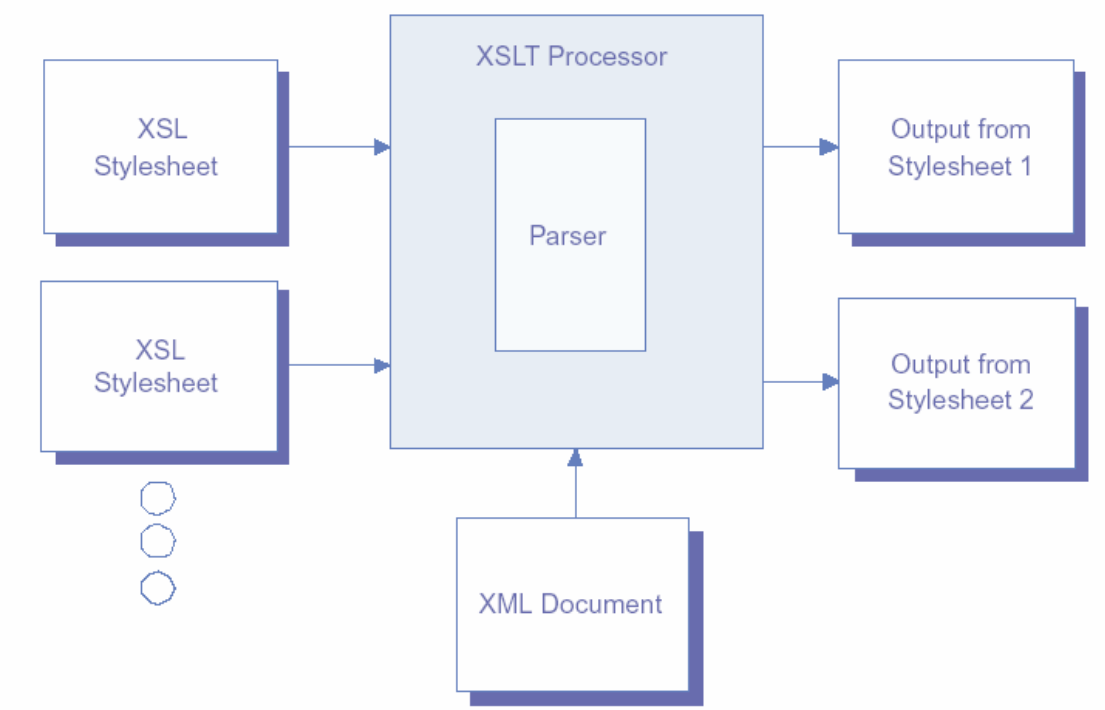


استفاده از DOM سادر تر از SAX مي باشد و منابع کمتري از آن مي خواهد. مي توان DOM را بصورت موثر با ساختارهاي داده xml کوچك در وضعيتي که سرعت براي برنامه کاربري اهمت زيادي ندارد استفاده کرد.به دليل اين که کل ساختار را در حافظه يکجا پارس مي کند سرعت کمتري دارد.
در تصميم گيري براي اينکه از کدام روش براي پارس کردن استفاده کنيد بايد به نياز هاي نرم افزار ، سرعت ، دستکاري داده ها واندازه سند xml توجه داشته باشيد.

تکنولوژي تبديل XML

يکي از مزيتهاي کليدي xml در مقايسه با ساير فرمتهاي داده که در معناي کلي قادر به برگرداندن يك مجموعه داده هاي xml از يك شکل به شکل ديگر مي باشد.
تکنولوژي اين ترجمه را قابل انجام مي کند eXtensible Stylesheet Language for Transformations(XSLT) نام دارد.

XSLT
در حالت ساده xslt يك چارچوب(framework) براي ترجمه ساختار سند xml مي باشد.xslt يك سند xml ورودي را با يك xslt stylesheet براي ايجاد سند خروجي ترکيب مي کند. Xslt stylesheet مجموعه دستورات تبديل براي برگرداندن يك سند xml به سند خروجي مورد نظر است.
شکل زير فرآيند xslt را نشان ميدهد.
اجراي تبديل به کمك xslt نيازمند يك پردازشگر xslt است. معروف ترين موتور xslt براي جاوا Apache Software Foundation Xalan مي باشد و به صورت open source در دسترس است.



يك پردازنده xslt يك درخت سند xml را با اختصاص الگو هاي داخل سند خروجي به كمك XSLt stylesheet تطابق داده و آنها را ترجمه مي كند. براي مثال فرض كنيد كه شما نياز به ترجمه سند product catalog xml به يك فايل html براي تحويل دادن كاربرد آن محصول داريد.

اين كار شامل گنجاندن داده هاي محصولات اختصاصي در سند xml داخل html است.

در زير xslt كه اين وظيفه را انجام مي دهد ملاحظه مي كنيد.

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:template match="/">
<html>
  <head><title>My Products</title></head>
  <body>
    <h1>Products Currently For Sale in the U.S.</h1>
  <xsl:for-each select="//product">
    <xsl:value-of select="@name"/> : $
    <xsl:value-of select="./price[@unit='USD']"/> USD
 </xsl:for-each>
  </body>
</html>
</xsl:template>
```

```
</xsl:stylesheet>
```

ويژه گي match در يك عبارت XPath به معني عنصر ريشه XML مي باشد. پس اين الگو براي اجرا در سند كامل است.در قسمت پائين كد، هر عنصر محصول در داخل سند منبع xml با مشخصاتش چاپ خواهد شد. ابتدا نام محصول سپس قيمت به همراه علامت دلار و در آخر رشته USD .

پردازنده xslt قادر به تغيير در اصطلاح ارزيابي مشخصات است و بيشتر به راه هاي پيش كامپايل براي ترجمه بر اشاره دارد. اين كار اجازه مي دهد كه موتور xslt روز به روز سريع تر شود.
جزئيات بيشتر درباره xslt در آدرس http://www.w3.org/Style/XSL موجود است.

تبديل دودوي براي XML

قابليتهاي xslt محدود به تبديلهاي متني نمي باشد. بعضي از مواقع لازم است كه داده هاي متني به شكل دودوي(binary) تبدل شوند. يك مثال معمولي تبديل داده هاي تجاري به فرمت PDF براي نمايش مي باشد.براي همين دليل XSLT 1.0 يك سري ويژه گيها براي تنظيم اشياء فرمت(formatting object) تعين كرده. اشياء فرمت دستوراتي هستند كه يك طرح بندي از ارائه اطلاعات مي دهند. اشياء فرمت در چاپ و طراحي چندرسانه اي بيشتر مفيد خواهند بود.
برخي كتابخانه هاي جاوا هم اكنون براي تبديل انواع معمولي در دسترس هستند.

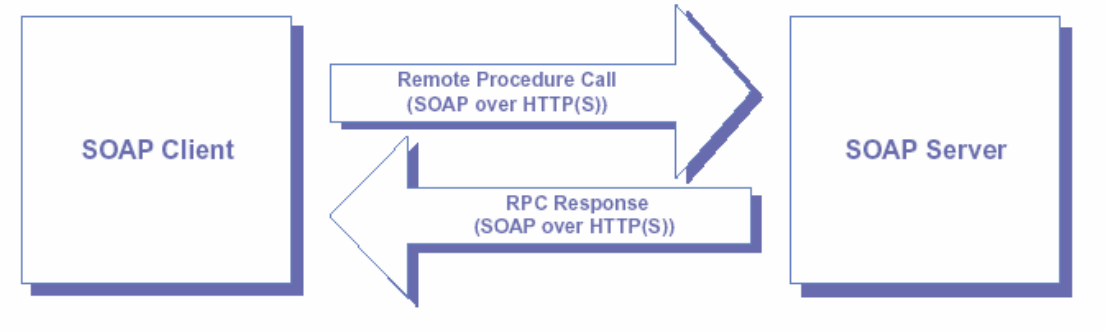
تكنولوژي هاي ارسال پيام

چندين تكنولوژي براي انتقال داده ها با ساختار xml بين برنامه ها و سيستمهاي enterprise هم اكنون در حال توسعه هستند. پتانسيل عظيم xml، آن را به عنوان پل در ميان فاصله بين فرمت داده ها و پرتكولهاي پيام رساني تبديل كرده. استفاده از xml شركتها را قادر مي سازد كه استانداردهاي رابط را براي سيستمها و سرويسها فراهم آورد و صاحبان صنايع را قادر به برقراي ارتباط با كمترين هزينه توسعه مي كند. در اين بخش يك شرح خلاصه از اين تكنولوژي ها براي شما بيان خواهيم كرد.

SOAP

بيشتر پيشرفتها در ارتباط به كمك xml، حوال اين تكنولوژي يعني Simple Object Access Protocol (SOAP) بوده. پرتكولSoap يك مشخصه پيام رساني است كه داده هاي رمزگذاري و قواعد بسته بندي را براي ارتباط بر پايه xml را شرح ميدهد. پرتكولsoap مشخص ميكند كه چگونه يك پيام بايد ايجاد، بسته بندي و به مقصد ارسال شود.
اين ارتباط شامل يك به هم بستن يا نگاشت به پرتكول HTTP مي باشد و به به اين معني مي باشد كه پيامهاي soap قابل انتقال با سيستم وب جهاني كنوني است. بيشتر پايه soap بر اساس XML-RPC مي باشد اين قابليت شرح براي خصوصيت فراخواني روال راه دور كه به كمك xml اجرا مي شود است. پرتكول soap هر دو مدل انتقال پيام بصورت همگام و ناهمگام بين سرويس دهنده وسرويس گيرنده را پشتيباني ميكند.
در پيام رساني به صورت همگام سرويس گيرنده يك درخواست را براي يك روال به همراه پارامتر ها در قالب xml به سرويس دهنده soap و به فرم پيامsoap request ارسال مي كند. سرويس دهنده نيز داده هاي درخواست سرويس گيرنده را به فرم

soap response message براي آن ارسال مي كند. اين اعمال در شكل زير نمايش داده شده.



پيام رساني نا همگام بطور كامل توسط soap پشتيباني شده.اين در وضعيتي مناسب مي باشد كه در آنجا قصد به روز رساني اطلاعات را با ارسال و دريافت آنها در زماني كه تغيير اتفاق مي افتد داريم. رخداد به روز رساني اغلب نياز به پاسخ سريع ندارد و يك پيام پاسخ به همان نقطه در آينده ارسال خواهد شد. اين پاسخ ممكن است تصديق براي دريافت پيام باشد و گزارشي از وضعيت پردازش در سمت گيرنده پيام باشد.
علاوه بر به روز رساني سروسي گيرنده مي تواند درخواست براي روال را داده و در اتصال بعدي كه زمان آن مشخص نيست پاسخ را دريافت كند. در شكل زير پيام رساني به صورت ناهمگام نشان داده شده.

سرويسهاي وب

رابطه نزديكي بين توسعه با soap و مفاهيم سرويس هاي وب وجود دارد.
سرويسهاي وب يك عبارت با كاربردهاي مختلف در استاندارد سازي براي معماري سرويس هاي تجاري توزيع در اينترنت است. سرويس هاي وب بر روي سرويس دهنده و گيرنده soap براي انتقال پيامهاي inter-enterprise تكيه كرده اند.
دو موضوع پيام رساني با xml و سرويس هاي وب در كنار هم در حد كافي پيچيده مي باشند و در اين بخش ما فقط پايه اي از سرويس هاي وب و تكنولوژيهاي وابسته را شرح خواهيم داد.
ادامه پيشرفت سرويس هاي وب تا به اين جا رسيده كه يك استاندارد براي ثبت(register) و پيدا كردن سرويس هاي وب با استفاده از يابنده سرويسهاي توزيع شده يا موتور جستجو تعريف شده. اين يابنده هاي سرويس، از xml براي شرح دادن سرويس هاي وب و شركتهاي كه آنها را ارائه مي دهند استفاده مي كنند.

اين سرويس هم اكنون با Universal Description, Discovery, and Integration (UDDI) نام گذاري شده و فروشنده گاني مثل ميكروسافت و IBM نمونه هاي از آن را ايجاد كرده اند.

UDDI

يك كنسرسيوم از شركتهاي بزرگ با هم براي ايجاد يك مجموعه استاندارد حول محور ثبت و فرآيند كشف سرويسهاي وب با هم همكاري كردند و نتيجه به آنجا رسيد كه UDDI به عنوان استاندارد پذيرفته شد.
هدف UDDI ايجاد قابليت ثبت online و جستجوي سرويس هاي وب به كمك يابنده هاي در دسترس است. همانند عمل Domain Name System (DNS) كه آدرس url يك سرويس وب را گرفته و آدرس IP كامپيوتري را كه اين سرويس روي آن مي باشد بر مي گرداند UDDI عمل مي كند. رجيستري سرويس(نام سرويس هاي كه در آن سرويس دهنده وجود دارد) به صفحات سبز اشاره دارند و در يك الگوي xml تعريف شده اند.
صفحات سبز اتحاديه هاي(syndicate) در سراسر سايتهاي عامل هستند. هر سايت عامل اطلاعات كلي درباره سرويس ها در سطوح مختلف مي دهند. اين اطلاعات مثل فرا داده (metadata) يا داده در مورد داده بوده و با نام tModel شنا ساي مي شوند.
يكي از چالشها در ثبت كردن سرويس وب تصميم گيري براي رده بندي آن مي باشد.يك ليست راكد الفبا پيدا كردن سرويسها با نوعهاي ويژه را ناممكن ميكند. بنابراين UDDI اجازه مي دهد كه رده بندي سرويسها بر اساس نواحي جغرافياي و كدهاي استاندارد صنعتي مثل NAICS, UN/SPC صورت بگيرد.براي اطلاعات بيشتر راجبع UDDI مي توانيد به آدرس http://www.uddi.org مراجعه كنيد.

WSDL

ايجاد كننده گان UDDI درك كردند كه نياز به يك استاندارد براي شرح دادن سرويسهاي وب در داخل رجيستري دارند و آنها Web Service Description Language (WSDL) را ايجاد كردند.
استاندارد WSDL از يك زبان xml كه براي شرح كلي سرويس وب مي باشد استفاده مي كند. اطلاعات داخل هر شرح دهنده شامل يك آدرس شبكه، پروتكل و مجموعه اعمال پشتيباني شده است.

تكنولوژي ذخيره و بازيابي داده ها

ذخيره و بازيابي داده ها در فرمت xml به يكي ازكاربردهاي xml تبديل شده. نياز ما به ذخيره و بازيابي داده ها در فرمت xml در نتيجه ايجاد تعداد بيشماري از ويژگي هاي آن كه نزديك به هم مي باشد. در اين بخش با خلاصه اي از اين ويژيگي ها آشنا مي شويد.

XPath

زبان XPath براي آدرس دهي ساختارهاي xml بوده كه در ساير استاندارد هاي گوناگون xml مثل xslt ، XPointer، XQuery استفاده مي شود. آن تعريف كننده نحو(syntax) براي ايجاد عبارات، آنهاي كه در يك سند xml ارزيابي شده اند استفاده مي شود.
براي مثال علامت (/) يك عبارت ساده XPath است و همان طور كه قبلآ گفتيم اين عبارت گره ريشه در يك سند xml مي باشد. عبارتهاي XPath مي توانند يك

مجموعه گره، مقدار منطقي، عدد يا رشته را نشان دهند. آنها از عنصر ريشه يا از محل خاص در آن سند شروع كرده و اكثر انواع عبارتهاي XPath عبارتهاي مسير يك محل در يك مجموعه گره مي باشند. براي مثال XPath product catalog زير تمام گره هاي محصول در كاتالوگ را نشان مي دهد.

/product

در داخل XPath يك مجموعه از توابع وجود داردكه شما را قادر مي سازد كه عبارات بسيار پيچيده را توسعه دهيد.

XPointer

XPointer يك سري خوصوصيات زبان خاص است كه بر روي XPath طراحي شده است. عبارات XPointer فقط براي يك مجموعه عبارات نبوده و براي مشخص كرده يك محدوده يا وضعيت در داخل يك مجموعه گره ها كه در يك شرط خاص صادق هستند. توابع XPointer داراي متدهاي قدرتمند براي جستجو در داخل ساختارهاي xml مي باشند.
براي مثال گره هاي زير را در نظر بگيريد.

```
<desc>This chapter provides an overview of the J2EE echnologies.</desc>
<desc>This chapter provides an overview of the XML landscape.</desc>
<desc>This chapter is an introduction to distributed systems.</desc>
```

يك عبارت ساده XPointer كه روي اين گره ها عمل مي كند به شكل زير ميباشد.

```
xpointer(string-range(//desc, 'overview'))
```

اين عبارت تمام گره ها با نام desc كه شامل رشته overview مي باشند را بر مي گرداند. عبارتهايXPointer در اشكال مختلف مي توانند شكل بگيرند و سريع پيچيده شوند.

XInclude

مكانيزم XInclude براي ضمينه كردن يك سند xml در داخل ساير سندهاي xml مي باشد. اين به ما اجازه مي دهد كه وابستگي هاي پيچيده را بين سند هاي مختلف xml ايجاد كنيم. براي ايجاد از تگ <include> استفاده كرده و مكان مشخص را براي سند مورد نظر تعين كنيد. ميتوانيد ذكر كنيد كه آيا آن قسمت مي تواند توسط پارسر، پارس شود يا خير.
تگ <include> در هر كجا از سند xml مي تواند استفاده شود و اشاره به يك سند كامل xml يا يك XPointer براي قسمتي از يك سند داشته باشد. استفاده از XInclude با XPointer امكان شامل كردن داده هاي خاص xml و گرفتن آنها از فايلهاي مختلف را فراهم مي كند.
اضافه كردن خط زير به يك سند xml يك مجموعه گره را از فايل خارجي afile.xml به سند مورد نظر در مكان نوشتن آن ضمينه خواهد كرد.

```
<xi:include href=”afile.xml#xpointer( XPath expression )” parse=”xml” />
```

XLink

تكنولوژي XLink برا آسان كردن ارتباط بين منابع مختلف در سندهاي xml مورد استفاده قرار مي گيرد.دليل ايجاد آن، ارتباط منابع xml همانند يك مكانيزم آنچه كه ابر لينكها(hyperlinks) در html ارائه مي كردند ضروي شمرده مي شد. تكنولوژي XLink در نوع پيمايش را در خود دارد ساده و توسعه يافته. XLink هاي ساده

قواعدي شبيه به ابر لينكها در html دارند و XLink هاي توسعه يافته قابليتهاي اضافي علاوه بر قابليتهاي XLink ساده دارند.
قابليت XLink اين امكان را به ما مي دهيد كه وابستگي هاي رابطهاي بين اشياء را به وجود آوريم.در مثال زير يك رابطه بين سفارش و مشتري ايجاد مي كنيم.
متن سند xml زير يك مشتري را نشان مي دهد:

```
<customer id=”0059”>
  <name>ABC Company</name>
  <employees>1000-1500</employees>
</customer>
```

سند زير يك لينك به سند مشتري در خود دارد:

```
<orders>
 <order xlink:type=”simple”
  href=”customers.xml#//customers/customer/@id[.='0059']”
    title=”Customer” show=”new”>
  <number>12345</number>
  <amount>$500</amount>
 </order>
</orders>
```

XBase

مكانيزم XBase يا XML Base براي مشخص كردن پايه براي شناساي كننده منابع يكسان(Uniform Resource Identifier URI) در سندهاي xml همانند زير دنباله از ارجاعات كه به چيزي اشاره مي كنند استفاده مي شود. استفاده از آن ساده بوده و به طور فراوان در دسترس است و به شما اجازه مي دهد Xlinkها را با طول قابل قبول استفاده كنيد.
در زير يك URI پايه با استفاده از XBase شرح داده شده.
هر تعداد ارجاع URI در داخل عنصر كاتالوگ مي تواند به راحتي با استفاده از http://www.manning.com/books به آن ارجاع كند و به عنوان پايه براي ارجاعات مي باشد.

```
<catalog xml:base=http://www.manning.com/books/>
        .....
        .....
</catalog>
```

زبانهاي پرس وجو

زماني كه داده هاي ذخيره شده در سند xml افزايش يافتند اين نكته اهميت خواهد داشت كه چندين زبان پرس وجوي مختلف براي سندهاي xml توسعه داده شده اند. يك از موثرترين اين زبانها XML Query Language (XQL) مي باشد. XQL يك زبان براي پرس وجو از ساختار داده اي xml بوده و اشتراك گسترده اي با XPath

دارد. بعد از استفاده از پرس وجوهاي XQL آن يك مجموعه از گره ها را از يك يا چند سند xml برمي گرداند.
ساير زبانهاي پرسوجو XML QL , Quilt مي باشند و W3C در حال كار بروي استاندارد كردن اين زبانها و تركيب ويژگي هاي هر كدام با هم است. نتيجه به دست آمده از اين استاندارد XQuery بود. زبان XQuery از احاظ نحوي شبيه به زبان SQL بوده و شامل كلمات كليدي RETURN, WHERE, LET, FOR مي باشد.
عبارت XQuery زير تمام گره هاي محصول را از afile.xml برميگرداند.

```
document( "afile.xml" )//product
```

عبارت كمي پيچيده زير گره warranty براي هر محصول را انتخاب مي كند.

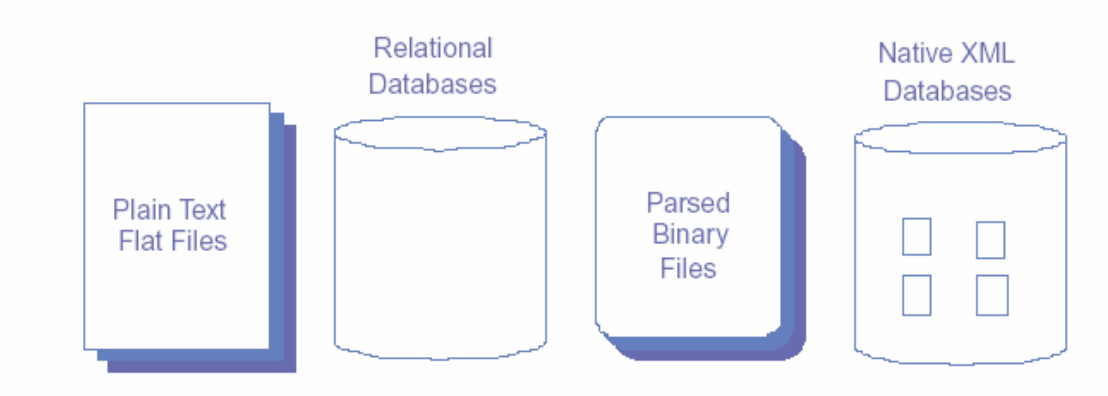
```
FOR $product in  //product
      RETURN $product/warranty
```

براي گرفتن اطلاعات بيشتر در مورد آنچه كه در اين بخش بحث شد مي توانيد از آدرس http://www.w3c.org/TR/ كمك بگيريد.

تكنولوژي ذخيره داده ها

ذخيره داده هاي xml در سيستم فايلها(filesystem) هم اكنون بسيار مورد استفاده قرار ميگيرد. اما ذخيره xml به صورت متني و با فرمت پارس نشده قابليت استفاده را پائين آورده و غير موثر مي باشد. مسندات ايستا در هر زمان كه آنها مورد نياز هستند بايد پارس شوند و يك مكانيزم مياني براي ذخيره فايلهاي متني Persistent Document Object Model(PDOM) مي باشد. در W3C روش PDOM مشخصات W3C DOM را كاملتر كرده و تفاوت در اين است كه سند پارس شده xml را در فرمت باينري در روي سيستم فايلها نگهداري مي كند و نياز نيست هر بار آن را پارس كنيم.
سندهاي PDOM ممكن است از DOM موجود يا از جريان ورودي xml ايجاد شوند ونياز به در حافظه بودن كامل در هر زمان ندارند. اين مزيت در جاهاي كه مستندات xml بزرگ مي باشند قابل استفاده است.
PDOM اعمال استاندارد مورد انتظار از يك كامپوننت ذخيره داده را در خود دارد مثل پرسوجوها با XQL ، حذف كردنها، فشرده سازي و كش كردن(caching).
روش ديگر ذخيره در سيستم فايلهاي ايستا استفاده از Native-XML Database است.نرم افزار هاي مثل Tamino بطور ويژه براي xml طراحي شده اند. اين نرم افزار مستندات xml را در فرمت مخصوص به خود(native) نگهداري مي كند، اما بانكهاي اطلاعاتي رابطه اي سلسه مراتب سندهاي xml را در داخل جداول ذخيره مي كنند.
اين كار كاراي Tamino را وقتي كه با مستندات xml كار ميكند را بالا مي برد. بنا براين ظهور فروشنده گان بانك هاي اطلاعاتي Native-XML، فروشنده گان بانكهاي اطلاعاتي تجاري مثل IBM, Oracle كه قصد دست كشيدن از تجارت ذخيره دادها را ندارند را به تكاپو وا داشت.
آنها با ايجاد قسمتهاي بروي محصولاتشان همكاري با داده اي xml را بالا برده و آنها را قادر به اجراي قابليت هاي پرسوجو كردند.
اين مزيت براي بسياري از شركتها كه محصولاتشان بروي بانكهاي اطلاعاتي رابطهاي بود بسيار مفيد ولقع گرديد.

در شكل زير خلاصه اي از چهار گزينه براي ذخيره داده هاي xml نشان مي دهد.



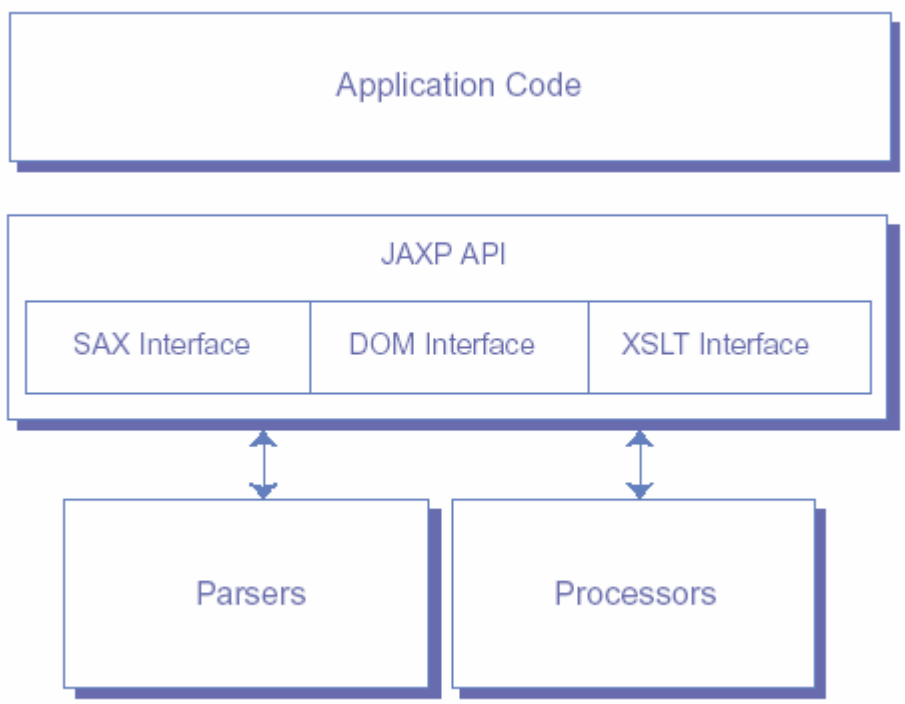
APIهاي جاوا براي XML

كميته توسعه جاوا در زمينه تكنولوژيهاي xml كه در قسمت قبل گفته شد فعاليت زيادي انجام داده و اغلب جاوا اولين زباني است كه تكنولوژي هاي جديد xml را توسعه مي دهد. هر چند كه گروه هاي زيادي در زمانهاي مختلف توابع مختلفي براي جاوا طراحي كرده اند و در برخي مواقع با هم ناسازگار مي باشند. شركت سان با كميته فرآيندهاي جاوا java community process (JCP) استاندارد Java API for XML (JAX) را براي توسعه برنامه هاي كاربردي ايجاد كرد. در جدول زير قسمتهاي مختلف اين مجموعه API را ملاحظه مي كنيد.

Java API for XML	JAX acronym	Functional description
Java API for XML parsing	JAXP	Provides implementation-neutral access to XML parsers and XSLT processors.
Java Document Object Model	JDOM	Provides a Java-centric, object-oriented implementation of the DOM framework.
Java API for XML binding	JAXB	Provides a persistent XML mapping for Java object storage as XML.
Long Term Java-Beans Persistence		Similar to JAXB, provides XML serialization for JavaBean components.
Java API for XML messaging	JAXM	Enables the use of SOAP messaging in Java applications, using resource factories in a manner similar to the Java Messaging Service (JMS).
JAX-RPC	JAX-RPC	An XML-RPC implementation API for Java. Similar to JAXM.
Java API for XML repositories	JAXR	Provides implementation-neutral access to XML repositories like ebXML and UDDI.

JAXP

Jaxp يك سري APIعمومي كامل براي ايجاد و استفاده از DOM, SAX و XSLT API در جاوا است.برنامه شما مي بايد از jaxp استفاده كنيد و استفاده از API هاي كه توسط فروشنده گان خاص كه با سليقه خود ايجاد و پشتيباني شده اند خودداري كنيد. اين API ها روزبه روز سزيعتر وبهتر مي شوند. به منظور ارتقا با تعويض فايلهاي JAR(Java Archive) مي توتنيد APIهاي جديد را استفاده كنيد. با اين قابليت به هدف اصلي توسعه برنامه هاي توزيع شده يعني تطابق دست خواهيد يافت.
در شكل زير معماري jaxp را مشاهده مي كنيد.



همان طور كه در معماري مشاهده مي كنيد قابليت انعطاف با جدا كردن كدهاي برنامه كاربري از زير لايه XML APIاست.شما مي توانيد با زير لايه SAX يا DOM استراتژي نحوه پارس كردن سند xml خود را مشخص كنيد و همچنين براي تبديل xml از XSLT محصول فروشنده گان مختلف استفاده كنيد.
Jaxp شامل چهار بسته مي باشد كه در جدول زير آنها مشخص شده.

Package	Description
javax.xml.parsers	Provides a common interface to DOM and SAX parsers.
javax.xml.transform	Provides a common interface to XSLT processors.
org.xml.sax	The generic SAX API for Java
org.w3c.dom	The generic DOM API for Java

دو بسته از اين API ها در javax.xml قرار دارد.
بسته javax.xml.paraers شامل كلاسها و رابطهاي مورد نياز براي پارس كردن سند xml است. بسته javax.xml.transform تعريف كننده رابط به پردازنده XSLT است.

پيكره بندي JAXP

براي استفاده از jaxp براي پارس كردن شما به JAXP XML Parser نياز داريد.
همچنين براي پردازش xslt به XSLT Engine نياز داريد.
در اولين دستيابي به كلاس پارسر jxap در كدتان گامهاي زير را framework جاوا براي خود مقدار دهي خواهد كرد.

- آن ابتدا چك مي كند كه آيا خصيصه(property) سيستم يعني javax.xml.parsers.DocumentBuildeFactory يا javax.xml.parsers.SAXparserFactory تنضيم شده مي باشند يا نه. اگر نياز به تبديل xslt داريد خصيصه سيستمي javax.xml.transform.TranformerFactory را نيز چك مي كند.
- اگر اين خصيصه هاي به طور صريح تنظيم نشده باشند framework به دنبال فايل jaxp.properties در دايركتوري كتابخانه هاي JRE خواهد گشت. در زير يك مثال از محتواي اين فايل نشان داده شده.

```
javax.xml.parsers.DocumentBuilderFactory=
  org.apache.crimson.jaxp.DocumentBuilderFactoryImpl
javax.xml.parsers.SAXParserFactory=
  org.apache.crimson.jaxp.SAXParserFactoryImpl
javax.xml..transform.TransformerFactory=
  org.apache.xalan.processor.TransformerFactoryImpl
```

Sets DOM builder, SAX parser, and XSLT processor implementation classes

توجه داشته باشيد كه مثال بالا در چند خط نمايش داده شده و در فايلهاي jaxp.properties هر دستور دريك خظ بوده وبين علامت (=) نمي تواند فاصله وجود داشته باشد.

- اگر فايل jaxp.properties كه نام كلاس دلخواه براي پارسر است پيدا نشود framework به صورت پيش فرز پارسر Crimson را فراخواني خواهد كرد و از Xalan براي xslt استفاده خواهد كرد.

براي اينكه خصيصه هاي مربوط به پارسر و پردازنده را براي نگاشت به كلاس آنها تنظيم كنيد يك راه ديگر ساده تر براي پيكره بندي گذاشتن فايل JAR پارسر و پردازنده دلخواه در مسير كلاسهاي برنامه است. اگر در مسير كلاسهاي برنامه دو نوع JAXP API وجود داشته باشد شما بايد يكي از آنها را در فايل jaxp.properties گفته شده در بالا مشخص كنيد.
هم اكنون jaxp به قسمتي از مشخصه هاي J2EE تبديل گشته و به اين معني مي باشد كه شركتهاي پشتيباني كنندهJ2EE بايد از آن پشتيباني كنند. اين استفاده از jaxp را ساده تر كرده و استفاده مستقيم از XSLT, SAX, DOM فراهم مي كند.

استفاده از JAXP با SAX

كلاسهاي كليدي jaxp براي استفاده با SAX در جدول زير به نمايش در آمده. پيش از شرح استفاده از SAX ما مي بايست از توضيحات جزئي در باره پارس كردن به فرم SAX براي لحظاتي گذر كنيم. براي استفاده از SAX بدون jaxp شما مي بايد هميشه يك يا چند دستگير رخداد براي پارسر تعريف كنيد.

JAXP class or interface	Description
javax.xml.parsers.SAXParserFactory	Locates a SAXParserFactory implementation class and instantiates it. The implementation class in turn provides SAXParser implementations for use by your application code.
javax.xml.parsers.SAXParser	Interface to the underlying SAX parser.
javax.xml.parsers.SAXReader	A class wrapped by the SAXParser that interacts with your SAX event handler(s). It can be obtained from the SAXParser and configured before parsing when necessary.
org.xml.sax.helpers.DefaultHander	A utility class that implements all the SAX event handler interfaces. You can subclass this class to get easy access to all possible SAX events and then override the specific methods in which you have interest.

توجه كنيد كه يك دستگير رخداد SAX يك كامپوننت مي باشد كه خود را براي فراخواني هاي پارسر زماني كه يك رخداد اتفاق مي افتد رجيستر مي كند.
SAX API چهار دستگير رخداد اصلي را تعريف مي كند كه در رابط هاي org.xml.sax ازبسته EntityResolver ,DTDHandler ,ContentHandler ,ErrorHandler كپسوله سازي شده اند. رابط ContentHandler رابط اصلي است كه اغلب برنامه ها بايد آنر كامل كنند. اين رابط شامل متدهاي رخداد startDocument ,startElement endElemet ,endDocument, مي باشد.برنامه شما بايد اين متدهاي رخداد رابط ضروري را كه لازم دارد كامل كند .
ساير دستگيرهاي موجود در SAX براي كار هاي جانبي در پارس كردن سند xml استفاده مي شوند. رابط EntityResolver ما قادر مي سازد كه نگاشت ارجاعات به منابع خارجي مثل بانكهاي اطلاعاتي يا URL ها داشته باشيم. رابط ErrorHandler براي دستگير كردن خطاهاي ويژه در پردازش از SAXExceptions استفاده مي كند.رابط DTDHandler براي گرفتن اطلاعات درباره اعتبار سنجي سند از سندهاي DTDاستفاده مي شود.
همچنين SAX يك كلاس براي راحتي توسعه دهنده گان به نام org.xml.helpers.DefaultHandler ارائه مي دهد كه به كمك آن برنامه شما به تمامي رخدادهاي كامل شده پيش فرز SAX دسترس پيدا ميكند.
حال كه ما اطلاعاتي در باره نحوه كار SAX به دست آورديم نوبت گذاشتن jaxpبراي كار با آن مي باشد. براي مثال اجازه دهيد كه سند xml گذشته يعني product catalog را به كمك رخدادهاي SAX و jaxp بخوانيم.
براي كوتاه و مناسب كردن اين مثال ما فقط كلاس دستگير رخداد endElement را تعريف كرده ايم. هر زمان كه يك عنصر محصول به طور كامل از سند xml با پارسر SAX خوانده شد ما يك پيغام كه به آن اشاره دارد چاپ خواهيم كرد.
كد اين مثال در زير نوشته شده.

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class ProductEventHandler
  extends DefaultHandler {
//Extends this class to only handle the endElement event
```

```
  // other event handlers could go here
public void endElement( String namespaceURI,
                        String localName,
                        String qName,
                        Attributes atts )
    throws SAXException {
      // make sure it was a product node
      if (localName.equals("product"))
       System.out.println( A product was read from the catalog.);
 }
}
```

حال كه ما يك دستگر رخداد تعريف كرديم ، ما مي توانيم يك پارسر كامل SAX را به كمك jaxp در كد برنامه شما اجرا كنيم و دستگيرها را براي اعمال كامل كنيم. متد دستگير endElement زماني كه سند مثال ما حتي يك گره محصول در آن باشد فراخواني خواهد شد.
در زير مثالي كه از JAXP SAXاستفاده مي كند را مشاهد مي كنيد

```
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import java.io.File;
public class JAXPandSAX  {
  public static void main(String[] args) {
//Instantiates our event handler
    ProductEventHandler handler
      = new ProductEventHandler();
    try {
      SAXParserFactory factory
        = SAXParserFactory.newInstance();
//Obtains a SAXParser via JAXP
SAXParser parser
        = factory.newSAXParser();
      File ourExample
        = new File("product-catalog.xml");
      parser.parse( ourExample, handler);
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

زماني كه كد مثال بر روي سند product catalog اجرا مي شود شما خروجي زير را خواهيد ديد.

Product read from the catalog.

زماني كه ما يك محصول را در داخل سند مشخص كرده باشيم اين دستور فقط يك بار چاپ خواهد شد. اگر چند محصول در داخل سند باشد ايندستور به تعداد آن محصولات چاپ خواهد شد.

استفاده از JAXPبا DOM
استفاده از jaxp با DOM با استفاده آن در SAX كمي كمتر پيچيده مي باشد. اين به اين دليل مي باشد كه ما نياز به به توسعه دستگير رخدادها و پاس كردن آنها به پارسر نداريم.در استفاده از DOM آن به طور كامل سند xml را به داخل حافظه به صورت درخت واره خواهد خواند. اين به شما اين اجازه را مي دهد كه در هر زمان سند را بطور كامل دستكاري و تغيير دهيد. همچنين همانند SAX نياز به برنامه نويسي ماشين حالتها در كد خود نداريد(در SAX چون حجم سند ميتوانست بزرگ باشد وبطور كامل در حافظه قرار نداشت حالتهاي مختلفي براي برنامه اتفاق مي افتاد).
البته اين ساده گي و راحتي منابع و هزينه بيشتري براي سيستمي كه برنامه روي آن در حال اجرا مي باشد مصرف خواهد كرد و سرعت كمي كاهش پيدا مي كند. كلاسهاي اصلي JAXP DOMدر جدول زير بطور خلاصه نشان داده شده.

JAXP class or interface	Description
javax.xml.parsers.DocumentBuilderFactory	Locates a DocumentBuilderFactory implementation class and instantiates it. The implementation class in turn provides DocumentBuilder implementations.
javax.xml.parsers.DocumentBuilder	Interface to the underlying DOM builder.

زماني كه سند product catalog طول كمي دارد مشكلي با DOM براي خواندن از آن نداريم. كد خواندن از اين سند براي خواندن به كمك DOM در زيزر نوشته شده و مي توانيد ببينيد كه گامهاي كلي از گرفتن يك پارسر از JAXP و فراخواني آنها همانند روش در SAX مي باشد. فرق اساسي در دستگيرهاي SAX ميباشد.
توجه كنيد كه پارسر يك اشاره گر به DOM بعد از پارس كردن بر مي گرداند. براي استفاده از DOM API از بسته org.w3c.dom استفاده مي كنيم.شما مي توانيد سند به فرم DOM را با كمك كدها در حافظه پيمايش كنيد و محصولات در كاتالوگ را مشاهده كنيد.

```
import javax.xml.parsers.DocumentBuilderFactory;
//Imports the JAXP DOM classes
```

```
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import java.io.File;
public class JAXPandDOM  {
  public static void main(String[] args) {
    try {
   DocumentBuilderFactory factory
        = DocumentBuilderFactory.newInstance();
//Obtains a DOMBuilder via JAXP
      DocumentBuilder builder
        = factory.newDocumentBuilder();
      File ourExample
        = new File("product-catalog.xml");
//Parses the XML and builds a DOM tree
      Document document
        = builder.parse( ourExample );
  } catch (Exception e) {
   System.out.println(e.getMessage());
  }
 }
}
```

استفاده از JAXP با XSLT

JAXP از XSLT بطور مستقل از پياده سازي(از سازنده گان مختلف) به همان روشي كه با پارسرها برخورد مي كرد پشتيباني مي كند. رابط JAXP به XSLT در بسته javax.xml.transform موجود است.كلاسها و رابطهاي اصلي در جدول زير مشخص شده. به علاوه اين رابطهاي سطح بالا JAXP شامل سه زيربسته براي پشتيباني از استفاده I/O Stream, DOM, SAX به همراه XSLT است. اين بسته ها در جدول بعدي به نمايش درآمده اند.

JAXP class or interface	Description
javax.xml.transform.TransformerFactory	Locates a TransformerFactory implementation class and instantiates it.
javax.xml.transform.Transformer	Interface to the underlying XSLT processor.
javax.xml.transform.Source	An interface representing an XML data source to be transformed by the Transformer.
javax.xml.transform.Result	An interface to the output of the Transformer after XSLT processing.

در قسمتهاي قبلي ما روي فرآيند تبديل صحبت كرديم و گفتيم كه چگونه مي توان product catalog را به يك سند html با كمك XSLT ترجمه كرد. حال به شرح چگونه گي فراخواني فرآيند XSLT با كدهاي جاوا به كمك JAXP خواهيم پرداخت. به منظور ساده گي و وضوح از كلاس كمكي I/O Stream در داخل بسته

javax.xml.transform.stream براي شي هاي Source و Result استفاده خواهيم كرد.

Package name	Description
javax.xml.transform.dom	Contains classes and interfaces for using XSLT with DOM input sources and results.
javax.xml.transform.sax	Contains classes and interfaces for using XSLT with SAX input sources and results.
javax.xml.transform.stream	Contains classes and interfaces for using XSLT with I/O input and output stream sources and results.

كد جاوا كه كار تبديل يك سند xml بهhtml را دارد در زير نوشته شده. براي كامپايل كردن آن شما بايد فايل JAXP jar را در مسير كلاس هايتان داشته باشيد. همچنين فايل xml و xsl گفته شده در قسمتهاي قبل به نامهاي product-catalog.xml و product-catalog-to-html.xsl كه stylesheet مربوط به ساختار تبديل است بايد را داشته باشيد.
شما اين فايلها را با با ويراستار دلخواه مي توانيد تايپ كرده يا از سايت http://www.manning.com/gabrick دانلود كنيد. قبل از تست كردن اين مثال نياز به جاي دادن يك JAXP-compliant XSLT engine همانند Xalan در مسير كلاس هايتان داريد.

```
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
//Imports the JAXP XSLT API
import java.io.File;
public class JAXPandXSLT {
  public static void main(String[] args) {
    File sourceFile
      = new File("product-catalog.xml");
   // Loads the XML and XSL files
File xsltFile
      = new File("product-catalog-to-html.xsl");
    Source xmlSource = new StreamSource(sourceFile);
//Creates I/O Stream sources and results
    Source xsltSource = new StreamSource(xsltFile);
    Result result = new StreamResult(System.out);
//Returns an instance of TransformerFactory
    TransformerFactory factory
      = TransformerFactory.newInstance();
try {
//BFactory returns new Transformer
    Transformer transformer
```

```
      = factory.newTransformer(xsltSource);
//C Performs transformation
    transformer.transform(xmlSource, result);
    } catch (TransformerConfigurationException tce) {
      System.out.println("No JAXP-compliant XSLT processor
found.");
    } catch (TransformerException te) {
      System.out.println("Error while transforming document:");
      te.printStackTrace();
    }
  }
}
```

شي TransformerFactory مشخصات ويژه تبديل را كامل مي كند. دقت داشته باشيد كه قواعد تبديل در فايل XSLT stylesheet به كارخانه(مبدلي كه كار واقعي تبديل را انجام ميدهد) براي ايجاد شي تبديل كننده پاس شده اند.
متد Transform كار تبديل XSLT را انجام مي دهد و نتايج به به شي Result جريان داده مي شوند. در اين مثال جرياني كه نتايج خروجي را دريافت مي كند خروجي استاندارد يعني System.out است.
در يك نگاه استفاده JAXP از XSLT براي تبديل زياد پيچيده نيست. اين درست است كه اين تبديل ساده مي باشد، اما بسياري از ويژگي هاي فرآيند XSLT با رابطهاي Transformer و TransformerFactory وجود دارند كه قابل پيكره بندي مي باشند.همچنين شما مي توانيد دستگير هاي خطا براي خطاهاي كه ممكن است در حين تبديل اتفاق بافتند ايجاد و ثبت كنيد.براي اطلاعات بيشتر ليست مستندات كامل JAXP را نگاه كنيد.

كمي احتياط

استفاده از XSLT با JAXP بدون چالش نخواهد بود. يك مانع بزرگ براي استفاده گسترده از XSLT كاراي در زمان اجرا است. اجراي يك تبديل XSLT روي يك سند زمان بر بوده و منابع بسياري را مي گيرد. برخي پردازنده هاي XSLT همانند Xalan به شما اجازه مي دهند كه قواعد تبديل داخل stylesheet را براي كاراي بيشترپيش كامپايل كرده.

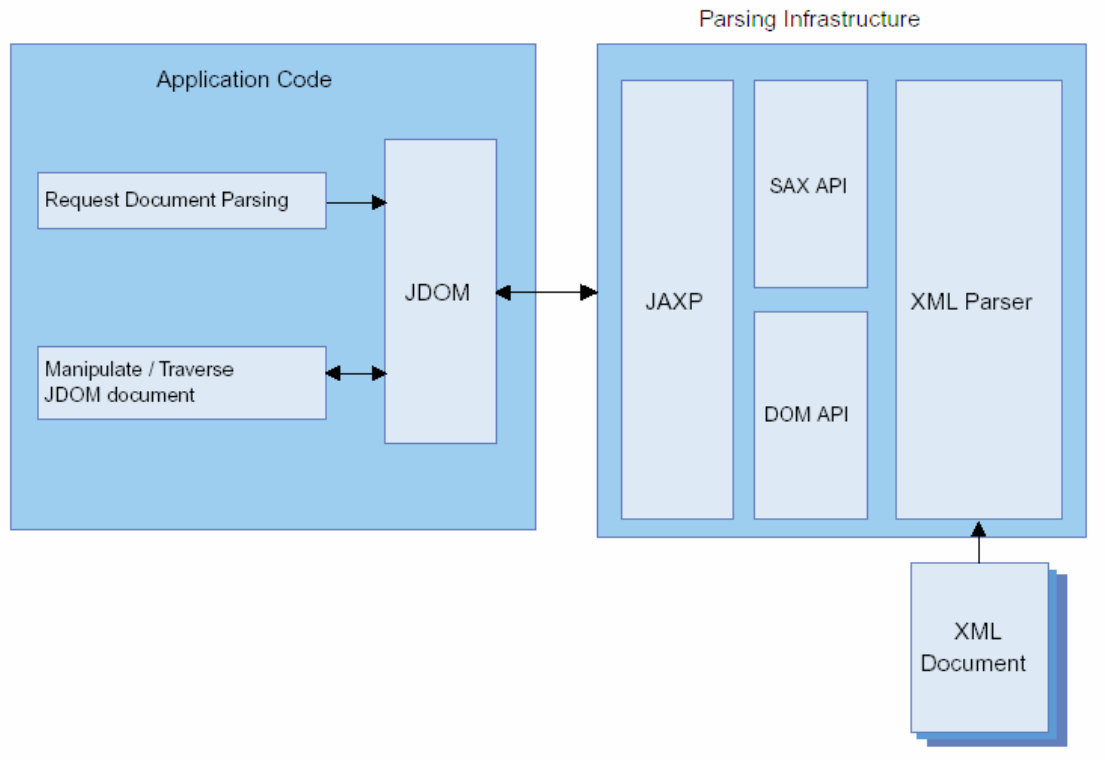
JDOM

اولين چيز كه درباره اعضاء خانواده Java API for XML(JAX) فهميده مي شودحروف مخفف براي بسته هاي آن است. شما هم اكنون مي توانيد كد براي برنامه هاي كاربردي بنويسيد كه مستقل از پارسر مي باشند. در اينجا ديگر API ها وجود دارند كه اين كار ها را بيشتر ساده مي كنند. آنها را با Java Document Object Model (JDOM) نام گذاري كرده اند و هم اكنون توسط كميته فني جاوا به صورت رسمي پذيرفته شده.

اين سري از APIها براي كار با ساختار هاي xml به صورت جاوا محور ايجاد شدند. بطور ويژه براي جاوا طراحي شده اند يك مدل شي قابل استفاده آسان مي باشد. براي مثال JDOMاز كلاسهاي Java Collection مثل java.util.list براي مجموعه گره ها در كار با سندهاي xml استفاده مي كند.علاوه براين كلاسهاي JDOM ارتباط قوي با هم دارند و كامل مي باشند در حالي كه كلاسهاي DOM به صورت انتزاعي مي باشند. اين قابليت استفاده از آن را ساده كرده و وابستگي ها و تنظيمات مخصوص فروشنده گان را همانند JAXP را حذف مي كند.

استفاده از اكثر ورژنهاي JDOM همانند استفاده از JAXP API مي باشند. اين به اين معني است كه JDOM تغيير زيادي در معماري JAXP انجام نداده و فقط روي آن ايجاد شده و زماني كه يك شي xml ايجاد مي كنند از API هاي قابل دسترس JAXP استفاده مي كنند.

در شكل زير معماري JDOM نشان داده شده.



در جدول زير كلاسهاي مركزي JDOM به نمايش درآمده. همان طور كه شما مي بينيد آنها كاملآ با معنا نام گذاري شده ند. يك سند JDOM مي توانند در حافظه ايجاد شود يا از يك جريان و URL ايجاد شود.

Class name	Description
org.jdom.Document	The primary interface to a JDOM document.
org.jdom.Element	An object representation of an XML node.
org.jdom.Attribute	An object representation of an XML node's attribute.
org.jdom.ProcessingInstruction	JDOM contains objects to represent special XML content, including application-specific processing instructions.
org.jdom.input.SAXBuilder	A JDOM builder that uses SAX.
org.jdom.input.DOMBuilder	A JDOM builder that uses DOM.
org.jdom.transform.Source	A JAXP XSLT Source for JDOM Documents. The JDOM is passed to the Transformer as a JAXP SAXSource.
org.jdom.transform.Result	A JAXP XSLT Result for JDOM Documents. Builds a JDOM from a JAXP SAXResult.

براي شرح سريع نحوه كار JDOM اجازه دهيد كه سند product catalog از حافظه خود پاك كرده و سپس ما آترا از نوع بسازيم و به داخل يك فايل بنويسيم.براي اين كار ابتدا ما به ساده گي درخت JDOM Element و يك JDOM Document از آن ايجاد مي كنيم. كد براي اين كار در قسمت زير مشخص شده و زماني كه شما اين كد را كامپايل و اجرا مي كند بعد از آن مي بايست يك ورژن خوش شكل(well-formatted) از سند xml كه بعد از در فهرست جاري شما ايجاد شده و موجود باشد.

```
import org.jdom.*;
import org.jdom.output.XMLOutputter;
import java.io.FileOutputStream;
public class JDOMCatalogBuilder {
  public static void main(String[] args) {
    // construct the JDOM elements
    Element rootElement = new Element("product-catalog");
    Element productElement = new Element("product");
//Creates element attributes
    productElement.addAttribute("sku", "123456");
    productElement.addAttribute("name", "The Product");
    Element en_US_descr = new Element("description");
    en_US_descr.addAttribute("locale", "en_US");
// Adds text to the element
   en_US_descr.addContent("An excellent product.");
    Element es_MX_descr = new Element("description");
    es_MX_descr.addAttribute("locale", "es_MX");
    es_MX_descr.addContent("Un producto excellente.");
```

```
    Element en_US_price = new Element("price");
    en_US_price.addAttribute("locale", "en_US");
    en_US_price.addAttribute("unit", "USD");
    en_US_price.addContent("99.95");
    Element es_MX_price = new Element("price");
    es_MX_price.addAttribute("locale", "es_MX");
    es_MX_price.addAttribute("unit", "MXP");
    es_MX_price.addContent("9999.95");
    // arrange elements into a DOM tree
  //Builds the DOM by adding one element as content to another
    productElement.addContent(en_US_descr);
    productElement.addContent(es_MX_descr);
    productElement.addContent(en_US_price);
    productElement.addContent(es_MX_price);
    rootElement.addContent(productElement);
    //Wraps root element and processing instructions
    Document document = new Document(rootElement);
    // output the DOM to "product-catalog.xml" file
//Indents element two spaces and uses newlines

   XMLOutputter out = new XMLOutputter("  ", true);
    try {
      FileOutputStream fos = new FileOutputStream("product-
catalog.xml");
//Writes the JDOM representation to a file
      out.output(document, fos);
    } catch (Exception e) {
      System.out.println("Exception while outputting JDOM:");
      e.printStackTrace();
    }
  }
}
```

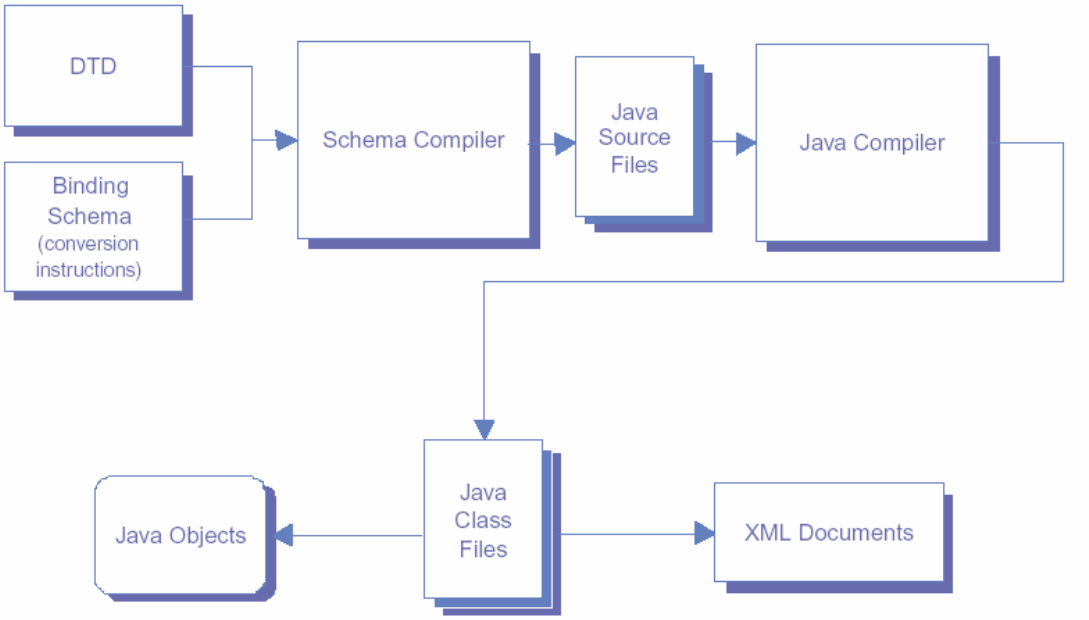
در قسمتهاي بعدي به طور گسترده از JDOM استفاده خواهيم كرد. JDOM با رابطهاي JAXP كار كرده و آن را پشتيباني مي كند. براي اطلاعات بيشتر درباره JDOM مي توانيد به آدرس http://www.jdom.org مراجعه كنيد.

JAXB

به منظور نگاشت دو طرفه بين اشياء داده جاوا و ساختارهاي xml از JAXB كه مخفف كلمه هاي Java API for XML Binding مي توانيم در برنامه هاي خود استفاده كنيم. هدف از اين كار ايجاد ايجاد ماندگاري و سازگاري اشياء جاوا با xml براي راحت شدن

كار توسعه است.
بدون JAXB فرآيند ذخيره و بازيابي(serializing و deserializing به ترتيب) اشياء جاوا با xml نياز به ايجاد و نگهداري كد هاي سنگين و زيادي براي خواندن،پارس كردن و خروجي گرفتن از آنها داريم.
مي توان فرآيند سريال سازي(serialization) نوشتن نتيجه خروجي يك شي نرم افزاري در حال اجرا به يك جريان خروجي تعريف كرد. اين جريانهاي خروجي مي توانند يك فايل يا سوكتهاي داده TCP باشد.
با JAXB شما قادر خواهيد بود با مستندات xml ،كه اگر آنها اشياء جاوا باشند كار كنيد.
فرآيند توسعه در JAXP نياز مند ايجاد يك DTD و binding schema در يك سند xml براي تعين نگاشت بين شي جاوا و شماي xml آن ميباشد.
شما بايد يك DTD , binding schema را به يك كامپايلر schema براي ايجاد كد هاي منبع جاوا ارسال كنيد. كلاسهاي نتيجه به وجود آمده از زمان كامپايل جزئيات فرآيند تبديل xml به java و برعكس را دستگير(اجرا) خواهند كرد.
اين به اين معني است شما به طور مستقيم نياز به اجراي پارسرهاي SAX يا DOM در كد برنامه خود نداشته و شكل زير فرآيند JAXB را روشنتر نشان مي دهد.

در زماني كه براي اولين بار JAXB ايجاد شد آن نشان داد كه مي توان كارائي بر روي پارسرهاي SAX و DOM را بهبود داد. اين به دليل مي باشد كه كلاسهاي آن پيش كامپايل شده و سبكتر مي باشند. اين از نشانه هاي آينده مثبت JAXB است و معلوم ميكند ارتباط عمومي با كارائي وقتي كه از xml استفاده مي شود دارد.



در يك ارزيابي در استفاده از JAXB اين است در اين جا انعطاف پذيري در سيستم كمتر مي شود و اين زماني كه هر تغيير در xml و ساختار اشياء نيازمند به دوباره كامپايل كردن JAXB كلاسها دارد. اين عيب براي ايجاد سيستمهاي كه از JAXB در برنامه هاي خود استفاده مي كند اشكال ايجاد خواهد كرد.
هر تغيير در ساختار JAXB نيازمند ايجاد دوباره JAXB binding و تست دوباره آنها را به همراه دارد. اعلاميه هاي در مورد JAXB اعلام شده در باره اينكه در ورژن جديد اين API پيش از استفاده آنها در برنامه هايتان شما مي بايد آنرا كاوش كنيد. به طور مثال

فرآيند مشخص مي كند كه كدام ساختار داده xml ايجاد شده اند و از جدولهاي رابطه كه ساده هستند و منابع زيادي دارند استفاده مي كنند. نتايج مورد انتظار و به دست امده از اين ارتقاع در اينجا امكان گفتن ندارند و درمورد آنها در يك زمان ديگر بايد بحث كرد. براي اطلاعات بيشتر مي توانيد به آدرس http://java.sun.com/xm/jaxb/ مراجعه كنيد.

مقيد(binding) كردن اشياء جاوا به XML

براي ديدن JAXB در عمل ما به مثال product catalog برمي گرديم. در گذشته ما يك DTD براي آن توسعه داده ايم و آن را ملاحظه كرده ايد.ايجاد الگوي مقيد كردن(binding schema) كمي پيچيده تر از ايجاد DTD است و ما كار را با ايجاد فايل جديد binding schema به نام product-catalog.xjs شروع مي كنيم. الگو هاي مقيد كردن در ورژنهاي اوليه JAXB هميشه عناصر ريشه زير را دارند.

```
<xml-java-binding-schema version="1.0-ea">
```

اين عنصر تعين مي كند كه اين سند يك الگوي مقيد كردن ميباشد.حال ما عناصر پايه و دروني در سند product catalog را تعريف مي كنيم.

```
<element name="description" type="class">
  <attribute name="locale"/>
  <content property="description"/>
</element>
```

همچنين

```
<element name="price" type="class">
   <attribute name="locale"/>
   <attribute name="unit"/>
   <content property="price"/>
 </element>
```

صفت type از عنصر گره به اين اشاره دارد كه عناصر از نوع description و price در سند product catalog بايد همانند اشياء جدا در جاوا آنها برخورد شود. اين كار براي اين ضروري است كه هر دو description و price صفات خاص خود را همانند آنچه كه در سند داردن داشته باشند. عنصر content در هر دو تعريف بالا به كامپايلر JAXB مي فهماند كه صفات براي اين كلاسها را با نام ويژه شان ايجاد كند.
محتواي ايجاد شده كلاس Description با متد هاي getDescription() و setDescription() مي توانند مورد دسترسي قرار بگيرند. همچنين براي كلاس Price متدهاي getPrice() و setPrice() ايجاد مي شوند.براي اين كه يك شرح خلاصه از اين عناصر داشته باشيم ما مي توانيم به آنها در تعريفات عنصر product مراجعه كنيم.

```
<element name="product" type="class">
  <content>
    <element-ref name="description"/>
```

```
    <element-ref name="price"/>
  </content>
</element>
```

عنصر product به کلاس جاوا با نام Product نگاشت خواهد شد و شامل دو متغير نمونه که يکي متغير نمونه Description و ديگري Price خواهد بود. توجه کنيد که ميتوانيد از کلمه element-ref به جاي element در در تعريف گره هاي description, price استفاده کنيد. استفاده از اين ساختار مي تواند براي ساخت اشياء با ساختار پيچيده و از به دور انداختن اطلاعات در سند binding جلوگيري کند. عنصر آخري که به عنصر ريشه مقيد مي شود در تعريفات زير نشان داده شده.

```
<element name="product-catalog" type="class" root="true">
  <content>
    <element-ref name="product"/>
  </content>
</element>
```

توجه کنيد که صفت root=true در اين تعريف مقيد سازي، product-catalog را به عنوان ريشه عنصر xml تعين ميکند. از اين تعريف کامپايلر JAXB براي ايجاد کلاس به نام ProductCatalog شامل ليس نمونه محصولات استفاده ميکند. الگوي کامل مقيد سازي براي مثال ما در زير نشان داده شده.

```
<xml-java-binding-schema version="1.0-ea">
  <element name="description" type="class">
    <attribute name="locale"/>
    <content property="description"/>
  </element>
  <element name="price" type="class">
    <attribute name="locale"/>
    <attribute name="unit"/>
    <content property="price"/>
  </element>
  <element name="product" type="class">
    <content>
      <element-ref name="description"/>
      <element-ref name="price"/>
    </content>
  </element>
  <element name="product-catalog" type="class" root="true">
    <content>
      <element-ref name="product"/>
    </content>
  </element>
</xml-java-binding-schema>
```

حال ما يك DTD و يك الگوي مقيد سازي داريم و آمهده براي ايجادكد هاي منبع JAXB هستيم. از اين اطمينان پيدا كنيد كه در مسير كلاس هايتان(فايلهاي برنامه) فايل JAXB jar موجود بوده و دستور زير را اجرا كنيد.

java com.sun.tools.xjc.Main product-catalog.dtd product-catalog.xjs

اگر همه چيز درست باشد، فايلهاي زير را در فهرست جاري شما ايجاد خواهد شد.

Description.java
Price.java
Product.java
ProductCatalog.java

حل شما مي توانيد اين كلاسها را كامپايل كرده و در برنامه هايتان استفاده كنيد.

استفاده از اشياء JAXB

استفاده از كلاسهاس كامپايل شده JAXB در برنامه هاي كاربردي شما آسان مي باشد. براي خواندن اشياء از داخل فايلهاي xml شما به ساده گي با اشياء JAXB آنها را از سند آن مي خوانيد. اگر شما با استفاده از java.io.ObjectInputStream آشنا باشيد، اين كار بسيار ساده خواهد بود. در زير كد نمونه اي براي خواندن سند product catalog با JAXBنشان داده شده.

```
ProductCatalog catalog = null;
File productCatalogFile = new File("product-catalog.xml");
try {
  FileInputStream fis
    = new FileInputStream(productCatalogFile);
  catalog = ProductCatalog.unmarshal(fis);
} catch (Exception e) {
  // handle
} finally {
 fis.close();
}
```

براي فرآيند عكس آن يعني ذخيره كردن نمونه شي ProductCatalog به شكل سند xml بايد كد هاي زير را اجرا كنيد.

```
try {
  FileOutputStream fos
    = new FileOutputStream(productCatalogFile);
  catalog.marshal(fos);
} catch (Exception e2) {
  // handle
} finally {
```

```
  fos.close();
}
```

متد هاي unmarshal از شي ProductCatalog وظيفه تبديل عناصر داخل سند xml به كلاسهاي از نوع خود ProductCatalog را انجام مي دهد وهمچنين متد marshal از كلاس تعريف شده از نوع ProductCatalog قادر به تبديل كلاس كه از روي آن فراخواي شده به عناصر xml مي باشد. دقت داشته باشيد كه ورودي هر دو متد از نوع جريان ورودي و خروجي فايل مي باشد.
در طي پردازش برنامه شما مي توانيد از اشياء JAXB استفاده كنيد، همان طور كه با ساير اشياء كار ميكنيد مثل متغيير هاي نمونه آنها. در بسياري از حالات شما ممكن است نياز داشته باشيد به گرفتن زير مجموعه داخل يك عنصر نمونه گرفته شده به كمك تكرار و پيدا كردن دادهاي كه نياز داريد.براي مثال گرفتن شرح انگليسي براي يك محصول نمونه ، شما مي بايست از كد هاي زير استفاده كنيد.

```
String description = null;
List descriptions = product.getDescription();
ListIterator it = descriptions.listIterator();
while (it.hasNext()) {
  Description d = (Description) it.next();
  if (d.getLocale().equals(en_US)) {
    description = d.getDescription();
    break;
  }
}
```

اين نوع تكرارها زماني كه داده اي xml با API ها پردازش مي شوند لازم مي باشند و فقط مخصوص JAXB نمي باشند. آنها يك قسمت ضروي در پيمايش ساختارهاي درختي داده همانند xml مي باشد. ما شما را دعوت مي كنيم كه قابليتهاي كامل JAXP را در آدرس URL گفته شده در بخش قبلي، ديدن كرده و اين مفيد خواهد بود.

۴طولاني كردن ماندگاري JavaBeans

يك از نام گذاري هاي نادرست JavaXML API، نام گذاري Long Term JavaBeans Persistence بر روي API هاي مخصوص نگاشت xml به كامپونتهاي جاوا مي باشد. قابليتهاي آن شبيه به قابليتهاي JAXB بوده اما شيوه كار آن به اين صورت است كه به جاي كار با مقيد سازي الگو با كامپونتهاي جاوا براي نگاشت به xml كار ميكند.زماني كه كامپونتهاي جاوا(bean) بايد متدهاي get ,set براي دستيابي به هر صفت(property) ايجاد كنند، به كمك آن قادر به توسعه كامپونتهاي سريال شده(گرفته شده از جريان ورودي xml) خواهيم بود كه از سند xml بدون مقيد سازي الگو گرفته شده اند. همچنين ما را قادر به انجام عكس اين عمل يعني تبديل كامپونتهاي JavaBean به فرمت استاندارد xml ميكند.
اين APIها هم اكنون قسمتي از Java 2 Standard Edition (J2SE) شده و نياز به دانلو كردن كلاسهاي خارجي و اضافه كردن آنها در در مسير كلاسهاي برنامه نداريد. رابط اصلي اين APIها در جدول زير بهطور خلاصه نشان داده شده. اين كلاسها سبكي شبه به java.io.ObjectInputStream و java.io.ObjectOutputStream دارند و به جاي استفاده در فرمت باينري براي xml استفاده ميشوند.

Class name	Description
java.beans.XMLEncoder	Serializes a JavaBean as XML to an output stream.
java.beans.XMLDecoder	Reads in a JavaBean as XML from an input stream.

نوشتن JavaBean به داخل XML

مثال زير يك كامپوننت ساده JavaBean را با يك property به ما نشان ميدهد.

```
public class SimpleJavaBean {
  private String name;
public SimpleJavaBean(String name) {
    setName(name);
  }
// accessor
  public String getName() { return name; }
// modifier
  public void setName(String name) { this.name = name; }
}
```

همان طور كه مشاهده ميكنيد اين bean استاندارد javaBean براي دسترسي وتغيير property ها را با get, set ارائه كرده. با استفاده از كدهاي كوچك زير ميتوانيم اين bean را در داخل يك سند xml با نام simple.xml ذخيره كنيم.

```
import java.beans.XMLEncoder;
import java.io.*;
...
XMLEncoder e
  = new XMLEncoder(new BufferedOutputStream(
                  new FileOutputStream("simple.xml")));
e.writeObject(new SimpleJavaBean("Simpleton"));
e.close();
```

كد موجود در بالا يك XMLEncoder روي javax.io.BufferOutputStream ايجاد كرده و آنرا به داخل فايل simple.xml مي ريزد. سپس ما نمونه اي از شي SimpleJavaBean را به متد writeObject از encoder پاس ميكنيم و جربان را مي بنديم. محتواي فايل خروجي به صورت زير خواهد بود.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.0" class="java.beans.XMLDecoder">
  <object class="SimpleJavaBean">
```

```
    <void property="name">
      <string>Simpleton</string>
    </void>
  </object>
</java>
```

اینجا ما جزئیات نحوي xml را بیان نمیکنیم، زیرا شما نیاز به آموختن آنها در استفاده از این API ها ندارید.

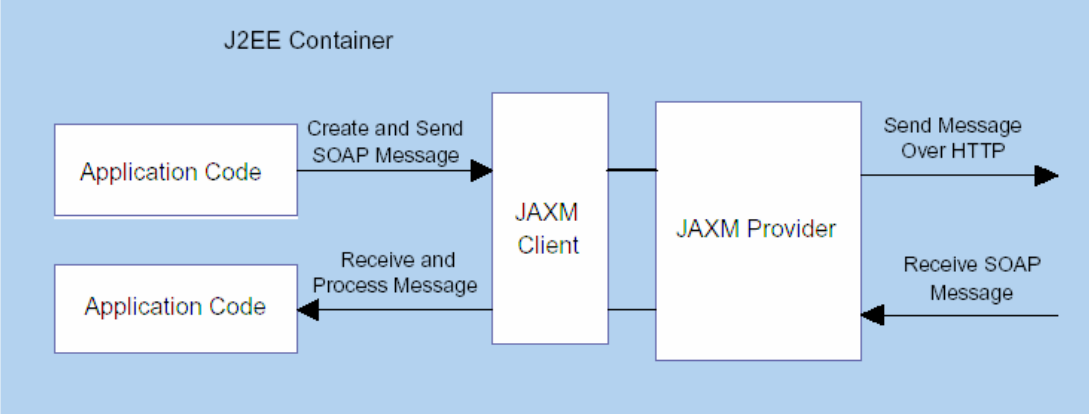
بازیابي JavaBean از XML
خواندن يك JavaBean ذخیره شده در xml به همان ساده گي ذخیره کردن آن میباشد. ما با استفاده از مثال SimpleJavaBean دو باره آن bean را خواهیم ساخت. کد هاي زیر این کار را انجام مي دهند.

```
XMLDecoder d
  = new XMLDecoder( new BufferedInputStream(
                    new FileInputStream("simple.xml")));
SimpleJavaBean result = (SimpleJavaBean) d.readObject();
d.close();
```

Decoder چگونه گي از نوع ساختن هر نوع bean ذخیره شده را با کامپوننت XMLDecoder را مي داند. این API مي توانند يك راه سریع و کم زحمت براي صادر کردن bean ها به xml براي استفاده دیگر ابزار ها و برنامه هاي کاربردي دیگر باشد. به خاطر بیاورید که شما قادر هستید که سندهاي xml شامل bean را به کمك XSLT به راحتي به سایر محیط ها و framework هاي توسعه وارد کنید!

JAXM ۵

نام گذاري JAXM براي Java API for XML Messaging که يك سري API هاي enterprise مخصوص جاوا براي استاندارد دستیابي به متدها و يك مکانیزم انتقال براي پیامهاي SOAP است. همچنین از SOAP با هر دو روش پیام رساني همگام و ناهمگام پشتیباني میکند.
مشخصات JAXM سرویس هاي گوناگوني را تعریف کرده که باید توسط تکمیل کننده (شرکت ایجاد کننده آن سرویس) آن JAXM ارائه شود. استفاده از هر کدام براي توسعه دهنده گان پوششي بر روي پیچیده گي هاي سیستمهاي پیام رساني است. دستیابي کامل به این سرویسها داخل معماري JAXM در شکل زیر نشان داده شده.



دو كامپوننت اصلي در معماري JAXM Client و Provider هستند. Client قسمتي از J2EE Web يا ظرف(container) EJB مي باشد كه قابليت دسترسي به سرويس هاي JAXM را در داخل برنامه شما فراهم مي كند.همچنين Provider ممكن است از روشهاي مختلفي ايجاد شده باشد و مسئول براي فرستادن و دريافت پيامهاي SOAP است. به كمك اين سازماندهي شما قادر به ارسال و در يافت پيامهاي SOAP انحصاراً با JAXM API خواهيد بود.
دو بسته در داخل JAXM API همان طور كه در جدول زير نشان داده شده وجود دارد. كامپونتهاي شما براي دستيابي به سرويسهاي JAXM از رابطهاي Connection و ConnectioFactory استفاده كرده و به همان روش شما مي بايست دستگير يك پيام در معماري Java Message Service (JMS) را بدست آوريد. بعد از گرفتن يك Connection شما ميتوانيد از آن براي ايجاد ساختار پيام SOAP و فرستادن آن به ميزبان راه دور از طريق HTTP استفاده كنيد.
همچنين در JAXM يك Servlet جاوا موجود است كه به كمك آن شما مي توانيد زماني كه نياز به دستگير كردن پيامهاي SOAP داخل شونده داريد از آن استفاده كنيد.

Package name	Description
javax.xml.messaging	Contains the ConnectionFactory and Connection interfaces and supporting objects.
javax.xml.soap	Contains the interface to the SOAP protocol objects, including SOAPEnvelope, SOAPHeader, and SOAPBody

ايجاد و استفاده از پيامهاي SOAP كمي پيچيده مي باشد و از JAXM براي ايجاد و دسترسي به سرويس هاي وب در J2EE استفاده مي شود. اطلاعات بيشتر را مي توانيد از http://java.sun.com/xml/jaxm دريافت كنيد.

JAX-RPC ۶

JAX-RPC براي اجراي فراخواني روال راه دور(Remote Procedure Call) توسط xml در جاوا است. با JAX-RPC اكثر مكانيزمهاي XML-RPC كه پايه اي براي SOAP هستند را مي توانيم استفاده كنيم. بطور مثال شما مي توانيد متدهاي يك كامپوننت bean را كه در ظرف EJB در حال اجرا مي باشد را به سرويس گيرنده هاي تحت جاوا يا ساير براي اجرا اشكار كنيد. بسته JAX-RPC يك از بسته هاي Java

XML Pack شده و اطلاعت به روز در اين مورد در http://java.sun.com/xml/jaxrpc موجود است.
اين نكته قابل توجه است كه SOAP با سرعت در حال رايجتر شدن نسبت به XML-RPC براي ايجاد سرويسهاي وب است. هم اكنون JAXM، پروتكل SOAP را كاملتر كرده و بلوغ بيشتري براي تكميل برنامه ها پيدا كرده و آينده JAX-RPC API تا حد زيادي نا معلوم است.

۷ JAXR

يك كامپوننت حياطي براي موفق تر كردن سرويسهاي وب بايد قادر به دستيابي و انتشار اطلاعات درباره سرويس هاي موجود در رجيستر هاي در دسترس سراسري باشد. اين رجيستري ها در واقع سرويس دهنده هاي مخصوص پيدا كردن ساير سرويس هاي وب هستند. هم اكنون چندين استاندارد محاسباتي در ضمينه رجيستر هاي سرويسهاي وب وجود دارد. معمول ترين اين رجيستري UDDI ebXML, مي باشند.
تفاوتهاي اندكي بين اين رجيستري ها از انواع مختلف وجود دارد و تلاشها بر اين است كه تنها يك سري API براي همه اين رجيستري ها استفاده شود. نتيجه اين طرح يك سري API با نام Java API for XML Registries (JAXR) بود.در JAXR يك لايه انتزاعي براي شناساي سيستم هاي رجيستري ايجاد گرديده، كه آنرا قادر به استاندارد سازي دستيابي به اطلاعات سرويسهاي وب از جاوا مي سازد.
انتظار ميرود كه JAXR شامل API هاي باشد كه پرسوجوهاي پيچيده رجيستري را را اجرا كرده و نتايج را براي ارسال و به روز رساني داده هاي خود به يك سيستم رجيستري مشخص ارسال كند. مزيت اصلي آن اين است كه شما تواناي دستيابي به محتواي رجيستري هاي ناهمگون را بدون داشتن فرمت خاص آن رجيستري ويژه داريد و از فرمت استاندارد جاوا استفاده مي كنيد.
فقط در JNDI اين امكان وجود دارد كه به طور پويا مي توانيد منابع را كشف كنيد. JAXR قادر است كه بطور پويا اطلاعات رجستري ها بر پايه xml را كشف كند.

قابليتهاي JAXR هم اكنون به طور عمومي در دست مطالعه و بررسي ايست و هنوز استاندارد كاملي براي رجيستري ها به وجود نيامده و ورژن هاي اوليه JAXR در آينده تغييرات عمده اي خواهد داشت.

فصل هفتم

EJB

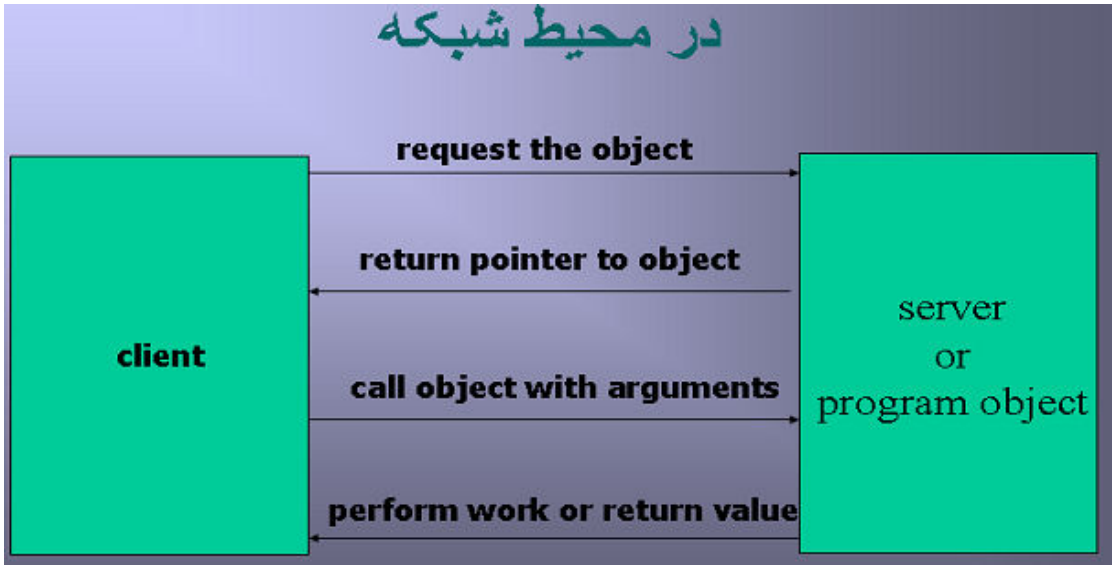
قبل از پرداختن به EJB لازم است کمی در مورد CORBA(Common Object Request Broker Architecture) برای شما توضیح دهیم.

CORBA

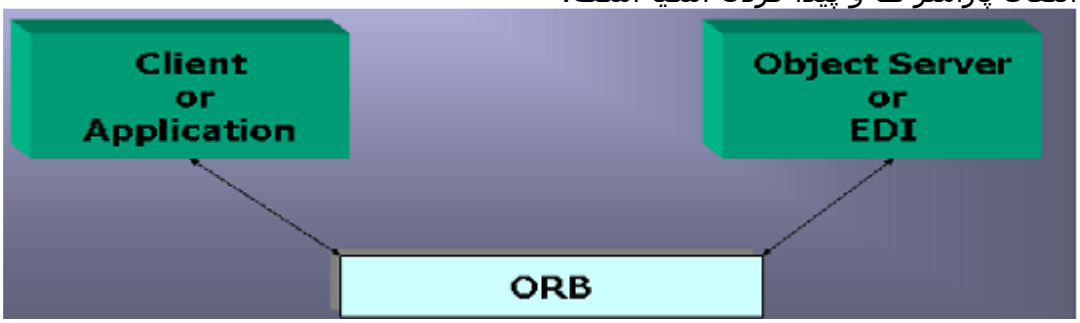
امروزه زمانه برنامه های تک کاربره و desktop به پایان رسیده است و برنامه های جدید باید دارای قابلیت هایی در سطح Enterprise باشند که در آنها هیچ محدودیتی روی تعداد کاربران و مکان آنها نداریم.این گونه برنامه ها نیاز به زیر ساخت هایی دارند که مانند زیر ساخت هایی برای اشیا موجود در شبکه و شناسایی آنها ، امنیت دسترسی به آنها و تراکنش ها و پیام رسانی متصل و غیر متصل و همچنین کار با رویدادها و پروتکل های کنترل مدیریتAuthentication و کنترل دسترسی باRule ها و موارد دیگر می باشد.
CORBA در واقع این زیر ساخت می باشد که مستقل از هر گونه سخت افزار وسیستم عامل و زبان برنامه نویسی بطور وسیعی پیشرفت نموده است.
شبیه این زیر ساخت به نامCOM+ که محصول شرکتMicrosoft و محیطی از قبل آماده شده می باشد که به راحتی بر رویWindows platform بکار می رود.
Microsoft هزینه و زمان زیادی را برای رسیدن به این قابلیت صرف نموده است که در ابتدا از ویندوز NT 3.1 با **Microsoft Transaction Server** و در4 NT با DCOM(Distributed COM) ودر ویندوز ۲۰۰۰ با COM+ ارایه داده است.

ORB (Object Request Broking)

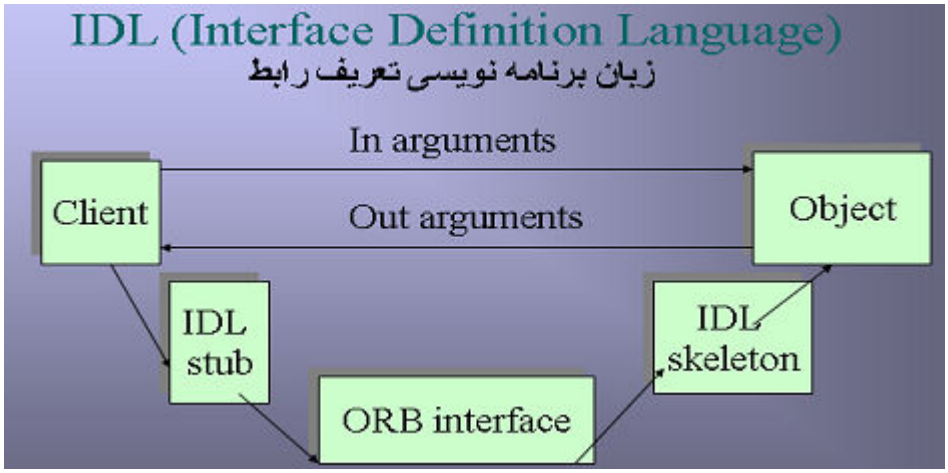
ORB واسط بین client و سرویس دهنده که وظیفه پیدا کردن متد و دادن اشاره گر ان به client و سپس ارسال فراخوانی متد به همراه پارامتر ها از طرف client به سرویس دهنده.



ORB مکانیزمی است برای انتقال Client Request به شی مقصد با رابط که در برنامه های توزیع شده شرح و مختصات فراخوانی متد را از دید Client مخفی می کند بطوری که به نظر می رسد تابع محلی را فراخوانی می کند ORB مسئول انتقال پارامتر ها و پیدا کردن اشیا است.



انتقال بین تعریفاتIDL CORBA و زبان برنامه نویسی نهایی که بطور اتوماتیک توسط IDL COMPILER انجام می شود.در Java به وسیله کامپایلر زمان اجرا JIT(Just in Time) ودر در+COM Stubsکه به عنوان یک PROXY عمل می کند و عمل تبدیل آدرس شبکه به اشاره گر را انجام می دهد.



Marshalling / Unmarshalling

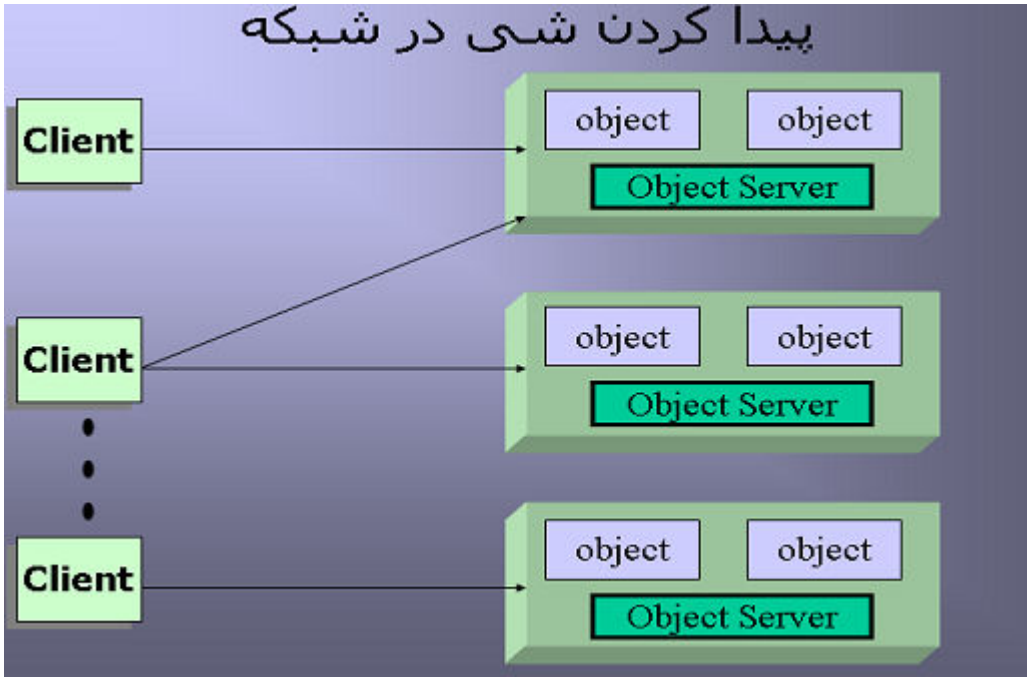
در دریافت یا انتقال آرگومان ها بین دو کامپوننت تبدیل آرگومان ها توسط Xml بطوری که بعدا بین اشیا شناسایی شوند را می گویند.
Argument -> Marshall -> Unmarshall -> Argument

نحوه پیدا کردن اشیاء در شبکه:
با دو روش
برپایه نام :
پیدا کردن اشیاء بر پایه نام آنها که عملکرد آن شبیه به روشی که در URL استفاده می شود.
بطور مثال در ویندوز NT ارجاع به یک فایل چنین است.

نام فایل\ نام پوشه اشتراکی \نام سرور\\

برپایه آدرس:
بر پایه مشخصات ویژه شی به همراه Socket
Socket = IP : Port Number
پس از اینکه شی پیدا شد و توسط ORB اشاره گری برگردانده می شد که به این عمل Bind کردن یا Binding گویند.

WWW.IRANMEET.COM



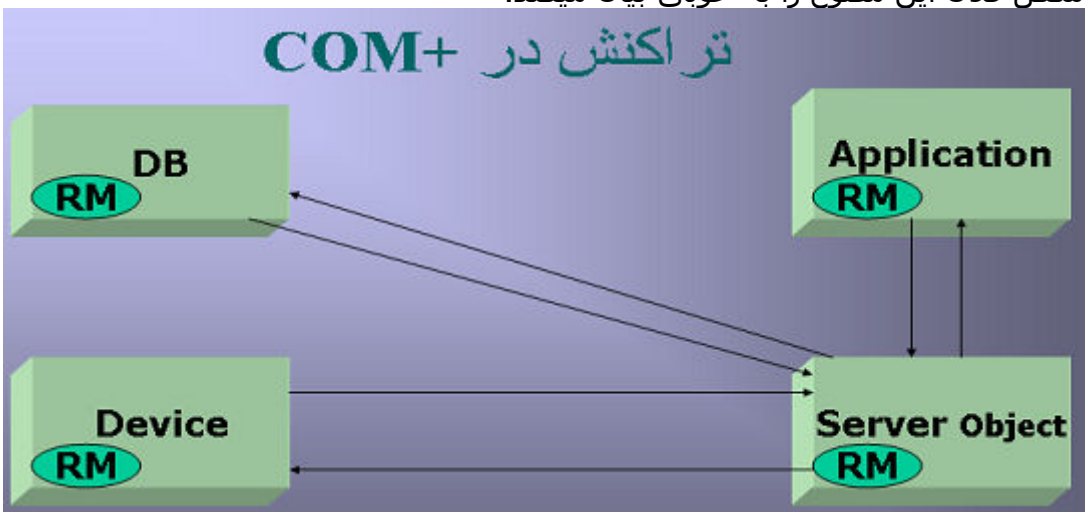
:Thread

اگر ما چند client که از اشیاء یکسان استفاده داشته باشیم به جای اجرای چند Process از اشیاء همه آنها را در داخل یک Process سیستم عامل گذاشته که قسمت کد و حافظه انها یکی میباشد و برای هرکدام از client ها یک نخ (thread) اجرا می کنیم به این ترتیب در منابع صرفه جوی میشود.
البته این اعمال توسء خود Framework انجام میشود.
یک مثال در ویندوز مثل برنامه IE (internet explorer) در زمانیکه ما چند صفحه را باز کرده ایم که قسمت کد وحافظه برنامه IE یکی بوده وبرای هرکدام از صفحات یک نخ اجرا میشود.

پشتیبانی تراکنش:

یکی از مهمترین نیازهای بعضی از اشیاء در منطق تجاری شرکت در تراکنش است بطور مثال فرض کنید که می خواهیم پولی را از حسابی به فردی پشت دستگاه بانک یا حسابی دیگر انتقال دهیم .
در این بین ممکن است یک سری اشیاء یا دستگاه (مثل خود پرداز ATM) وبانکهای اطلا عاتی در تراکنش شرکت کنند.
Resource Manager(RM) در واقع کنترل کننده نحوه شرکت آن موجودیت در تراکنش است که موفقیت یا عدم موفقیت آنرا برای کنترل کننده تراکنش بیان می کند
تراکنش عملی می باشد که برای اتمام آن می بایست تمام کسانی که در آن شرکت داشته اند موافق باشند.

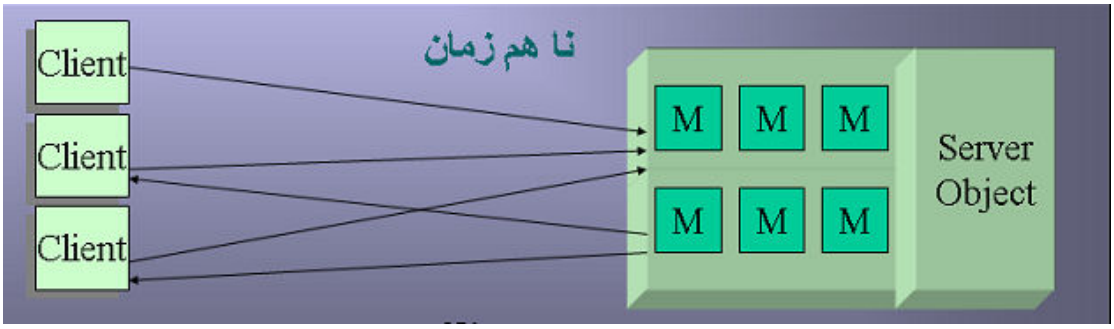
اگر حتی یکی از آنها مخالف باشند عمل Roll Back یا برگشت به عقب انجام می شود.
شکل فلان این مضوع را به خوبی بیان میکند.



صف پیغامها (Message Queue)

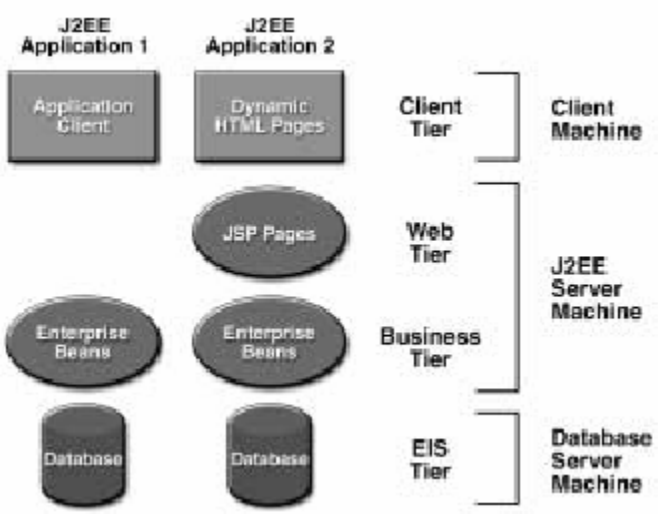
در اتصال همزمان client مستقیماً به سرویس دهنده وصل شده و به کمک ORB پاسخ را ردیافت میکند بطور مثال در اینترنت از یک سایت دینامیک وقتی صفحه ای را دریافت مکنیم .

در اتصال نا همزمان Offline Client بطور همزمان و مستقیم با سرویس دهنده کار نمی کند بلکه با گذاشتن پیغام خود در صف پیغامها (درخواست مربوط به شی مورد نظر و پارامتر های آن) در زمان مناسب متصل شده وپاسخ را میگیرد.
این روش در زمانی استفاده میشود که تعداد client ها ها زیاد است وسرویس دهنده توانایی پاسخ همزمان به همه را ندارد یا در جاهای که اتصال دائم سرویس دهنده و گیرنده وجود ندارد و همچنین زمان اتصال مشخص نمباشد .
M=Message



برنامه های توزیع شده چند طرفه(Multitiered) :

منطق برنامه تقسیم شده به اجزا وتوابع که هرکدام روی ماشین های جدا کار های مختلف را انجام داده که به هرکدام J2EE Tier میگویند(شکل ۱)



شکل۱

Client Tier: بر روی سرویس گیرنده قرار دارد و از سرور سرویس وب یا سرویس نرم افزار دریافت می کند

Web Tier: کامپوننت JSP که روی J2EE Server اجرا میشود

Business Tier: کامپوننت Bean که وظیفه کنترل منطق تجاری را دارد.

EIS :Enterprise Information System Tier نرم افزار نگهداری اطلاعات که یک سرویس دهنده پایگاه داده رابطه ای میباشد وبا Enterprise Resource Planning (ERP) ساختار آن مشخص میشود.
بطور کلی سیستمهای Multitier دارای سه طرف هستند که Web ,Business Tier یکسان فرض میشود.

J2EE Components

کامپوننتهای J2EE دارای توابعی هستند که خود را در سرور J2EE اسمبل (رجیستر) میکنند و دارای کلاسها و فایلهای هستند که با کامپوننتهای دیگر ارتباط برقرار میکنند .
J2EE دارای انواع کامپوننتهای زیر است

Client Application and Applets : تکنولوژی که در سمت سرویس گیرنده برای ارتباط با سرور

Java Server Page and Servlet : کامپوننت های مخصوص سرویس دهنده وب.
Enterprise Java Beans (EJB) : کامپوننت Bean که وظیفه کنترل منطق تجاری و Application Server .
این کامپوننتها به زبان جاوا نوشته وکامپایل شدهاند و تفاوت بین کامپوننتهای J2EE وکلاسهای استاندارد جاوا در این است که کامپوننتهای J2EE باید در سرور J2EE اسمبل (رجیستر) شده ونصب (deploy) شود ومدیریت آن به عهده J2EE Runtime Manager است.

Applets:

اپلت یک برنامه سمت سرویس گیرنده کوچک به زبان جاوا است و ماشین مجازی جاوا آن را روی مرورگر اینترنت اجرا میکند وباید Java Plug in نصب باشد . کامپوننتهای وب جاوا راه حل خوبی برای طراحی نرم افزار بصورت ماجولار و واضح میباشد زیرا راه ساده ای برای تقسیم برنامه به دو سمت سرویس دهنده وگیرنده و هردو به زبان جاوا طراحی می شود

Application Clients:

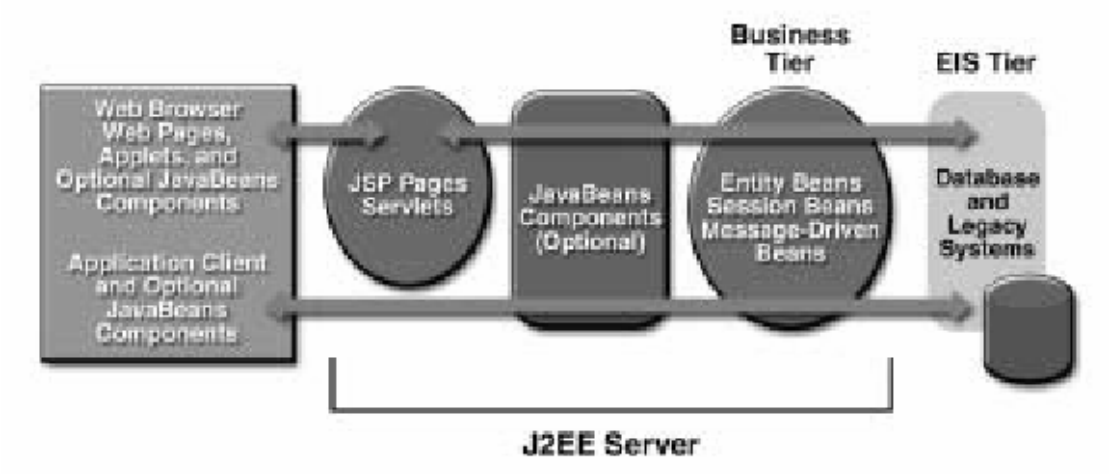
Application Clients های J2EE در روی ماشین سرویس گیرنده اجرا شده وبا اتصال به سرویس دهنده J2EE Enterprise Bean با پرتکل های ارتباطی یا استفاده از XML وهمچنین با ایجاد رابط کاربر GUI با Swing or Abstract window Toolkit (AWT) یا با خط فرمان همانند Ms Dos متصل میشود.
روش دیگر با http و ارتباط با Servlet اجرا شده روی jsp سرور.

JavaBeans Component Architecture

معماری کامونت JavaBeans برای مدیریت جریان داده بین برنامه Client یا Applet به یک پایگاه داده یا کلاسهای اجرای روی سرویس دهنده.
JavaBeans Component دارای متغیر های نمونه و متدهای Get, Set برای دستیابی به دادههای متغیر ها است (همانند Property ها در میکروسافت .NET) و منطق تجاری برنامه شما رادر کامپوننت کپسوله سازی میکند. سرویس گیرنده با اتصال و فراخوانی متد های داخل کلاس های کامپوننت انها را اگر مجوز داشته باشد اجرا می کند ونتیجه به ان فرستاده میشود .
کامپوننتها در داخل ظرفهای به نام Container که در قسمتهای بعد توضیح داده خواهد شد قرار میگیرند که وظیفه نگهداری وایجاد ومدیریت کامپوننتها رادارد .

Business Components:

کدهای تجاری آنها در حوزه های مختلف تجارت از قبیل بانکداری و مالی و...که درداخل Bean (کامپوننت J2EE) در سمت Business tier اجرا می شوند(درشکل ۲).
درشکل۲ نحوه دریافت داده های Client و پردازش آن وفرستادن به Enterprise Information System :Tier EIS وسپس فرستادن پاسخ به Client .



شکل۲

انواع Bean

۱ Session Bean :زمانی که Client به سرور J2EE متصل میشود برای آن اتصال فقط فعال میشود(بصورت گذرا) وبعد از قطع اتصال (Disconnect) آن اشیاءو داده های انها از بین میروند.

۲ Entity Bean :اشیاء داده ای که برعکس Session داده های آنها پایدار است وبعد از قطع اتصال دادههای آنها در جداول پایگاه داده ذخیره می شود

۳ Message Driven Bean : ترکیبی از مشخصات Session Bean وJava Message Service که با آنها کامپوننت از JMS بصورت ناهمگام پیغام دریافت میکند.
بیشتر بحث ما روی Session , Entity Bean میباشد.

J2EE Containers

به طور عادی نوشتن کد سرویس گیرنده Thin Client مشکل میباشد بخاطر اینکه بسیاری از کدها را برای مدیریت حالتهای مختلف وکنترل تراکنش ،چند نخی(multithreading) وغیره بسیار پیچیده میباشد.
معماری بر پایه کامپوننت ومستقل از ماشین J2EE این امکان را به ما میدهد تابه راحتی برنامه نوشته چون منطق تجاری برنامه در کامپوننتهای با قابلیت استفاده مجدد سازمان دهی شده و Container در زیر لایه کامپوننتها برای آنها سرویس مورد نیاز را برای مدیریت منابع،اشیاء ،تراکنشها وچند نخی را ارائه می دهد

Container Services:

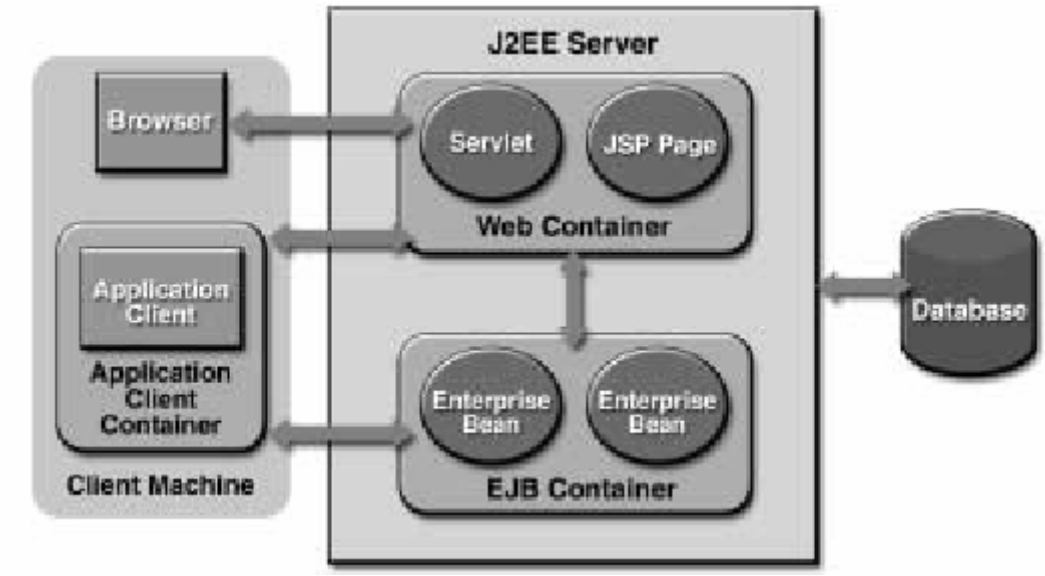
Containers رابط بین کامپوننتها و توابع ویژه در زیربنای سطح پائین مورد نیاز هستند پس یک Bean وب یا Enterprise باید به داخل یک Container اسمبل شود .
اسمبل کردن تنظیم مشخصات ویژه Container برای سرور J2EE و این تنظیمات شامل :

مشخص کردن مدل امنیتی بر پایه دستیابی با شناسایی شدن (authorized)
مشخص کردن مدل تراکنش که روابط ویژه بین متدها در تراکنش (یک مثال مثل انتقال پول از یک حساب به حساب گیرنده که توابع گیرنده وفرستنده وپایگاه داده هر دو داخل تراکنش قرار می گیرند ودر نهایت تراکنش زمانی موفق میشود که همه راضی باشند برای اطلاعات بیشتر میتوانید به کتابهای پایگاه داده مراجعه کنید.)
مشخص کردن JNDI (Java Naming and Directory Interface service) رابط ارتباطی برای سرویس فهرست راهنما که در شبکه های سازمانی موجود است(بطور مثال آدرس E mail شما در اینترنت به همراه @ یکی ازآن نوع سرویس ها است).
مشخص کردن ارتباط راه دور که برای ارتباط سرویس گیرنده و EJB(Enterprise Java Bean) (بطور مثال RMI(Remote Method Invoktion) و RPC (Remote Procedure Call) که بیشتر در +COM استفاده میشود.

معماری J2EE سرویس پیکره بندی نیز ارائه میکند بطور مثال سرویس گیرنده به پایگاه داده متصل شده با تنضیمات امنیتی و ارتباطی کهJ2EE با API پایگاه داده خود ا نجام می دهد.

انواع Container:

همان طور که درشکل۳ مشخص میباشد در سرویس دهنده J2EE سه نوع Container وب ، Client و EJB موجود است در نوع وب وظیفه مدیریت اجرای صفحات وب جاوا وکامپوننتهای Servelt رابه عهده دارد و در نوع EJB مدیریت اجرای کامپوننتهای Bean .
در نوع Client نیز مدیریت اجرای نرم افزار و Applet در روی Client (می توان گفت که Applet کالاسهای شبیه به ActiveX در تکنولوژی میکروسافت می باشد.)



شکل۳

بسته بندی Packaging:

میتوان کامپوننتهای J2EE را در داخل بسته های نرم افزاری جدا گذاشت و در جاهای دیگر از آنها استفاده کرد(با deploy کردن) بسته ها شامل فایلهای وابسته نیز مثل کلاسهای مفید وgif, html میباشند.
بسته ممکن است از یک یا چند Bean وب و Enterprise بوده به همراه شرح دهنده (descriptor) که یک سند XML که تنظیمات نصب (deploy) کردن کامپوننت درداخل ان نوشته شده می باشد.
این شرح میتواند شامل مشخصات امنیتی یا مدل ومتد های تراکنش وغیره که به راحتی قابل تغییر بوده و لازم به تغییر کد منبع نیست چون بصورت بیانی مباشند.

بسته ها با پسوند .EAR (Enterprise Archive) بسته های برای کامپوننتهایEnterprise هستند که فایلهای EAR یک استاندارد ازبایگانی فایل جاوا JAR (Java Archive) می باشند
بسته ها با پسوند .WAR (Web Archive) یک شرح دهنده نصب برای فایل کامپوننتها ی وب ومنابع وابسته می باشد . از نوع JAR در بسته های سمت Clientاستفاده میشود .

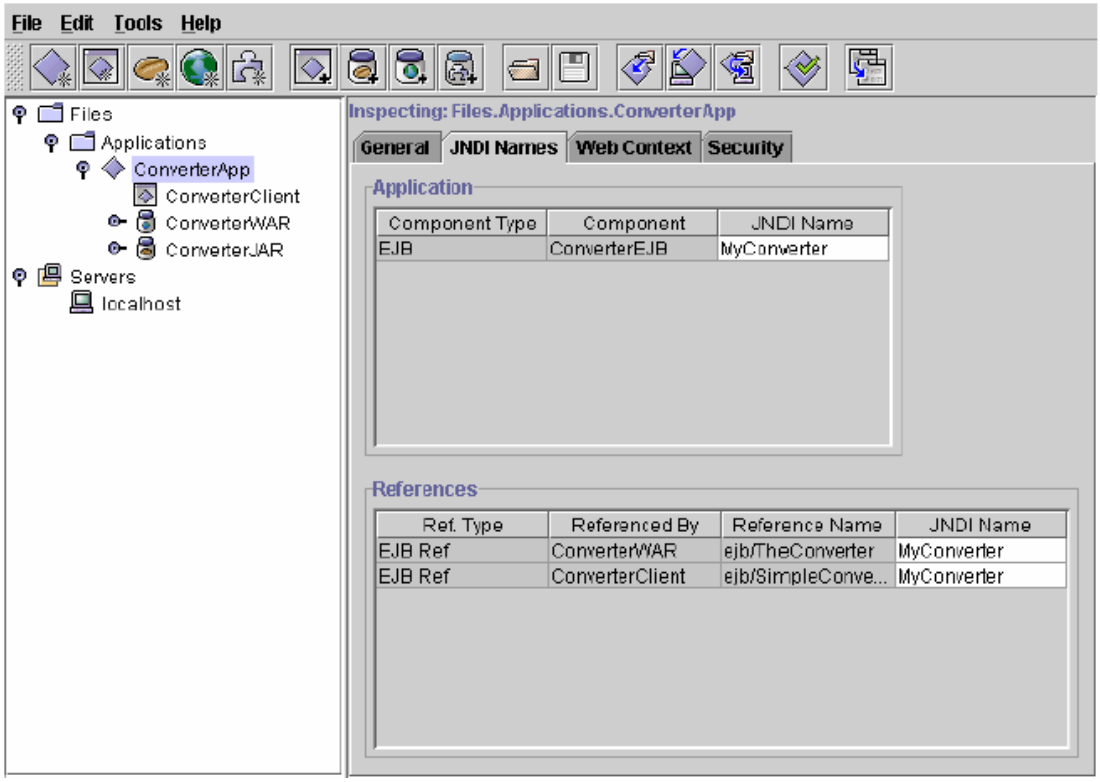
در مدل EAR میتوانید چند عدد از کامپوننتهای داخل EAR را بطور جدا با تنظیمات متفاوت نصب کنید.

نقشهای توسعه (Development Roles) :

تقسیم فرایند نصب وارتقاء (deploy) به نقشها مثل افراد وشرکتها برای اجرای قسمتهای جدا در سازمانهای بزرگ.خرید ونصب ابزارها وبرنامه ها و بعد اسمبل کردن برنامه یا کامپوننتهای J2EE که این مراحل می تواند توسط تیمهای جدا انجام شود مثلاً تیمی برای خرید یا توسعه EAR آماده وتیمی برای تبدیل فایلهای EAR,WAR,JAR به یک برنامه کاربردی J2EE .

Enterprise Bean Developer: فرد یا سازمان که این کارها را انجام میدهد
نوشتن وکامپایل کد منبع ، نوشتن شرح دهنده (deploy) و بسته بندی فایلهای class. وشرح دهنده به فایل EAR JAR .
Web Component Developer: فرد یا سازمان که کارهای نوشتن وکامپایل کد منبع Servelt، نوشتن فایل های JSP, Html ، ، نوشتن شرح دهنده (deploy) و بسته بندی فایلهای .Html .JSP, class, وشرح دهنده به فایل WAR.

Application Deployment Tool: یک ابزار برای deploy کردن به نام deploytool که امکانات اسمبل کردن وتغییر برنامه های J2EE را دارد ودارای دو ورژن خط فرمان و گرافیکی میباشد که شکل نوع گرافیکی را آن را در زیر مشاهده می کنید(شکل۴). در مدل گرافیکی Wizard ها ی مختلف برای بسته بندی کردن ونصب دارد.
درنوع خط فرمان انواع Script ها را برای مدیریت سرویس دهنده J2EE می توان نوشت مثل ایجاد کلید برای کامپوننت ، بسته بندی کامپوننت ، اجرای کامپوننت در سرور J2EE یا در کلاینت ، فعال ویا غیر فعال کردن سرورJ2EE وسایر اعمال.



Enterprise JavaBeans

برای اموختن طرز کار EJB با یک مثال ساده شروع میکنیم شی PersonServer یک شی توزع شده تجاری با رابط Person که دارای دو متد getName() و getAge() البه در برنامه های واقعی این متد ها بیشتر میباشد

```
public interface Person {

public int getAge() throws Throwable;

  public String getName() throws Throwable;
}
```

حال به تعریف کلاس PersonServer که از رابط Person استفاده میکند می پردازیم .

```
public class PersonServer implements Person {
  int age;
  String name;
   public PersonServer(String name, int age){
     this.age = age;
     this.name = name;
  }
  public int getAge(){
     return age;
  }
  public String getName(){
     return name;
  }
}
```

Now we need some way to make the `PersonServer` available to a remote client. That's the job of the `Person_Skeleton` and `Person_Stub`. The `Person` interface describes the concept of a person independent of implementation. Both the `Person-Server` and the `Person_Stub` implement the `Person` interface because they are both expected to support the concept of a person. The `PersonServer` implements the interface to provide the actual business logic and state; the `Person_Stub` implements the interface so that it can look like a `Person` business object on the client and relay requests back to the skeleton, which in turn sends them to the object itself. Here's what the stub looks like:

```
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;
```

Copyright (c) 2001 O'Reilly & Associates

حلا ما باید کاری کنیم تا PersonServer برای سرویس گیرنده قابل دسترس باشد این کار با Person_Skeleton و Person_Stub انجام میشود . PersonServer و Person_stub رابط Person را کامل میکنند و سرویس گیرنده به کمک Person_Stub برای PersonServer تغاضا می فرستد

```
import java.io.ObjectInputStream
import java.io.ObjectOutputStream;
import java.net.Socket;

public class Person_Stub implements Person {
  Socket socket;
   public Person_Stub() throws Throwable {
    socket = new Socket("localhost",9000);
  }
  public int getAge() throws Throwable {
    // When this method is invoked, stream the method name to the
    // skeleton.
    ObjectOutputStream outStream =
       new ObjectOutputStream(socket.getOutputStream());
    outStream.writeObject("age");
    outStream.flush();
    ObjectInputStream inStream =
       new ObjectInputStream(socket.getInputStream());
    return inStream.readInt();
  }
  public String getName() throws Throwable {
     // When this method is invoked, stream the method name to the
     // skeleton.
     ObjectOutputStream outStream =
        new ObjectOutputStream(socket.getOutputStream());
     outStream.writeObject("name");
     outStream.flush();
     ObjectInputStream inStream =
        new ObjectInputStream(socket.getInputStream());
     return (String)inStream.readObject();
  }
}
```

به کمک جریان ورودی و خروجی و Socket این امکان را فراهم میکنیم تا از پورت ۹۰۰۰ از Localhost توابع getName() , getage() را فراخوانی یا آنها را مقدار دهی کنیم

وقتی که یک متد روی Person_Stub فراخوانی میشود یک token ازنوع رشته ای ایجاد شده و به skeleton جریان داده میشود . این توکن (نام تابعی که فراخوانی شده) روی شی تجاری مورد نظر جستجو واجرا میشود و نتیجه توسط skeleton به سرویس گیرنده ارسال میشود حال به کدهای skeleton توجه کنید.

```
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;
```

```
import java.net.Socket;
import java.net.ServerSocket;

public class Person_Skeleton extends Thread {
  PersonServer myServer;
   public Person_Skeleton(PersonServer server){
     // Get a reference to the business object that this skeleton wraps.
     this.myServer = server;
  }
  public void run(){
    try {
     // Create a server socket on port 9000.
     ServerSocket serverSocket = new ServerSocket(9000);
     // Wait for and obtain a socket connection from stub.
     Socket socket = serverSocket.accept();
     while (socket != null){
        // Create an input stream to receive requests from stub.
        ObjectInputStream inStream =
           new ObjectInputStream(socket.getInputStream());
        // Read next method request from stub. Block until request is
        // sent.
        String method = (String)inStream.readObject();
        // Evaluate the type of method requested.
        if (method.equals("age")){
           // Invoke business method on server object.
           int age = myServer.getAge();
           // Create an output stream to send return values back to
           // stub.
           ObjectOutputStream outStream =
              new ObjectOutputStream(socket.getOutputStream());
           // Send results back to stub.
           outStream.writeInt(age);
           outStream.flush();
        } else if(method.equals("name")){
           // Invoke business method on server object.
           String name = myServer.getName();
           // Create an output stream to send return values back to
           // the stub.
           ObjectOutputStream outStream =
              new ObjectOutputStream(socket.getOutputStream());
           // Send results back to stub.
           outStream.writeObject(name);
           outStream.flush();
```

```
        }
      }
    } catch(Throwable t) {t.printStackTrace();System.exit(0); }
  }
  public static void main(String args [] ){
    // Obtain a unique instance Person.
    PersonServer person = new PersonServer("Richard", 36);
    Person_Skeleton skel = new Person_Skeleton(person);
    skel.start();
  }
}
```

Person_Skeleton درخواست stub را به شی تجاری مورد نظر یعنی PersonServer میفرستد
اساساً Person_Skeleton تمام وقت خود را برای گرفتن درخ واست سروس گیرنده صرف میکند .
Person_Skeleton نتیجه گرفته از شی را به صورت جریان داده به stub میفرستد .

حالا نوبت ایجاد کد های ساده لازم برای سرویس گیرنده است.

```
public class PersonClient {
  public static void main(String [] args){
    try {
      Person person = new Person_Stub();
      int age = person.getAge();
      String name = person.getName();
      System.out.println(name+" is "+age+" years old");
    } catch(Throwable t) {t.printStackTrace();}
  }


}
```

برنامه client چگونه گی استفاده از stub را نشان می دهد وبا کمک Proxy وتبدیل آدرس فیزیکی به اشاره گردر شبکه این کار عملی میشود .

RMI (Remote Method Invocation)

Rmi درواقع کل این فرآیند میباشد که معادل ORB در CORBA و RPC در COM+ میباشد RMI پایه سیتمهای اشیاء توزیه شده J2EE میباشد

در پرتکولهای اشیاءتوزیع شده مثل CORBA, DCOM , Java RMI یک سری ساختار وابزارها برای ایجاد وکنترل skeleton, stub وجود دارد همچنین آنها میتوانند اعمال پاس کردن پارامترها handle کردن خطاها وسایر سرویسها مثل تراکنش و امنیت را در ضمینه خود ارائه دهند.

مدل کامپوننتها (Component Models)

Enterprise JavaBeans دارای مدل مشخص و ویژه خوددر سمت سرور می باشد که از یک سری کلاسها و رابطهای پکیج Java.ejb استفاده میکند.
توسعه دهنده گان میتوانند برای ایجاد و اسمبل و deploy کردن کامپوننتها ارز ویژگی های EJB استفاده کنند .

JavaBeans یک مدل کامپوننت برای بسته بندی اشیائ میباشد اما مثل EJB مدل کامپوننت Server side نیست و ویزگیهای کنترل اشیاء صف پیغامها و سایر موارد گفته شده را ندارد.

یک کامپوننت برای کاربرد خاص توسعه داده می شود و آن یک برنامه نیست. بطور مثال در داخل یک دکمه یا صفحه گسترده در داخل یک نرمافزار با GUI (رابط گرافیکی کاربر) برطبق قواعد JavaBeans استفاده میشود امادر مدل EJB کامپوننت(شی تجاری مشتری) باید به داخل یک EJB Server وارد شده ومدیریت آن به عهده سرور و نه برنامه کاربردی می باشد .
سایر مدلهای کامپوننت جاوا شامل Servelt , JSP (ASP در .NET) و Applets (ActiveX در .NET) می باشند.

مانیتور تراکنش کامپوننتها CTM (Component Transaction Monitors)

CTM همزمان با پلاتفرم اشیاءتوزیع شده و ORB رشد کرد و از مانیتورهای TP (Transaction processing) استفاده می کند .TP اول در سیستم عاملها برای برای مانیتورینگ برنامه برنامه های تجاری استفاده شد سپس برای مدیریت محیط اشیاءتجاری استفاده شد

دراین روش client درخواست خود را به TP میدهد و TP آن روال را به همراه پارامترهایش برای سرویس گیرنده اجرا میکند .تفاوت اساسی بین RPC , RMI دراین است که RPC از برنامه های بر پایه روال استفاده میکند و Procedural base است ولی RMI از سیتم اشیاء توزیع شده استفاده میکند و شی گرا می باشد.

CTMs در واقع پیوند ORB و TP میباشد که تراکنش و مدیریت منابع(RM) را نیز پشتیبانی می نماید البته پیچیده تر هم شده .
اسراتژی مدیرت منابع شامل سرکشی منابع ، فعال سازی و جابهجای ها میباشد و توسه دهنده گان بدون آن مجبور هستند که خود کدهای کنترلی بنویسند.

این را میتوان با مثال شرح داد همان طوری که ما در پایگاه داده رابطه ای محتوای جداول ، کلید اصلی ، شاخصها و روالهای ذخیره شده راایجاد میکنیم و شما

الگوریتمهای شاخص سازی و مدیریت Cursor SQL را توسعه نمدهید بلکه از DBMS پایگاه داده که ساخت شرکتهای متخصص این کار هستند استفاددده میکنید .
در این جا هم CTMs همان قابلیتهای مورد نیاز را میدهد .
کاموپیننتهاي سمت سرور بر پایه مشخصات ویژه خود هستند و رابطه بین کامپوننتهاي سمت سرور و CTMs شبیه رابطه بین CD و CD-ROM میباشد.
همچنین رابطه بین CTMs و مدل کامپوننت مثل رابطه ریل راه آهن و ترن قطار میباشدبطوري که راه آهن محیط ترن را مدیریت میکند مثل همزماني یا کنترل ترافیك و توازن بار.

پیغامدهی غیرهمزمان (Asynchronous Messaging)

به نرم افزار ها این امکان رامیدهد که اطلا عات را با پیغام تبادل کنند. منظور ما از پیغام یک بسته داده های تجاری در شبکه که همراه سرآیند های آدرس دهی است که البته فرستنده وگیرنده یا گیرنده ها باید مشخص باشند و از MOM Message oriented middleware)) استفاده می کند .
MOM تضمین میکند که پیغامها بین نرم افزارها توزیع وجا به جا میشوند و MOM تحمل بار(fault tolerance)،ایجاد توازن درخواست (load balancing) و پایداری پیغامها رابرای پیغامهای نرم افزار Enterprise پشتیبانی می کند .
به همن منظور دارای API های مخصوص به خود که میتوانید دربرنامه خود استفاده کنید

Java Message Service

Java Message Service یک MOM محصول شرکت SUN با نام اختصاری JMS که API های آن را در J2EE استفاده شده و میتوانید برای تبادل پیغام باسایر محصولات مثل Microsoft Message Queue از انها بهره ببرید.

Message Driven Beans

در EJB 2.0 یک نوع کامپوننت جدید ابداء شد که Message driven Beans نامیده شد که همان کامپوننتهای استاندارد JMS هستند.
آنها می توانند پیغامهای JMS را را ارسال یا دریافت نمایند و با کامپوننتهای RMI تعامل داشته باشند همچنین در تراکنشهای سمت سرور شرکت کنند. از این کامپوننتها باید با توجه به منطق ونیاز تجاری استفاد ه کر دکه قطعآ تاثیر مطلوبی در برقرای ارتباط با سایر سیتسم شبیه به خود دارد.

Microsoft .NET Frame work

میکروسافت اولین تولید کننده CTM بود که با نام Microsoft Transaction Server یا به اختصار MTS نامیده شد که در سال 2000 نام آن به +COM تغیر کرد.
+COM بر پایه COM (Component Object Model) بوده و MTS در سال ۱۹۹۶ تولید شد و توسعه دهنده گان کامپوننتهاي COM را بدون مشخصات سطح سیستمي مینوشتند .
+COM به طور خودکار ازهمه کنترلهاي لازم در CTMs پشتیباني میکرد و امروزه جزئي از .NET Framework. شده که در آن اشیاء COMتبدل به اشیاء .NETشده که در آنها به CLR (Common Language Runtime) براي اجرا شدن نیاز دارتد شبیه Java Virtual Machine .

هر چند که .NET Framework. بسیار ي مزایا را ارئه داده این استاندارد فقط بر روي پلات فرم میکروسافت اجرا شده و درسازمان هاي بزرگ امروزي که انواع مختلف سیستم عاملها و پرتکولهاي شبکه وجود دارد قابلیت اجرا نداشته .
دلیل دیگر رقابت سایر شرکتها با میکروسافت در ضمینه فناوري اطلا عات که محصولات آنها ارزانتر میباشد.

EJB and CORBA

زماني میکرو سافت درحال توسعه CTMs خود بود سایر شرکتها بر روي یك استاندارد آزاد(مستقل از سیستم) کار میکردند که CORBA نام گرفت.

CORBA چون توسط شرکتهاي وافراد مختلف و نه مهندسان پرهزینه(مثل مهندسان ماکروسافت) طراحي شده داراي قابلیتهاي بالا با سطح مشخصات براي انواع محیطها میباشد و اکنون توسط OMG (Object Management Group) کساني که CORBA راتوسعه دادند استاندارد سازي میشود .
CORBA براي برقرار ي ارتباط بین سیستمهاي ناهمگون Object SOAP(Simple Access Protocol) را ابداء کرده که قسمتي از CORBA IIOP(Internet Inter Operability Protocol) میباشد .

در سال ۱۹۹۷ شركت سان استاندارد Enterprise JavaBeans بر اساس CORBA و زبان Java توسعه داد يك سري API پايه براي اينكار طراحي كرد كه بانام EJB 1.0 بيرون آمد.

Enterprise JavaBeans

انواع Beans :
Session, Entity, Message Driven كه در آنها Session, Entity بر پايه RMI بوده و Message Drivenدر EJB 2.0 براي كار كردن راحتر با JMS طراحي گرديد.

۱ Session Bean :زمانی که Client به سرور J2EE متصل میشود برای آن اتصال فقط فعال میشود(بصورت گذرا) وبعد از قطع اتصال (Disconnect) آن اشیاءو داده های انها از بین میروند.

۲ Entity Bean :اشیاء داده ای که برعکس Session داده های آنها پایدار است وبعد از قطع اتصال دادههای آنها در جداول پایگاه داده ذخیره می شود

۳ Message Driven Bean : ترکیبی از مشخصات Session Bean وJava Message Service که با آنها کامپوننت از JMS بصورت ناهمگام پیغام دریافت میکند .

بیشتر بحث ما روی Session , Entity Bean میباشد.

يك راه فهم bean بررسي نحوه كامل كردن كد آنها ميباشد .براي Session , Entity شما بايد يك رابط تعريف كرده به همراه يك كلاس Bean و كليد اصلي.
بطور اساسي دو نوع رابط كامپونت وجود دارد Local , Remote .

:Remote Interface

براي دسترسي به يك متد در داخل EJB container از بيرون از آن ظرف . رابط راه دور بايد javax.ejb.EJBObject را با extend دركد خود توسعه دهد وهمچنين Java.rmi.Remote رابا Import داخل برنامه خود كند .

:Local Interface

براي دسترسي به يك متد در داخل EJB container از داخل آن ظرفتوسط يك Bean ديگر درداخل همان فضاي آدرس . اين قابليت در EJB 2.0 ارائه شد كه ارتباط bean ها را برقرار مي كرد بدون هزينه بالا سري براي اشياء توزيع شده.
. رابط راه دور بايد javax.ejb.EJBLocalObject را با extend دركد خود توسعه دهد

: Local Home Interface

تعريف كننده توابع چرخه زندگي Bean هاي كه توسط ساير beanهاي ديگر درداخل همان فضاي آدرس آن EJB قابل استفاده اند . اين قابليت در EJB 2.0 ارائه شد كه ارتباط bean ها را برقرار مي كرد بدون هزينه بالا سري براي اشياء توزيع شده.
. رابط راه دور بايد javax.ejb.EJBHomelObject را با extend دركد خود توسعه دهد

Bean Class

كلاسهاي Session , Entity beans كامپوننتهاي تجاري و متدهاي چرخه اشياء راكامل ميكنند.
براي session bean بايد javax.ejb.SessionBean استفاده كرد همچنين براي entity bean بايد javax.ejb.EntityBean استفاده كرد .
هر دو Session , Entity beans در كد خود javax.ejb.EJBHomelObject را با extend توسعه ميدهند.

Message driven bean در EJB 2.0 از رابط كامپوننت همانند Session , Entity beans استفاده نميكنند چون از بيرون EJB container قابل دسترس نيستند و فقط شامل يك متد ميباشد onMessage() كه به وسيله خود container فراخواني ميشود يعني درزماني كه يك پيغام براي آن bean مي رسد .

Message driven bean رابطهاي javax.ejb.MessageDrivenBean و javax.ejb.MessageListener رااستفاده ميكند.رابط MessageListener براي برقراي ارتباط bean و سرويس JMS ميباشد .
Message driven bean مانند Session , Entity beans در كد خود javax.ejb.EJBHomelObject را با extend توسعه ميدهد.

كلي اصلي (Primary Key)

يك كلاس بسيار ساده ميباشد كه يك اشاره گر به پايگاه داده ميدهد و فقط entity bean به آن نياز دارد و از java.io.Serializable استفاده ميكند.

در EJB نياز به رابط محلي نداريد چون شما ميدانيد كه در توسعه Enterprise Bean فقط نياز به ارتباط با سرويس گيرنده را ه دور داريد پس لازم نيست در رابط محلي RemoteException تعين و throw كنيد.
سرويس گيرنده بطور مستقيم با كلاس bean تعامل ندارد و با stub به ظرف آن EJB متصل ميشود.
تعامل بين Enterprise Bean و سرويس دهنده آن كه به كمك container يا ظرف مديريت شده و مسئول ارائه رابط بين bean و سرور ميباشد .
container وظيفه ايجاد يك نمونه از bean و اطمينان از ذخيره شدن آن درسرور دارد و ابزار هاي براي آن توسط شركتهاي مختلف توليد شده . اين ابزارها يك نگاشت بين entity bean وركورد ذخيره شده در پايگاه داده ايجا ميكند .

حال بررسي يك مثال عملي

وقتي درمورد رابط Remote صحبت ميكنيم در Cabin EJB ما ميخواهيم تركيب كنيم نام تجاري عادي را با كلمه Remote براي مثال نام رابط راCabinRemote interface ميگذاريم وهمچنين براي رابطهاي محلي و خانه گي نامهاي CabinLocal interface وCabinHome interface انتخاب ميكنيم.
در نهايت نام كامپوننت نيز CabinBean مي شود.

:Remote Interface

حال كد رابط يك entity bean يا stateful Enterprise bean را باهم كامل ميكنيم .
ما فرض ميكنيم كه CabinEJB يك كابين كشتي در سفر درياي مورد نظر ماست وبا رابط راه دور كه كه داراي چهار متد با قابليت كنترل استسنا با extend كردن javax.ejb.EJBObject در زير مشاهده ميكنيد.

```
import java.rmi.RemoteException;

public interface CabinRemote extends javax.ejb.EJBObject {

  public String getName() throws RemoteException;

  public void setName(String str) throws RemoteException;

  public int getDeckLevel() throws RemoteException;

  public void setDeckLevel(int level) throws RemoteException;

}
```

متد اول ودوم براي كاركردن با نام كابين و متد دوم براي سطح آن كابين نسبت به دكل كشتي . البته شما ميتوانيد متدهاي ديگري نيز اضافه كنيد.

:Remote home interface

وظيفه Remote home interface تعين متد هاي چرخه زندگي كه توسط client از entity , session bean براي اشاره به enterprise bean استفاده ميكند .
نام آنرا CabinHomeRemote گذاشته ايم.

```
import java.rmi.RemoteException;

import javax.ejb.CreateException;

import javax.ejb.FinderException;

public interface CabinHomeRemote extends javax.ejb.EJBHome {

  public Cabin create(Integer id)

     throws CreateException, RemoteException;

  public Cabin findByPrimaryKey(Integer pk)

     throws FinderException, RemoteException;
```

```
}
```

تابع create() وظيفه مقدار دهي يك نمونه از اين شي bean دارد . اگر نرم افزار شما به متد create()
ديگري نياز دارد آنرا با يك آرگمان ورودي جدا بسازيد .
به علاوه متد findByPrimaryKey() كه براي جستجو با كليد است شما آزاد هستيد سايز متدهاي مورد نياز را تعريف كنيد براي مثال findByship() كه تمام كابين ها در يك كشتي را برميگرداند . چنين متدها ي جستجو و نظير آنها فقط رد entity bean استفاده ميشود و در session , message driven استفاده نمي شوند.

:Bean Class EJB 2.0

دراين جا يك entity bean واقعي را داريم البته در قسمتهاي بعد آنرا كامل خواهيم كرد و مي خواهيم نشان دهيم كه چگونه اين قسمتها با هم تركيب ميشود.

```
import javax.ejb.EntityContext;
public abstract class CabinBean implements javax.ejb.EntityBean {
   // EJB 1.0: return void
  public CabinPK ejbCreate(Integer id){
      setId(id);

       return null;
  }
  public void ejbPostCreate(int id){
     // do nothing
  }
   public abstract String getName();
  public abstract void setName(String str);
   public abstract int getDeckLevel();
  public abstract void setDeckLevel(int level);
   public abstract Integer getId( );
  public abstract void setId(Integer id);
   public void setEntityContext(EntityContext ctx){
     // not implemented
  }
  public void unsetEntityContext(){
     // not implemented
  }
  public void ejbActivate(){
     // not implemented
  }
  public void ejbPassivate(){
     // not implemented
  }
```

```
  public void ejbLoad(){
     // not implemented
  }
  public void ejbStore(){
     // not implemented
  }
  public void ejbRemove(){
     // not implemented
  }
}
```

CabinBean به صورت انتزاي داراي متدهاي براي براي دستيابي و به روز رساني EJB در حالت پايدار ميباشد .
نكته قابل توجه اين جاست كه در كلاس ما فيلدهاي براي نگهداري اطلا عات نداريم و اين به دليل است كه ما در حال كار با entity bean مديريت شده توسط ظرف EJB 2.0 آن هستيم .
در اين روش در bean فقط تعريفهاي دستيابي متدها موجود است البته در ذشد هاي ديگر مثل session ,... چنين چيزي نخواهيد ديد .

:Bean Class EJB 1.0

كد هاي زير براي همان كلاس بالا امادر EJB 1.0 كه فيليد ها درآن تعريف شده ميباشد.

```
import javax.ejb.EntityContext;
public class CabinBean implements javax.ejb.EntityBean {
   public Integer id;
  public String name;
  public int deckLevel;
   // EJB 1.0: return void
  public Integer ejbCreate(Integer id){
     setId(id);
     return null;
  }
  public void ejbPostCreate(Integer id){
     // do nothing
  }
   public String getName(){
     return name;
  }    public void setName(String str){
     name = str;
  }
   public int getDeckLevel(){
     return deckLevel;
  }
```

```
public void setDeckLevel(int level){
   deckLevel = level;
}
public Integer getId( ){
    return id;
}
public void setId(Integer id){
    this.id = id;
}
public void setEntityContext(EntityContext ctx){
   // not implemented
}
public void unsetEntityContext(){
   // not implemented
}
public void ejbActivate(){
   // not implemented
}
public void ejbPassivate(){
   // not implemented
}
```

مقايسه كدهاي EJB 1.0 , 2.0 :

متد هاي get , set براي نام و سطح دكل با رابط Remote اين كلاس يعني CabinRemote سازگار ميباشند .
اين متدهاي get , set فقط براي نرم افزار سرويس گيرنده و ساير متدها فقط براي ظرف EJB و كلاس Bean قابل ديد هستند.

متدهاي ejbPostCreate() , ejbCreate() براي مقدار دهي كلاس bean وقتي كه يك ركورد كابين به پايگاه داده اضافه ميشود مي باسند .
هفت متد آخر كلاس CabinBean ورژن EJB 2.0 در رابط javax.ejb.EntityBean تعريف شده ميباشند و متدهاي براي مديريت حالتها مي باشند. وقتي كه bean اجرا ميشود ظرف متد مربوط به آن رخداد را اجرا ميكند . براي مثال ejbRemove() به نكته اشاره ميكند كه يك entity bean داده هايش بايد از پايگاه داده حذف شود.

ejbLoad() , ejbStore() در حالتي اجرا مي شود كه مي خواهيم داده در پايگاه داده خوانده يا نوشته شود. هنگام فعال وغير فعال شدن فرآيند مربوط به CabiBean (گرفتن يا آزاد سازي منابع) متدهاي ejbPassivate() , ejbActivate() اجرا مي شوند.
setEntityContext() برقراي كردن ارتباط به EJB container كه اجازه گرفتن اطلاعات درباره خود bean را درظرف ميدهد .
unsetEntityContext() توسط ظرف براي گرفتن فضاي حافضه اشغال شده توسط bean درهنگام خروج از ظرف مي باشد .

Primary Key
يك اشاره گر كمك كننده به پيدا كردن دادها كه شرحي ازيك ركورد يا موجوديت داخل پايگاه داده است وبا متد findByPrimaryKey() از رابط home محل يك موجوديت پيدا ميشود. در Cabin EJB داده بانوع java.lang.Integer براي كليد اصلي استفاده ميشود

تفاوت entity bean با session beans

CabinBean مثال ما يك entity bean ميباشد اما يك session bean خيلي با آن متمايز نيست. درآن بايد SessionBean به جاي EntityBean دركد extend شودو متد ejbCreate() آنرا مقدار دهي مي كند اما ejbPostCreate() ندارد .

session bean داراي متدهاي ejbStore(), ejbLoad() نيست چون اين نوع bean ها نامانا هستند . session bean داراي متد setSessionContext() ميباشد اما unsetSessionContext() ندارد . متد ejbRemove() در session bean براي ثبت اتمام ارتباط و مثل كاربرد آن در entity bean نيست .
درsession bean كليد اصلي نيز وجود ندارد چون قرارنيست به كمك آن به پايگاه داده ارجاع كنيم ودرحالت كلي اطلاعات session bean ناماندگار است .

تفاوت entity bean با message driven beans

اين نوع bean هيچ گونه رابط كامپوننت براي home , local , remote بدارد و فقط بايد جند متد خاص را فراخواني كرد .
ejbCreate() هنگامي يك نمونه از كلاس bean را ايجاد ميكنيم و ejbRemove() زماني كه bean ميخواهد از سرور خاج شود فراخواني ميكنيم

متد onMessage() از طرف ظرف bean زماني كه يك پيغام براي آن ميرسد فراخواني ميشود و متد setMessagedrivenBeanContext() برقراي كردن ارتباط به EJB container كه اجازه گرفتن اطلاعات درباره خود bean را درظرف ميدهد .
message driven bean متدهاي ejbActivate() / ejbPassivate() يا ejbLoad() / ejbStore() پشتيباني نمي كند چون نيازي به آنها ندارد .

همانند session bean کلاس message driven bean درخود Primary Key ندارد.

شرح دهنده هاي نصب و ارتقاء (Deployment Descriptors)

در اجراي bean ها يك سري اطلاعات در مورد مديريت آن bean در زمان اجرا نياز داريم كه در كلاسها و رابط هاي گفته شده معين نكرده ايم بطور مثال نحوه تعامل با سرويسهاي امنيتي ، تراكنش ، نام و سايرسرويس ها . براي اين كار از deployment descriptors استفاده مي شود .
مزيت اصلي deployment descriptors در اين است كه بدون تغيير در كد ميتوان حالت نرم افزار را درزمان اجرا تغيير داد.و كاربرد آن شبيه فايلهاي property در Visual Basic است كه نحوه Customize كردن نرم افزار بدون تغيير آن است.
اكثر IDE هاي Java از deployment descriptors پشتيباني مكنند وبه طور خود كار آن راتوليد ميكنند.

بسته بندی Packaging:

میتوان کامپوننتهای J2EE را در داخل بسته های نرم افزاری جدا گذاشت و در جاهای دیگر از آنها استفاده کرد(با deploy کردن) بسته ها شامل فایلهای وابسته نیز مثل کلاسهای مفید وgif, html میباشند.
بسته ممکن است از یک یا چند Bean وب و Enterprise بوده به همراه شرح دهنده (descriptor) که یک سند XML که تنظیمات نصب (deploy) کردن کامپونت درداخل ان نوشته شده می باشد.

بسته ها با پسوند .EAR (Enterprise Archive) بسته های برای کامپوننتهایEnterprise هستند که فایلهای EAR یک استاندارد ازبایگانی فایل جاوا JAR (Java Archive) می باشند
بسته ها با پسوند .WAR (Web Archive) یک شرح دهنده نصب برای فایل کامپوننتها ی وب ومنابع وابسته می باشد . از نوع JAR در بسته های سمت Clientاستفاده میشود .

در مدل EAR میتوانید چند عدد از کامپوننتهای داخل EAR را بطور جدا با تنظیمات متفاوت نصب کنید.

در زیر deployment descriptors مربوط به کلاس CabinEJB رامشاهده میکنید.

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
EnterpriseJavaBeans 2.0//EN" "http://java.sun.com/j2ee/dtds/ejb-
jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>CabinEJB</ejb-name>
      <home>CabinHomeRemote</home>
      <remote>CabinRemote</remote>
      <local-home>CabinHomeLocal</local-home>
      <local>CabinLocal</local>
       <ejb-class>java.lang.Integer</prim-key-class>
```

```
        <persistence-type>Container</persistence-type>
        <reentrant>False</reentrant>
      </entity>
    </enterprise-beans>
</ejb-jar>
```

deployment descriptors از XML يك فايل متني كه ساختار آن بر اساس DTD(Document type Definition) استاندارد EJB DTD شركت سان ميباشد.

EJB Home

زماني كه يك enterprise bean در داخل ظرف نصب ميشود متد هاي داخل Home كمك ميكنند به Container به مديريت EJB bean در چرخه زندگي آن و رابطه نزديكي با ظرف دارد به طوري كه مسئول تخصيص ، ايجاد و حذف EJB bean است .

برا ي مثال وقتي متد create() روي رابط Home فراخواني ميشود يك EJB home شي EJB object ايجاد كرده و اشاره گر را برميگرداند ويا وقتي ejbCreate() در entity bean اجرا ميشود يك reference به client ميدهد.
اكنون client ميتواند با كمك stub آن شي را فراخواني كند.

استفاده از Enterprise Bean

حال به نحوه استفاده سرويس گيرنده از يك enterprise bean مي پردازيم .
يك كابين شي يا موجوديتي است كه شرح آن در داخل پايگاه داده ذخيره ميشود و براي درك بهتر ساير entity bean ها مثل كشتي ، بليت ، مشتري و كارمند را در نظر بگيريد .

گرفتن اطلاعات از يك entity bean

تصور كنيد كه سرويس گيرنده به كمك رابط گرافيكي خود بايد اطلاعات يك سفر دريايي را به كابر نمايش دهد و اين اطلاعات شامل نام گشت يا سفر ، نام كشتي و ليست كابينهاي آن ميباشد .
با استفاده از شماره گشت كه فيلد متني است ميتوانيم از آن در entity bean به عنوان Primary Key استفاده كرده وداده ها را بازيابي كنيم. در زير كدها يمشابه را ملاحظه ميكنيد.

```
CruiseHomeRemote cruiseHome = ... use JNDI to get the home
// Get the cruise id from a text field.

String cruiseID = textFields1.getText();
// Create an EJB primary key from the cruise id.
```

```
Integer pk = new java.lang.Integer.parseInt(cruiseID);
// Use the primary key to find the cruise.

CruiseRemote cruise = cruiseHome.findByPrimaryKey(pk);
// Set text field 2 to show the cruise name.

textField2.setText(cruise.getName());
// Get a remote reference to the ship that will be used
// for the cruise from the cruise bean.

ShipRemote ship = cruise.getShip();
// Set text field 3 to show the ship's name
.
textField3.setText(ship.getName());
// Get a list of all the cabins on the ship as remote references
// to the cabin beans.

Collection cabins = ship.getCabins();

Iterator cabinItr = cabins.iterator();
// Iterate through the enumeration, adding the name of each cabin
// to a list box.

while( cabinItr.hasNext())

  CabinRemote cabin = (CabinRemote)cabinItr.next();

  listBox1.addItem(cabin.getName());

}
```

در ابتدا با گرفتن يك reference يا اشاره گر به EJB home از entity bean با نام گشت درياي (cruise) كه از JNDI (Java Naming and Directory Interface) استفاده ميكند.
JNDIيك سري API پرقدرت براي پيدا كردن منابع واشياء روي شبكه مي باشد و كمي پيچيده ميباشد به همين دليل در قسمتهاي بعدي توضيح خواهيم دا د.
زماني كه شماره گشت cruiseID ازكاربر گرفته ميشود ازآن در ايجاد كليد اصلي entity bean استفاده ميشود . با CruiseRemote و pk آن گشت پيدا شده و نام گشت را به كاربر نمايش ميدهد .
در گشت ميتوان اشاره گر به كشتي آن را گرفت ودر شي با نوع ShipRemote گذاشت سپس به كمك آن نام كشتي را بازيابي كرد. همچنين نام كابين هاي كشتي به كمك نام كشتي بازيابي و در حلقه while نشان داده.
اينها اين واقعيت را مي رسانند كه مدل داده entity bean يك سيستم با رابط پايدار به داده هاي پايگاه داده ارائه ميدهد مثلآ relationship يا ارتباط منطقي بين گشت درياي و كشتي يا كشتي وكابين هاي آن كه به راحتي با ship.getCabins() اين عمل انجام ميشود.

مدل سازي جريان كار با session bean

Entity bean مناسب براي مفاهيم وداده هاي تجاري بوده در يك نام ميتوانند خلاصه شوند اما آنها براي ارائه يك فرآيند با وظيفه مناسب نبوده . يك ship bean متدهاي براي كار باشي كشتي دارد اما نمي تواند ضمينه تعريف كند كدام اعمال انجام شده يا نشده

فصل هشتم

J2ME

تكنولوژي Java 2 Platform Micro Edition (J2ME) زير ساخت جاوا براي اجراي برنامه هاي مخصوص جاوا روي دستگاه هاي كوچك همانند تلفنهاي سلولي (موبايل)، كامپيوترهاي كوچك همراه Personal Digital Assistant (PDA) و ساير سيستمهاي جاسازي شده است. زير ساخت J2ME شامل يك پيكره بندي، يك نمايه (profile) و يك بسته اختياري است.
در بسته موبايل NetBeans IDE شما مي توانيد بسياري از جنبه هاي فرآيند توسعه Mobile Information Device Profile (MIDP) را به ساده گي انجام دهيد. اين بسته شامل يك طراح ويژوال MIDP براي محتوا و روند صفحه برنامه، همچنين پشتيباني از ويرايشگر مخصوص آن دستگاه و حتي فرآيند توسعه برنامه براي يك سرويس دهنده راه دور است.
بمنظور استفاده از قابلتهاي موبايل جاوا شما بايد بسته مخصوص را نصب كنيد. اگر از NetBeans IDE 4.1 استفاده مي كنيد شما بايد اين بسته ار دانلود كرده و از نصب كننده add-on آن استفاده كنيد. در ورژن 5 اين IDE اين بسته را مي توان هنگام نصب بطور خودكار در داخل IDE قرار داد. قابليتهاي J2ME هم اكنون قسمتي از J2SE API شده. در زير تعدادي از اصلاحات در باره برنامه نويسي موبايل را شرح داده ايم.

CLDC : اين كمله مخفف Connected Limited Device Configuration است و در J2ME بمنظور پيكره بندي انواع موبايلهاي مختلف استفاده مي شود. اين پيكره بندي شامل يك محيط زمان اجرا و API هاي هسته مخصوص پردازندها با سرعت و حافظه محدود دستگاه هاي موبايل است. همان طور كه مي دانيد دستگاههاي موبايل توسط شركتهاي مختلف توليد مي شوند و در داخل آنها هر شركت IC پردازنده خاص كه خود آنها را ساخته اند به كار مي برند و فقط در مدارات مياني و ارتباطي از IC هاي استاندارد بازار استفاده مي كنند. پس در اينجا نياز داريم كه براي انواع موبايلهاي مختلف يك استاندارد توسعه داشته باشيم. اين مضوع همانند كامپيوترها با زير ساختها و برنامه هاي مختلف است، و ماشين مجازي جاوا اين مشكل را حل كرده. پس هر دستگاه موبايل CLDC مخصوص خود را دارد.

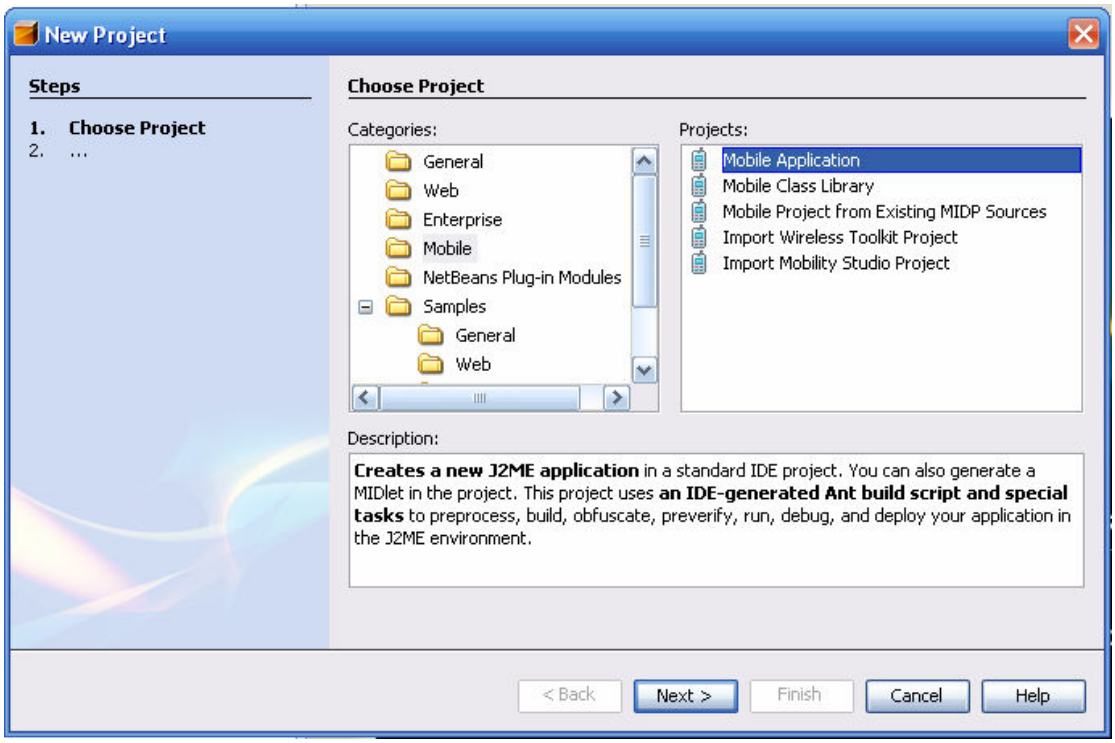
MIDP: نمايه اطلاعات دستگاه موبايل (Mobile Information Device Profile) يك مجموعه API به منظور ارائه قابليتهاي سطح بالا مورد نياز براي برنامه هاي موبايل است. اين قابليتها مي تواند شامل اجزاء قابل نمايش (مثل screen) و ارتباطات شبكه است.
MIDlet: يك كلاس مي باشد كه مورد نياز تمام برنامه هاي MIDP است. در عمل MIDlet يك رابط بين برنامه و دستگاه موبايلي كه برنامه در روي آن در حال اجرا است مي باشد. مي توان گفت كه MIDlet شبيه main class در پروژه هاي J2SE است.

Preverification: زماني كه يك برنامه با CLDC اجرا مي شود تمام كلاسهاي كامپايل شده بايد پالايش شوند. اين عمل يك فرآيند براي اضافه كردن تفسير اضافي به كمك ماشين مجازي CLDC به يك فايل بايت كد كلاس است. فرآيند Preverification اين اطمينان را به وجود مي آورد كه محتواي كلاس فقط كدهاي اجرا شوند در آن CLDC JVM باشند.

Device Fragmentation: اين عبارت براي گوناگوني زيرساختهاي مختلف موبايلها كه از داشتن يك مشخه برنامه نويسي اجرا شونده بر روي تمامي موبايلها جلو گيري مي كند است. اين گوناگوني ها مي تواند بصورت فيزيكي مثل اندازه و رنگ صفحه نمايش يا اندازه حافظه باشد، و يا بصورت نرم افزاري مثل APIهاي در دسترس و ورژنهاي CLDC/MIDP باشد.

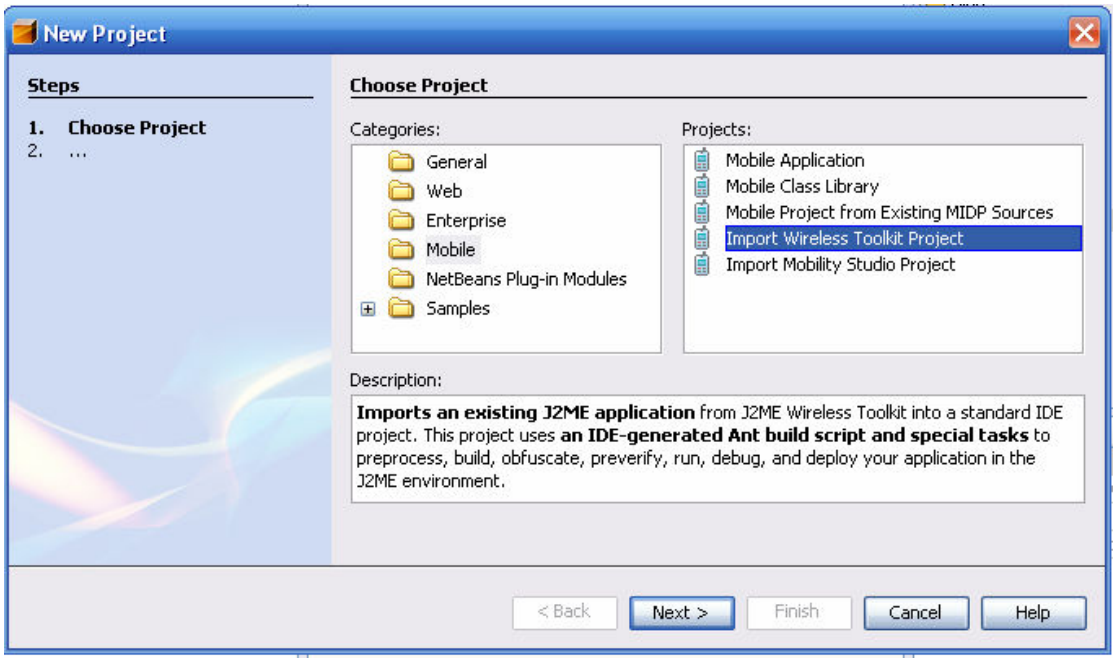
Preprocessor: پردازنده در بسته موبايل NetBeans به شكل يك وظيفه (task) است كه فايلهاي كامپايل شده J2ME را اجرا مي كند.
Obfuscation: فرآيندي كه فايلهاي كلاس را براي مهندسي معكوس (جلوي گيري از استفاده غير قانوني از كدهاي برنامه شما) مشكل مي كند. اين به طور معمول با جايگزيني نامهاي بسته ها، كلاسها، متدها و فيلدها با شناسه هاي عددي است. اين كار يكي از جنبه هاي فرآيند توليد نرم افزار موبايل مي باشد و نتيجه اين كار كاهش اندازه برنامه است.
در زماني كه يك پروژه براي موبايل ايجاد مي كنيد IDE يك مجموعه از كارها را انجام مي دهد. اين كارها شامل انتخاب يك شبيه ساز پروژه، مسير فايل كلاسهاي برنامه و ايجاد دستورات پيش فرز براي اجرا مي باشد.
براي ايجاد يك پروژه موبايل گامهاي زير را انجام دهيد:

- از منوي فايل گزينه New Projectرا انتخاب كنيد.
- در درخت اقلام فهرست Mobile را انتخاب كنيد.
- در زير درخت يك از دو گزينه Mobile Class Library يا Mobile Application را انتخاب كنيد. گزينه Mobile Application يك الگوي (template) از MIDlet بطور خودكار را براي شما ايجاد مي كند. گزينه دوم هيچ كلاسي براي شما ايجاد نمي كند و شما خود بايد كلاسها را ايجاد كنيد. مثالهاي MIDP و Bluetooth (پرتكول ارتباط دستگاهاي جانبي كوچك) از Visual Designer استفاده مي كنند.



- دكمه Next را كليك كنيد.
- به طور اختياري قسمتهاي زير را مي توانيد براي نام و محل پروژه در ويزارد وارد كنيد. در قسمت Project Name نام پروژه در رابط كاربر IDE را وارد كنيد و در قسمت Project Location محل پروژه را در سيستم خود وارد كنيد.
- به طور اختياري مي توانيد جعبه checkbox ايجاد Hello MIDlet را غير فعال كنيد. اين عمل از ايجاد مثال جلوگيري مي كند.
- در اين قسمت مي توانيد شبيه ساز و پيكره بندي مخصوص موبايلي كه برنامه را براي آن توسعه مي دهيد،را انتخاب كنيد. اين كار سبب مي شود در هنگام اجراي برنامه، برنامه شما بر روي شبيه ساز و دستگاه موبايل مورد نظر اجرا شود. توجه داشته باشيد كه اين تنظيمات را بعد از ايجاد پروژه نيز مي توانيد تغيير دهيد.
- در پايان گزينه Finish را كليك كنيد.

اين قابليت وجود دارد كه به پروژه خود پروژه هاي J2ME Wireless Toolkit، پروژهاي موبايل NetBeans و ساير منابع MIDP را وارد (import) كنيد. براي كار بعد انتخاب New Project ازمنوي فايل در زير درخت اقلام فهرست Mobile گزينه Import Mobility Studio Project يا Import Wireless Toolkit Project راانتخاب كنيد. سپس محل فايلهاي اين پروژه ها را مشخص كنيد. اين قسمت در شكل زير نشان داده شده.



بعد مشخص كردن پروژه بر روي Nextكليك كنيد. بقيه مراحل همانند ايجاد پروژه جديد گفته شده درقسمت قبل است.

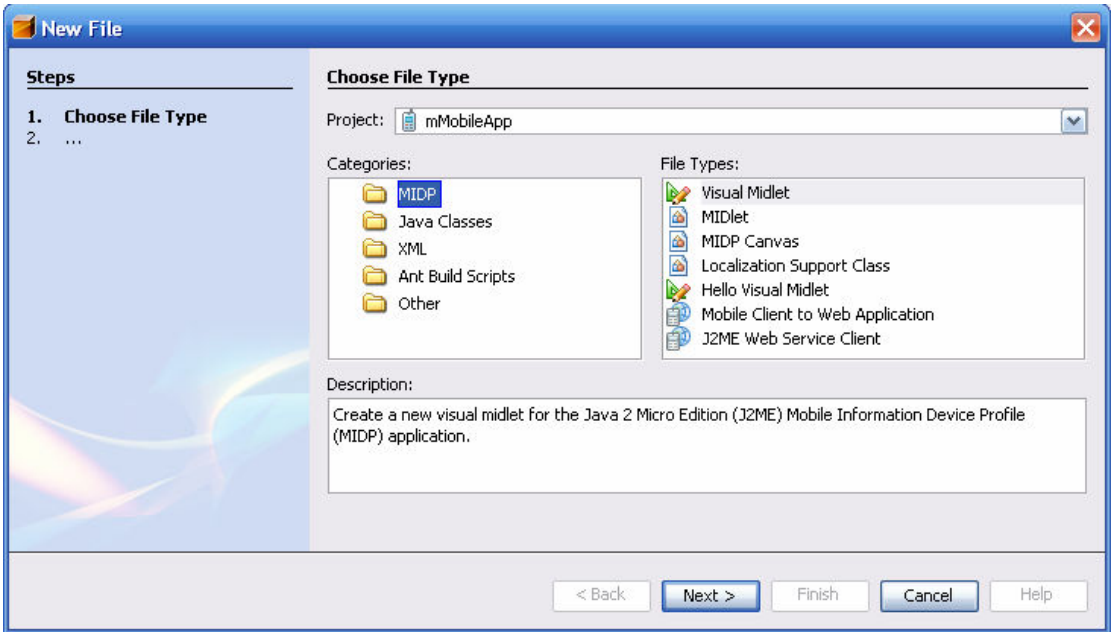
پيكره بندي پروژه
شما قادر به امتحان ساختار فيزيكي پروژه خود با استفاده از پنجره Files مي باشيد. زماني كه يك پروژه موبايل را ايجاد مي كنيد، بطور خودكار IDE فايلها و فهرستهاي لازم را كه براي همه پرژه ها ايجاد مي كند را به جز فهرست test مي سازد. اگر تستها ايجاد شده باشند بسته و كلاس آنها در فهرست test قرار خواهند داشت. فهرست build و dist زماني كه پروژه را ايجاد مي كنيد ساخته خواهند شد. در داخل فهرست build فهرستهاي زير موجود است:

- compiled: اين فهرست شامل تمام كلاسهاي كامپايل شده است.
- preprocessed: اين فهرست ورژنهاي از قبل پردازش شده فايلهاي منبع را در خود دارد. اگر شما از پيكره بندي استفاده مي كنيد اين فايلها از منبع هاي اصلي متمايز هستند.
- obfuscated: اين فهرست ورژنهاي obfuscated از فايلهاي كلاس شما نگهداري مي كند.
- preverified: نگهداري كننده ورژنهاي preverified از كلاس فايلهاي شما است.

بسته بندي فايلهاي كلاس در فايلهاي JAR توزيع مي شوند. اين فايلها سرعت فرايند توسعه را افزايش داده و با اطلاعاتي كه به شما مي دهند اشكالات فرآيند توسعه كه ممكن است در داخل برنامه شما ايجاد شوند را در انزوا قرار مي دهند.
فرآيند توسعه در NetBeans از استاندارد Ant براي مديرت كلاسهاي ايجاد شده موقت استفاده مي كند. معناي اين كار اين مي باشد كه فايلهاي منبع با بيش از چند برچسب زماني (تغييرات در زمانهاي مختلف) از كلاسها فايلها در فهرست build زماني كه دستور Build اجرا مي شود بطور خودكار دوباره ايجاد خواهند شد. همچنين در زمان حذف كردن يك فايل منبع از پروژه بطور خودكار فايلهاي كلاس وابسته نيز از فهرست build حذف خواهند شد.
در فهرست dist از پرژه فايلهاي JAR, JAD نگهداري مي شوند.

استفاده از الگوهاي فايل موبايل

فرآيند ايجاد فايلها در پروژه هاي موبايل شبيه بهايجاد فايلها در پروژه هاي عمومي جاوا است. الگوي ايجاد كلاسهاي MIDP داخل فهرست MIDP از الگوها در ويزارد New File مي باشد، اين فهرست در شكل زير نمايش داده شده. براي باز كردن ويزارد New File از منوي File گزينه New File را انتخاب كنيد.



MIDP Canvas

بمنظور ايجاد MIDP Canvas مراحل زير را انجام دهيد.

- در ليست File Types گزينه MIDP Canvas را انتخاب كنيد و Next را كليك كنيد.
- بطور اختياري مي توانيد در قسمت MIDP Class Name نام كلاس جديد را وارد كنيد. همچنين در قسمت Package بسته اي كه كلاس جديد در آن قرار دارد را وراد كنيد.
- دكمه Finish را كليك كنيد.

MIDlet

ويزارد ايجاد فايل MIDlet كمي متفاوت مي باشد و اين تفاوت در گرفتن اطلاعاتي كه در فايل JAD برنامه استفاده مي شود است.

- در ليست File Types گزينه MIDlet را انتخاب كنيد و Next را كليك كنيد.
- بطور اختياري در قسمت مي توانيد فيلدهاي نام و محل در ويزارد را تغيير دهيد. فيلد MIDlet Name مقداري مي باشد كه در زمان شروع نمايش برنامه نشان داده خواهد شد. اين نام جدا از نام كلاس خواهد بود. فيلد MIDP Class Name براي نام كلاس جديد MIDlet استفاده مي شود.براي وارد كردن محل icon كلاس MIDlet از فيلد MIDlet Icon استفاده كنيد. فيلد Package بسته اي كه كلاس جديد MIDlet در آن قرار دارد را مشخص مي كند.
- در نهايت روي دكمه Finish كليك كنيد.

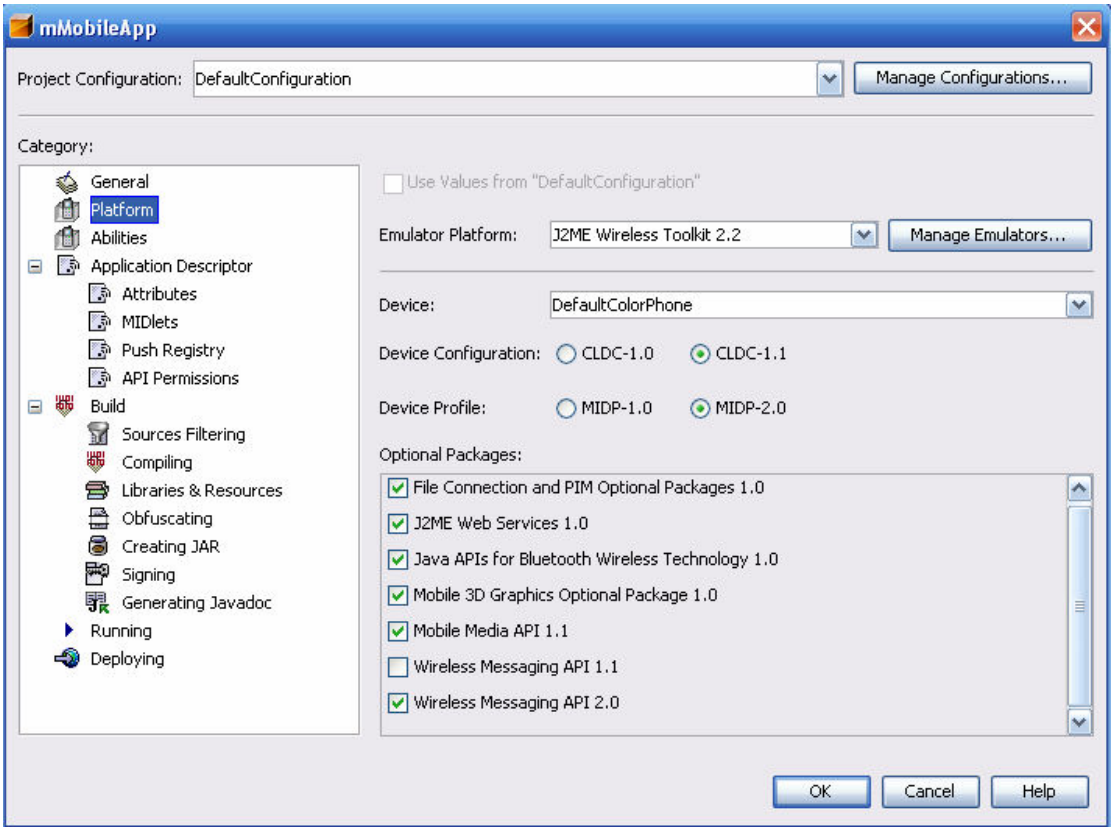
پيكره بندي پروژه

شبيه به اغلب پروژه ها تنظيمات كلاسهاي كه در كامپايل پروژه استفاده مي شود در Project Properties مي باشد. دو پانل Libraries&Resources و Platform به طور ويژه براي كنترل كلاسها هستند. به علاوه با اين تنظيمات كامپايل كد نهاي كنترل مي شود.

تغيير شبيه ساز پروژه

انتخاب زيرساخت شبيه ساز براي تعين كتابخانه هاي در هنگام كامپايل پروژه استفاده مي شوند مي باشد.به صورت پيش فرز پيكره بندي پروژه از آخرين ورژن J2ME Wireless Toolkit استفاده مي شود. شما مي توانيد در پانل Platform اين تنظيمات را تغيير دهيد. گامهاي زير را براي اين كار دنبال كنيد.

- در پنجره Project بر روي پروژه راست كليك كرده و Properties را انتخاب كنيد.
- بر روي گره Platform كليك كنيد. اين پنجره در شكل زير به نمايش در آمده.



- به طور اختياري هر كدام از فيلدهاي زير را كه كتابخانه شبيه ساز از آن استفاده مي كند مي توانيد تغيير دهيد.

Emulator Platform: زير ساخت نصب شده براي شبيه ساز را تعين مي كند. مقدار هاي اين كنترل در فيلدهاي زير قابل تغيير مي باشد.

Device: توسط اين گزينه مي توانيد شكل دستگاه موبايل براي اجراي برنامه ار انتخاب كنيد.

Device Configuration: پيكره بندي دستگاه كه بروي كلاسهاي برنامه تاثير گذار است را مشخص مي كند.

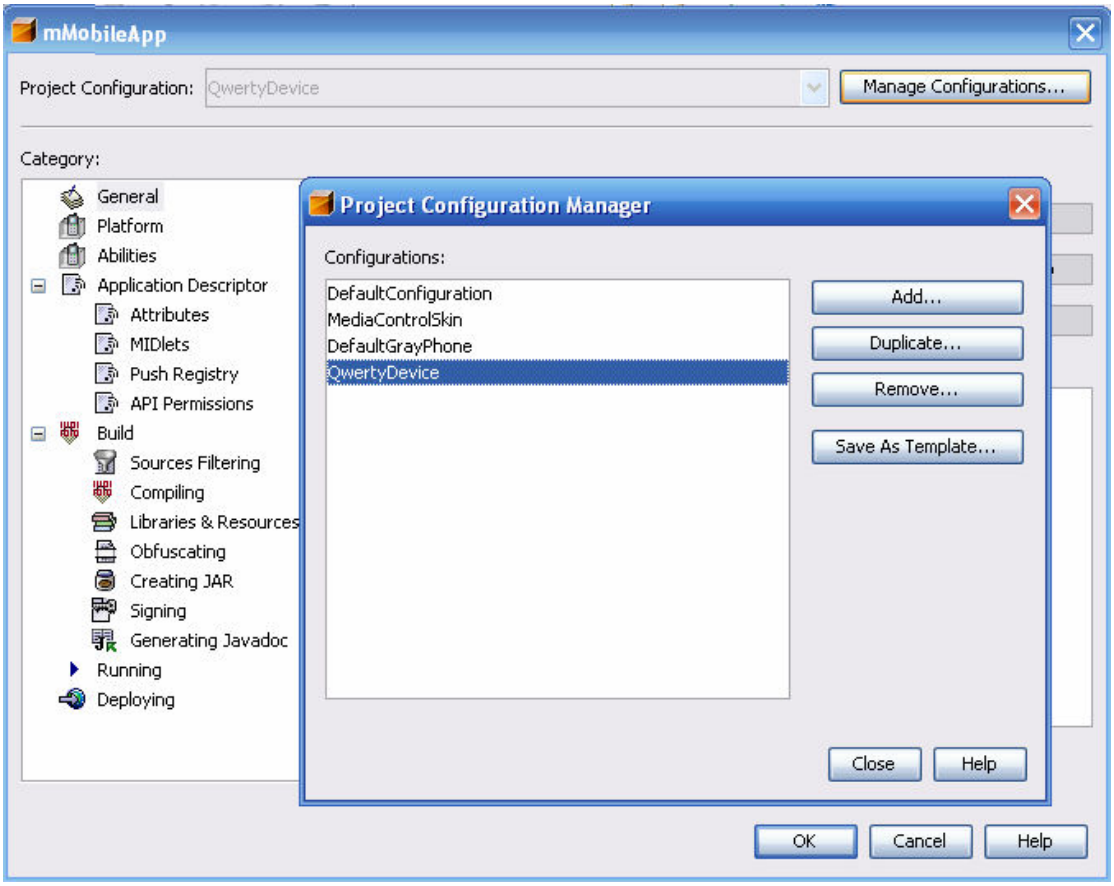
Device Profile: نمايه دستگاه موبايل (Mobile Information Device Profile) كه يك مجموعه API به منظور ارائه قابليتهاي سطح بالا مورد نياز براي برنامه هاي موبايل است را مشخص مي كند. اين قابليتها مي تواند شامل اجزاء قابل نمايش (مثل screen) و ارتباطات شبكه است.
Optional APIs: در قسمت مي توانيد بسته هاي اختياري كه مي توانيد به برنامه خود اضافه كنيد را انتخاب كنيد.

پيكره بندي پروژه براي دستگاه هاي مختلف

يكي از جنبه هاي توسعه برنامه هاي موبايل قابليت اجراي آنها بر روي انواع دستگاه ها است و اين به يك چالش تبديل شده است. دستگاه هاي مختلف داراي تفاوتهاي فيزيكي مثل اندازه صفحه و اندازه حافظه و غيره دارند. پس نياز به پيكره بندي هاي مختلف دارند. براي پيكره بندي از برگه General در Properties پروژه استفاده مي شود. توجه داشته باشيد كه تفاوت اصلي برگه Project Properties در پروژه هاي موبايل و پروژه هاي عمومي جاوا در جعبه تركيبي با برچسب Project Configuration در قسمت بالا است. اين جعبه تركيبي شامل يك عنصر براي هر پيكره بندي در پروژه شما است.
در حالت كلي شما بايد يك پيكره بندي براي هر فايل JAR توزيع شده ايجاد كنيد. براي مثال اگر شما قصد داريد يك برنامه با سه نوع اندازه صفحه و استفاده از دو مجموعه از API ايجاد كنيد بايد شما شش پيكره بندي را ايجاد كنيد.

ايجاد پيكره بندي

- بعد از باز كردن برگه Properties پروژه بر روي General كنيد.
- سپس بر روي كمه Manage Configuration كليك كنيد.بعد از اين كار جعبه محاوره اي پيكره بندي باز خواهد شد. اين برگه را در شكل زير مشاهده مي كنيد.



- براي افزودن پيكره بندي بر روي دكمه Add كليك كنيد.
- در قسمت Name نام پيكره بندي جديد را وارد كنيد و بر روي OK كليك كنيد.

براي مثال نام NokiaSeries40Bluetooth يا HighResNosizeLimit هر توزيع از پيكره بندي را مشخص مي كند. توجه داشته اين نام دلخواه است.

ايجاد پيكره بندي دلخواه

دو قاعده كلي در پيكره بندي داخل برگه Project Properties وجود دارد. قائده اول استفاده از پيكره بندي هاي پيش فرز و دوم وارد كردن پيكره بندي كه از تنظيمات پيش فرز گرفته شده توسط كاربر. براي استفاده از تنظيمات كاربر بايد check box با برچسب "Use Value from "Default configuration در برگه Platform بايد غير فعال (unchecked) شده و برنامه ازپيكره بندي داخل combo box بالا استفاده مي كند. در اين مورد گامهاي زير يك مثال مي باشد.

- در برگه Properties پروژه پانل Platform را انتخاب كنيد
- مطمئن شويد كه در جعبه پائين افتادني (drop down) مربوط به Project Configuration گزينه Default Configuration فعال شده باشد. در جعبه پائين افتادني Emulator Platform گزينه J2ME Wireless Toolkit و در جعبه پائين افتادني Device گزينه DefaultGreyPhone را انتخاب كنيد.
- در جعبه پائين افتادني Project Configuration نام پيكره بندي ايجاد شده در قسمت را انتخاب كنيد.
- اگر پانل Platform فعال باشد خواهيد ديد كه تمام گزينه ها غير فعال خواهند شد. گزيرنه "Use Value from "Default configuration را غير فعال كنيد. با اين

کار تمام گزینه ها در دسترس خواهند بود،حال شما يك پيكره بندي مخصوص خود را مي توانيد تنظيم كنيد.

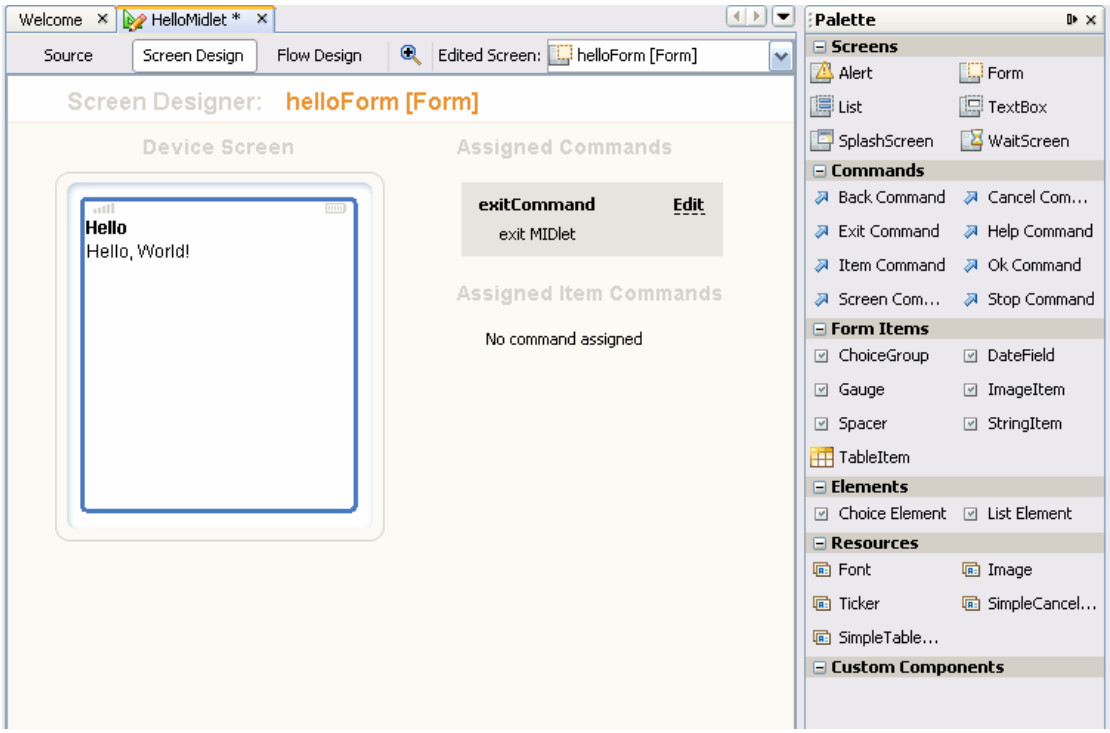
استفاده از MIDP Visual Designer

به كمك طرح ويژوال MIDP مي توانيد به طور بصري MIDlet ايجاد كنيد. اين يك محيط طراحي با تمام قابليتها مي باشد كه به شما اين امكان را مي دهد كه به طور سريع برنامه ها را ايجاد كرده و محتواي صفحات را تغيير دهيد. تغيير محتواي صفحات به كمك يك GUI بصري از استاندارد MIDP 1.0 و MIDP 2.0 استفاده ميكند.

ايجاد Visual Design جديد
براي اين كار گامهاي زير را دنبال كنيد.

- در پنجره پروژه بر روي بسته اي كه قصد داريد شامل Visual Design باشد كليك راست كرده و گزينه New | Visual Midlet را انتخاب كنيد. اگر اين قابليت فعال نيست مي توانيد از New | New File (Ctrl-N) از منوي اصلي استفاده كنيد.
- در جعبه ليست اقلام گره MIDP را انتخاب كنيد و سپس Visual MIDlet يا HelloMIDlet.java در ليست فايلها انتخاب كنيد. دكمه Next را كليك كنيد.
- حال ورژن MIDlet بر اساس دستگاهي كه براي آن برنامه را ايجاد مشخص كنيد. سپس Next را كليك كنيد.
- فيلدهاي مربوط به الگو (template) از قبيل نام و پيكره بندي را وارد كرده و آنها را ذخيره كنيد سپس دكمه Finish را فشار دهيد.

بعد از مراحل بالا Visual Design در حالت Flow Design باز خواهدشد. اگر HelloMidlet انتخاب شده باشد، اين MIDlet هم اكنون با يك جعبه متن در صفحه پيغام Hello World! را به شكل زير نشان خواهد داد.



کد این برنامه یه شکل زیر است.

```
package hello;

import javax.microedition.midlet;*.
import javax.microedition.lcdui;*.

public class HelloMidlet extends MIDlet implements CommandListener {

      public HelloMidlet} ()
{

   private Form helloForm;
   private StringItem helloStringItem;
   private Command exitCommand;


   private void initialize{()

       getDisplay().setCurrent(get_helloForm;(()
{

 **/    Called by the system to indicate that a command has been invoked on
a particular displayable.
@ *     param command the Command that ws invoked
@ *     param displayable the Displayable on which the command was
invoked
/*
   public void commandAction(Command command, Displayable displayable (
}
 //        Insert global pre-action code here
       if (displayable == helloForm} (
           if (command == exitCommand} (
 //                Insert pre-action code here
               exitMIDlet;()
 //                Insert post-action code here
{
{
 //        Insert global post-action code here
{

**/
 *     This method should return an instance of the display.
/*
   public Display getDisplay                    } ()
       return Display.getDisplay(this;(
                   {
```

```
**/
 *     This method should exit the midlet.
/*
    public void exitMIDlet                        } ()
        getDisplay().setCurrent(null;(
        destroyApp(true;(
        notifyDestroyed;()
                        {

 **/    This method returns instance for helloForm component and should be
called instead of accessing helloForm field directly.
@ *     return Instance for helloForm component
/*
    public Form get_helloForm} ()
        if (helloForm == null} (
 //             Insert pre-init code here
            helloForm = new Form(null, new Item[] {get_helloStringItem;({()
            helloForm.addCommand(get_exitCommand;(()
            helloForm.setCommandListener(this;(
 //             Insert post-init code here
{
        return helloForm;
{

 **/    This method returns instance for helloStringItem component and
should be called instead of accessing helloStringItem field directly.
@ *     return Instance for helloStringItem component
/*
    public StringItem get_helloStringItem} ()
        if (helloStringItem == null} (
 //             Insert pre-init code here
            helloStringItem = new StringItem("Hello", "Hello, World;("!
 //             Insert post-init code here
{
        return helloStringItem;
{

 **/    This method returns instance for exitCommand component and should
be called instead of accessing exitCommand field directly.
@ *     return Instance for exitCommand component
/*
    public Command get_exitCommand} ()
        if (exitCommand == null} (
 //             Insert pre-init code here
            exitCommand = new Command("Exit", Command.EXIT, 1;(
 //             Insert post-init code here
{
```

```
        return exitCommand;
{

    public void startApp} ()
        initialize;()
{

    public void pauseApp} ()
{

    public void destroyApp(boolean unconditional} (
{
```

{

اضافه كردن اشياء به جعبه كامپوننتها

صفحات تعين شده كاربر و اقلام آنها مي توانند به جعبه كامپوننتها براي استفاده اضافه شوند. البته ويراستار صفحه اجازه نمي دهد كه تمام مشخصات كامپوننتهاي كه كاربر اضافه كرده را ويرايش كنيم، بلكه Flow Designer از آنها استفاده مي كند. گامهاي زير را براي اضافه كردن دنبال كنيد.

- مطمئن شويد كه كلاسي كه از آن قصد استفاده داريد در مسير كلاسهاي برنامه موجود است.
- در منوي Tools از زير منوي Palette Manager گزينه Mobility Editor را انتخاب اين كار جعبه مديريت كامپوننتها را به شكل زير باز خواهد كرد.

- بر روي دكمه Add كليك كنيد.
- پروژه اي را كه شامل شي شما براي اضافه كردن است را انتخاب كنيد.
- با استفاده از ليست كلاسهاي پيدا شده MIDP در پروژه خود تمام اشياء لازم براي اضافه شدن را انتخاب كنيد.
- به صورت اختياري مي توانيد از جعبه تركيبي Add to Category براي انتخاب كردن جعبه اقلام استفاده كنيد.
- سپس بروي Finish و Close كليك كنيد.

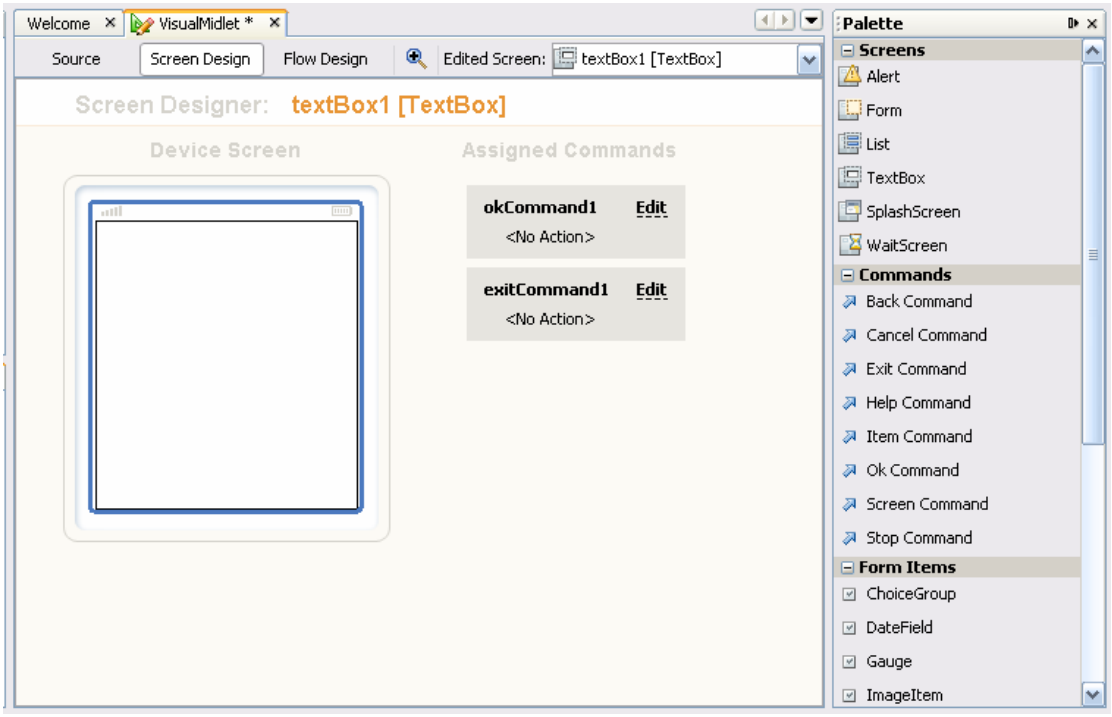
حال كامپوننتهاي شما قابل دسترس در Visual Designer خواهند بود.

ايجاد يك برنامه كوچك به كمك Visual Designer

بطور مثال گامهاي زير نحوه ايجاد استخوان بندي يك برنامه با دو صفحه كه مي تواند براي ارسال يك SMS استفاده شود را بيان مي كند.

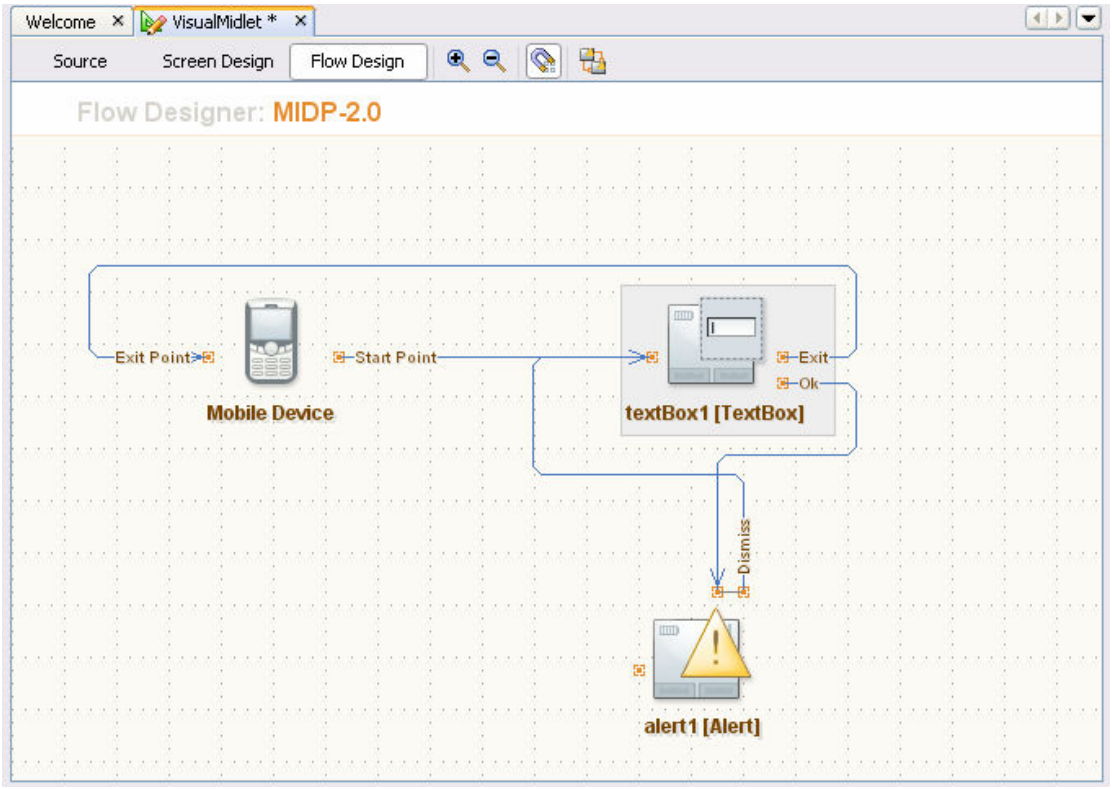
- با كليك راست بروي بسته پروژه و انتخاب Visual Midlet يك سند جديد Visual Design ايجاد كنيد و بر روي دكمه Flow Design در ميله ابزار designer كليك كنيد.
- بر روي TextBox از جعبه كامپوننتهاي Screen كليك كرده و آن را به Flow Design بكشيد (drag كنيد).
- Start Point برچسب دار شده در روي صفحه دستگاه موبايل را به جعبه متن تازه اضافه شده كليك و درگ كنيد. حال صفحه اول شما اضافه مي شود. اگر شما هم اكنون برنامه را اجرا كنيد برنامه با يك فيلد متني بدون دستور اختصاصي براي آن آغاز خواهد شد.

- از قسمت Commands در جعبه كامپوننتها OK command را كليك كرده و به صفحه TextBox درگ كنيد. حال به همان روش يك Exit command را نيز اضافه كنيد.
- يك Alert را از جعبه كامپوننتهاي Screen كليك كرده و آن را به Flow Designer درگ كنيد.
- حال صفحه Alert جديد را دابل كليك كنيد. اين كار آنرا در Screen Designer باز خواهد كرد.
- صفحه Device جاي كه <Enter Text> را خواهيد ديد كليك كنيد. حال اين جعبه متن قابل ويرايش خواهد بود. پيغام "SMS Sent" را وارد كنيد و Ctrl-Enter را براي ذخيره كردن تغييرات فشار دهيد.
- از combo box در بالاي Screen Designer براي رفتن به صفحه TextBox استفاده كنيد. اين صفحه در شكل زير به نمايش درآمده.



- بر روي جعبه متن كليك كرده و رشته <Edit Text> را حذف كنيد.
- به كمك ويرايشگر property رشته "Enter SMS text:" را در خصيصه Title وارد كنيد.
- بر روي دكمه Flow Design در مياه ابزار كليك كنيد تا به قسمت Flow Design برگرديد.
- از صفحه TextBox برچسب okCommand را كليك كردن و آنرا به صفحه Alert درگ كنيد.
- از صفحه TextBox برچسب exitCommand را كليك كردن و آنرا به صفحه Mobile Device درگ كنيد.

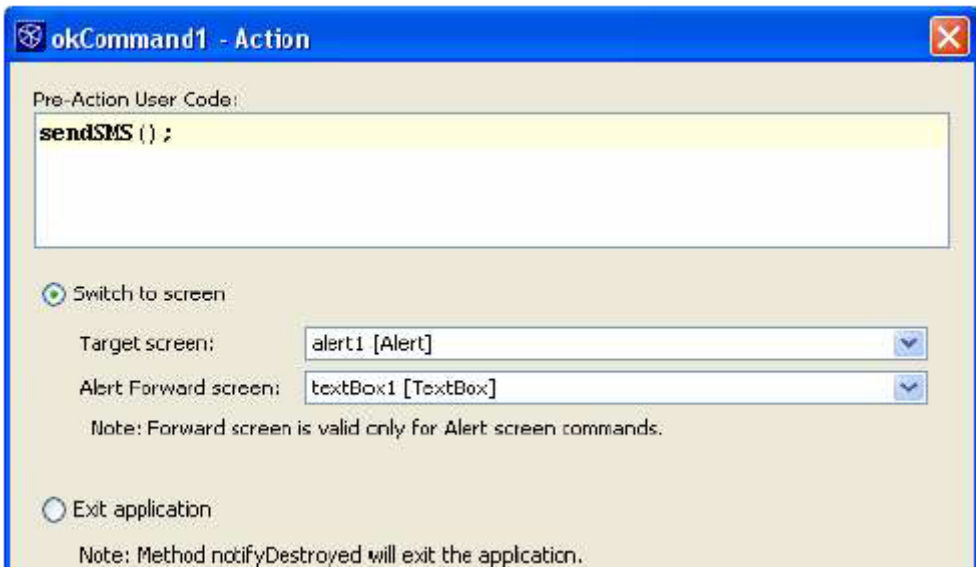
پس از انجام اين گامها Flow Designer شبيه به شكل زير خواهد شد.



اگر شما در این زمان برنامه را اجرا کنید، در شروع برنامه یك جعبه متن و دو گزینه OK و Exit را نشان خواهد داد. اگر کلید OK را فشار دهید برنامه شما را به صفحه Alert خواهد برد. و اگر کلید Exit را فشار دهید از برنامه خارج خواهید شد.

- در قسمت Properties مربوط به okCommand1 روي خصيصه Action كليك كرده و جعبه محاوره اي Action را باز كنيد.
- در جعبه محاوره اي Action در فيلد Callback Validation Method Name متن sendSMS را همان طور كه در شكل زير نشان داده شده وارد كنيد.

اگر فيلد Callback Validation Method Name در پنجره Action موجود نبود از جعبه راديوي گزينه Exit Application را انتخاب كرده و دكمه OK را كليك كنيد. سپس به پنجره Inespector رفته و از بر روي okCommand كليك راست كرده و گزينه Go To Source را انتخاب كنيد. در قسمت كد در زير شرط if(commad==okCommand1) به جاي متد exitMIDlet() متد sendSMS() را وارد كنيد و اين متد را ايجاد كنيد.
اين عمل متد sendSMS() را ايجاد خواهد كرد. دستورات اين متد جاي است كه شما كدهاي مربوط ارسال SMS را خواهيد نوشت. هر زمان كه دكمه OK فشرده شود اين متد فراخواني خواهد شد و اگر متد مقدار false را برگرداند اين به اين معني است كه انتقال انجام نشده.

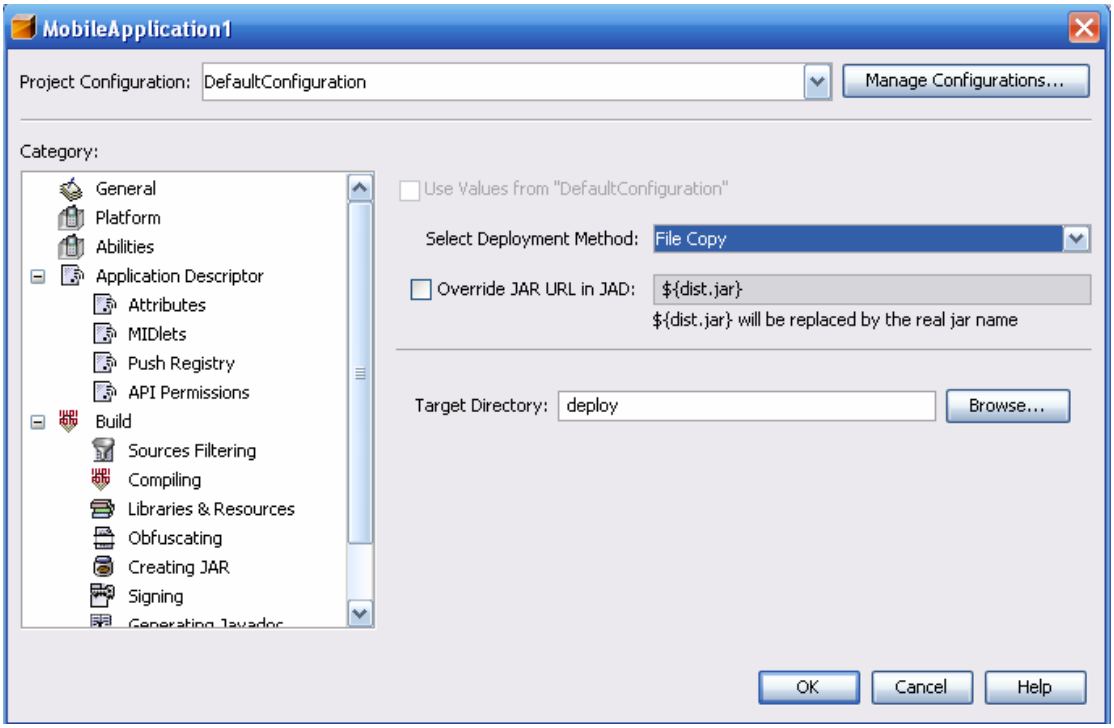


- در ميله ابزار Visual-Designer بر روي دكمه Source براي ديدن كدهاي منبع برنامه كليك كنيد. كدها به طور خودكار ايجاد شده اند و متد sendSMS() را خواهيد ديد. حال شما كدهاي لازم براي ارسال SMS را مي توانيد وارد كنيد. هم اكنون شما يك برنامه ساده MIDP را ايجاد كرده ايد.

Deploy كردن برنامه بصورت خودكار

چندين روش براي Deploy كردن برنامه هاي موبايل وجود دارد كه شامل گذاشتن برنامه در يك فايل JAR است. براي اين كار بهگامهاي زير را دنبال كنيد.

- بروي پروژه كليك راست كرده و گزينه Project Properties را انتخاب كنيد.
- پانل Deploying را انتخاب كنيد. اين پانل در شكل زير نشان داده شده.



- در combo box با برچسب Select Deployment Method روشي را كه مي خواهيد از آن براي Deploy كردن استفاده كنيد انتخاب كنيد.

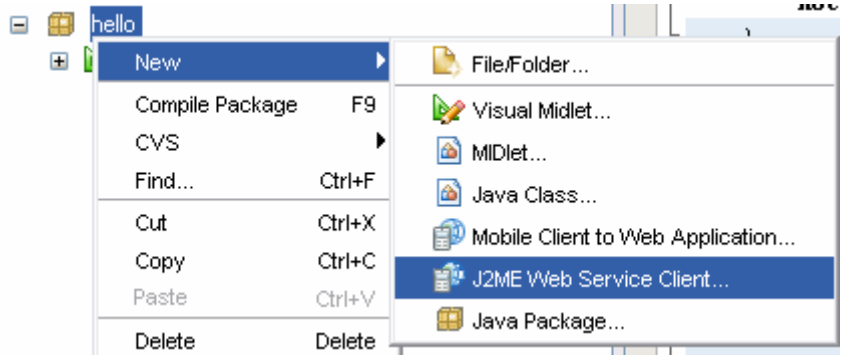
متد Copy در فهرستي كه در قسمت Target Directory مشخص مي شود اين فايل را خواهد ساخت. براي استفاده از متدFTP بايد نام سرويس دهنده راه دور و مقدار مجوزهاي لازم را وارد كنيد.

- به طور اختياري با تنظيم MIDlet-Jar-URL مي توانيد نام فايل JAR را تغيير دهيد.

ايجاد برنامه Client/Server

شما مي توانيد يك Client MIDlet ايجاد كنيد كه به يك سرويس وب متصل شود. اين اتصال به كمك يك برنامه ميان افزار (middleware) وب مي باشد. اين ويزارد يك ارتباط two-tier با استفاده از استاندارد JSR-172 كه مخصوص شناساي سرويس هاي وب مي باشد ايجاد خواهد كرد بديهي است كه دستگاه موبايل بايد از JSR-172 پشتيباني كند. اگر دستگاه موبايل شما از JSR-172 پشتيباني نمي كنيد برنامه را به صورت three-tierايجاد كنيد. در روش three-tier ويزارد به طور خود كار يك Servlet شامل برنامه client براي اتصال به سرويس وب ايجاد خواهد كرد و قبل از ايجاد با ويزارد بايد يك Web Project شامل Web Service Client را ايجاد كرده باشيد.
براي استفاده از ويزارد گامهاي زير را دنبال كنيد.

- بر روي بسته برنامه موبايل كليك كرده، همانند شكل زير و از زير منوهاي گزينه New گزينه J2ME Web Service Client را انتخاب كنيد.



- بعد از اجراي ويزارد در قسمت WSDL Service محل فايل شرح دهنده سرويس وب را مشخص مي كنيم. براي اين كار دو روش وجود دارد.

۱. اگر سرويس وب هم اكنون در اينترنت در حال اجرا باشد URL مخصوص فايل WDSL را وارد كرده و محل ذخيره شدن فايل دريافت شده را مشخص كنيد. اگر از ديوار آش مخصوص اينترت در شبكه شما استفاده مي شود بر روي دكمه Proxy Setting كليك كرده و نام ميزبان و پورت را وارد كنيد. براي دريافت فايل بر روي دكمه Retrieve WSDL كليك كنيد.
۲. اگر فايل WSDL هم اكنون در سيستم شما موجود است محل انرا براي IDE مشخص كنيد.

- زماني كه فايل WDSL گرفته شد باقيمانده فيلدها بطور خودكار پر مي شوند
- بر روي دكمه Finish كليك كنيد.

حال IDE به طور خود كار client stub به همراه تنظيمات را ايجاد مي كند.

استفاده از ويزارد Mobile Client to Web Application

- بر روي بسته برنامه موبايل كليك كرده، همانند شكل زير و از زير منوهاي گزينه New گزينه Mobile Client to Web Application را انتخاب كنيد.
- در صفحه انتخاب Web Application and Client type يك پروژه، Servlet يا يك بسته (Package) براي ايجاد كدهاي سرور انتخاب كنيد
- در صفحه انتخاب سرويس، سرويس صادر شونده از سرور به برنامه سرويس گيرنده را مشخص كنيد.
- در صفحه Client Option نام و بسته براي كلاس ايجاد شونده براي سرويس گيرنده را انتخاب كنيد
- در همان صفحه مي توانيد انواع كدهاي ايجاد شونده را تعين كنيد.

مراجع :

۱ مجله علم الكترونيك و كامپيوتر http://www.iranPcWordMaz.com

۲ برنامه نويسي شي گرا با جاوا در ۲۱ روز- محمد باقر معموري - انتشارت نص

۳ EJB – ريچارد مانسون هافل – انتشارات OReilly

۴ http://developers.sun.com/

۵ Java Servlt Programing – جيسون هانتر- انتشارات OReilly

۶ J2EE and XML Development - كورت گابريك – انتشارات MANNING

۷ http://java.sun.com/docs/books/tutorial

۸
NetBeans IDE Field Guide, Developing J2ME Mobility Applications