Eclipse ile Java EE EAR Projesi

(Püf Noktası Serisi)

KurumsalJava.com
KurumsalJavaAkademisi.com

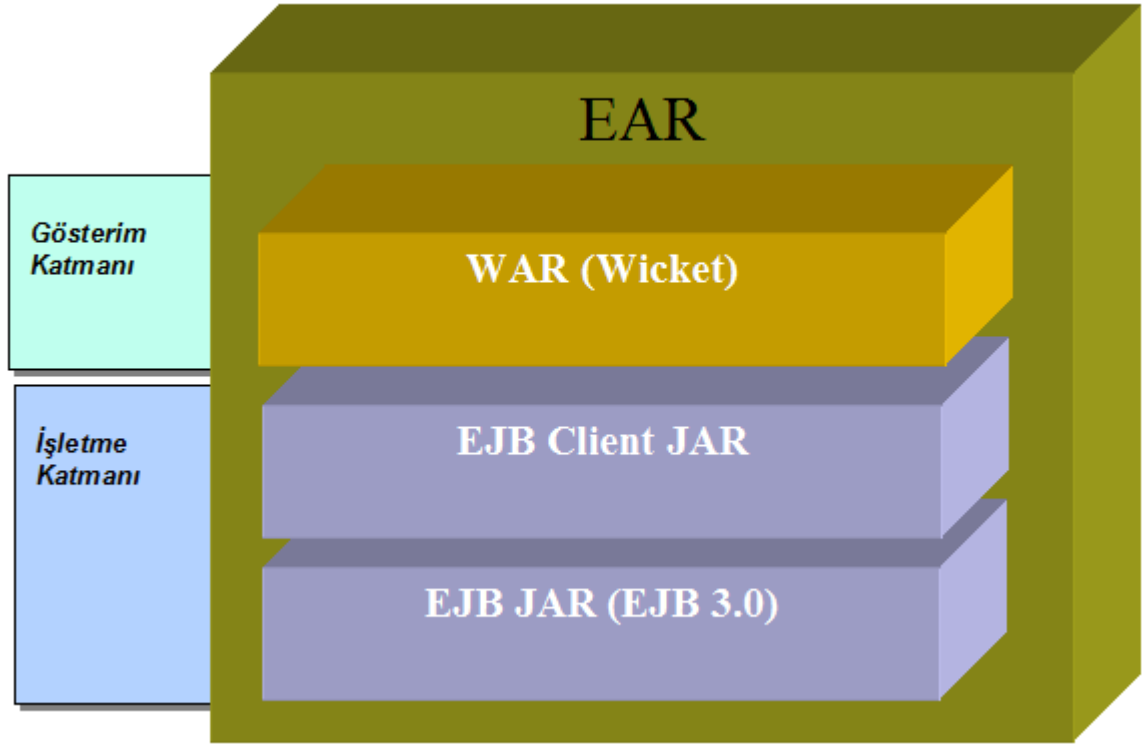
Özcan Acar
Bilgisayar Mühendisi
http://www.ozcanacar.com

Püf Noktası Serisi Hakkında

KurumsalJava.com bünyesinde, püf noktası ismini taşıyan bir bölüm yer almaktadır. Bu bölümde, değişik konular pratik çözümler sunularak incelenmektedir.

Giriş

Java EE 5 aplikasyonları EAR (Enterprise Archive) arşiv dosyaları içinde yer alır. Bir EAR dosyası JAR (Java Archive) dosyası yapısına ve .ear dosya ekine sahiptir. EAR dosyası içinde bir Java EE projesini oluşturan diğer modül arşivleri yer alır. Bir sonraki resimde bir EAR arşiv dosyasının yapısı yer almaktadır.



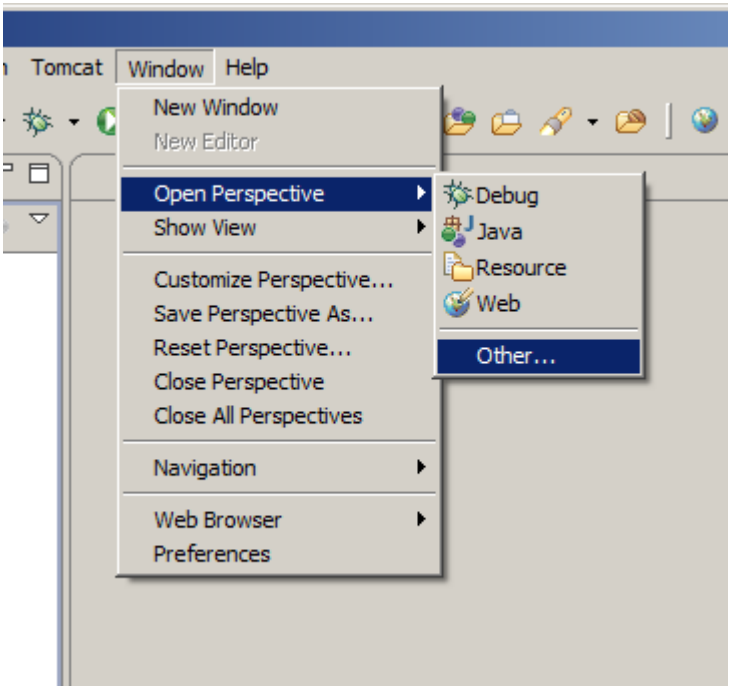
Bir EAR arşivinde yer alabilecek modül arşivleri şunlardır:

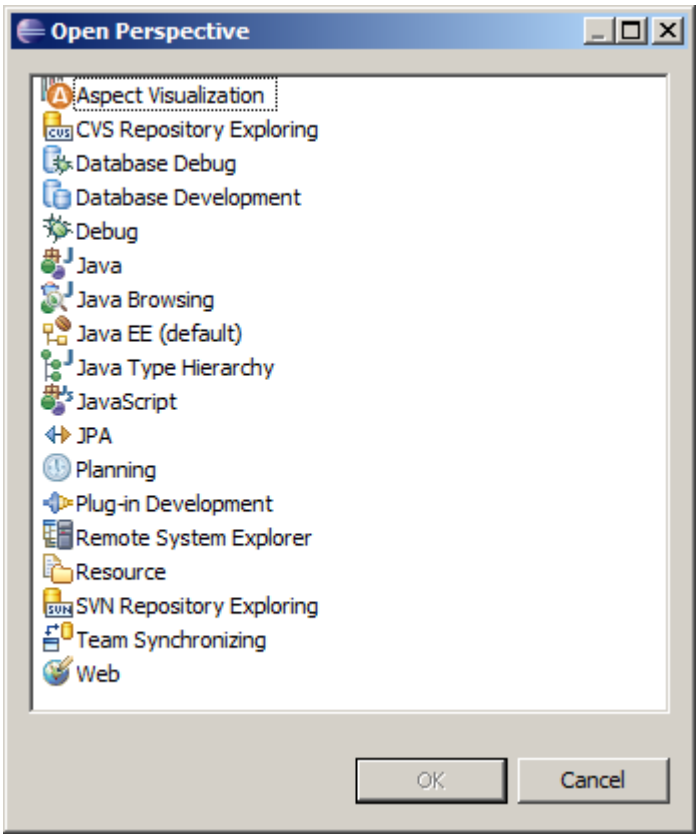
- **EJB JAR**: EJB komponentlerin yer aldığı JAR dosyasıdır. Bu dosya içinde sadece EJB implementasyonları yer alır.
- **EJB Client JAR**: EJB komponentlerinin sahip oldukları Remote ve Local EJB interface sınıflar EJB Client JAR arşivlerinde yer alır. Bu JAR dosyası EJB komponentlerini kullanmak isteyen diger modüllerin kullanımına sunulur.
- **WAR**: Web aplikasyonları WAR (Web Application Archive) arşiv dosyalarında yer alır.

Eclipse ile Java EE EAR Projesi

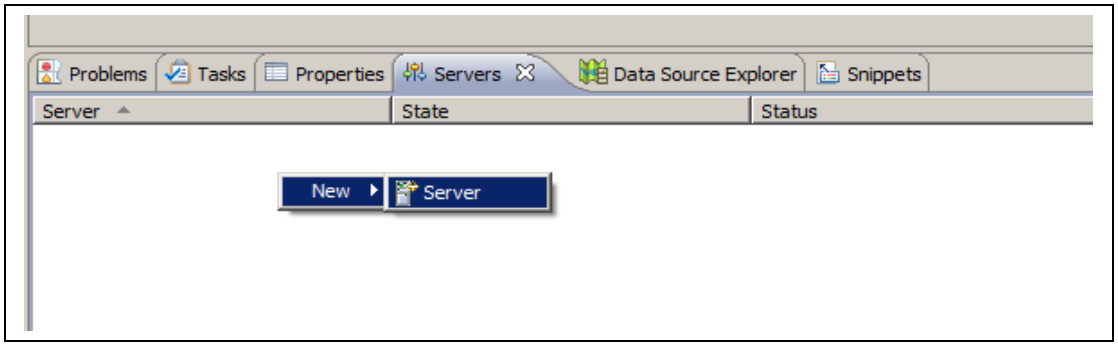
Tipik bir Java EE 5 aplikasyonu için gerekli EAR arşiv dosyasının nasıl oluşturulduğunu ve değişik tipteki aplikasyon modüllerinin bu arşive nasıl dahil edildiğini örnek bir proje üzerinde yakından inceleyelim. İlk işlem olarak yeni bir Eclipse Workspace oluşturuyoruz. Bu workspace bünyesinde EAR ve diğer arşivler Java EE projesi olarak yer alacaklar

Örnek proje için Eclipse 3.4 (Ganymede) versiyonunu kullandım. Yeni bir Java EE projesi oluşturmak için Eclipse Java EE perspektifine geçmemiz gerekiyor.

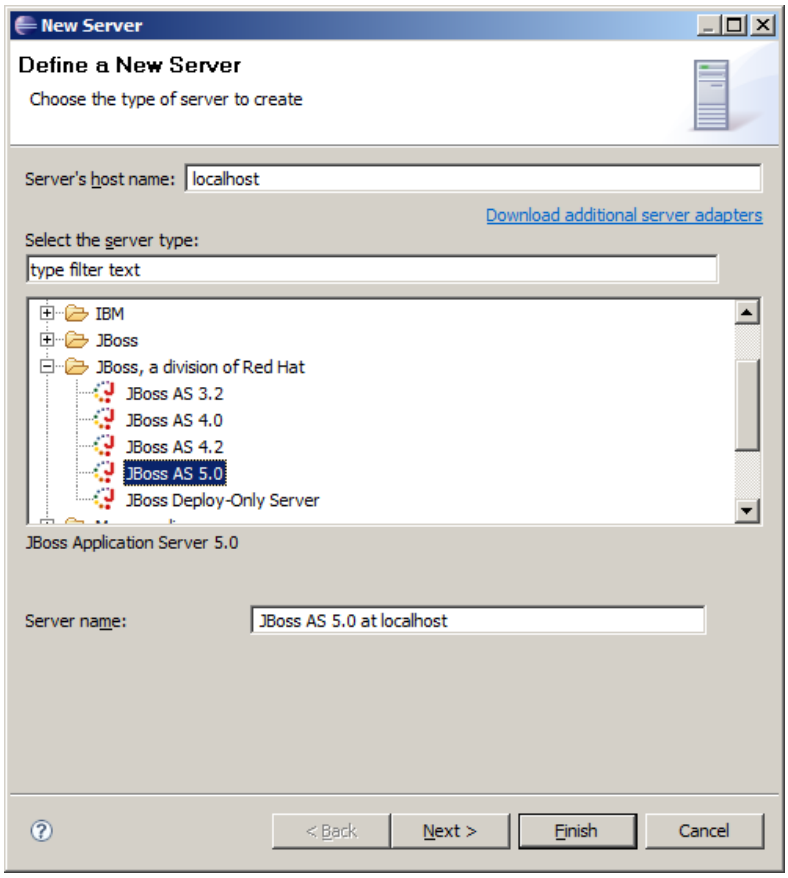


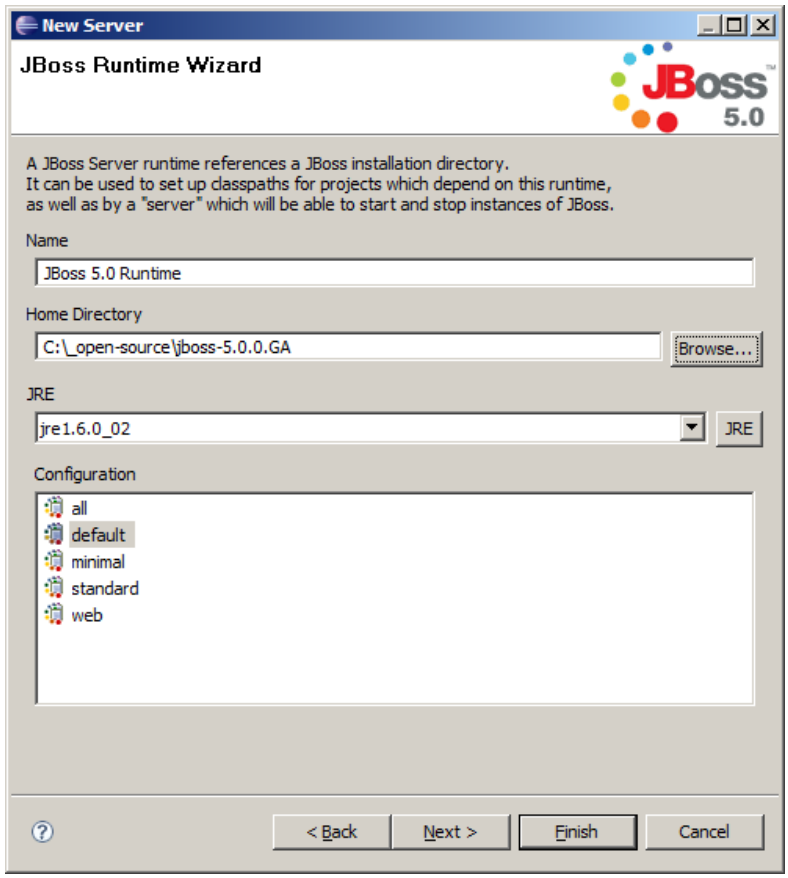


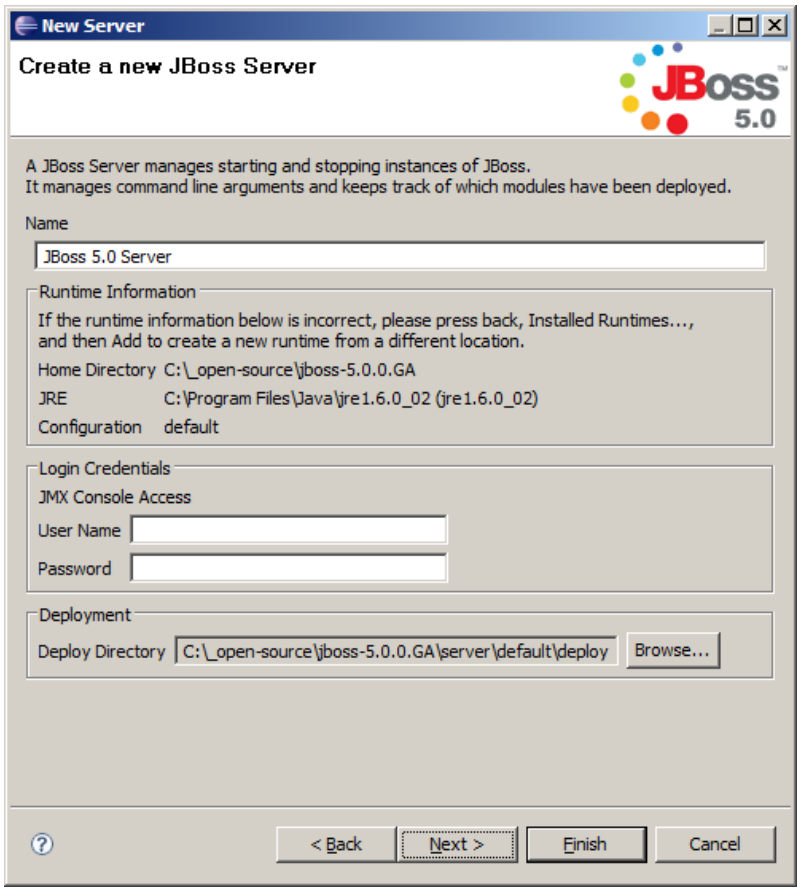
Perspektif panelinden Java EE (default) perspektifini seçerek, Java EE çalışma ortamını oluşturuyoruz. Oluşturacağımız projeyi Eclipse altında çalışır hale getirebilmek için JBoss gibi bir aplikasyon serverine ihtiyacımız var. Tasarladığımız aplikasyon EJB 3 teknolojisini ihtiva ettiği için JBoss 5.0 GA versiyonun http://www.jboss.org adresinden edindikten sonra, herhangi bir dizine yerleştiriyoruz. Akabinde JBoss Tools isimli Eclipse plugin serisini http://www.jboss.org/tools adresinden edinip, kullandığımız Eclipse versiyonun ana dizinine yerleştiriyoruz. JBoss Tools ile JBoss aplikasyon serverinin kullanımı ve kurulumu daha kolay bir hale gelmektedir.

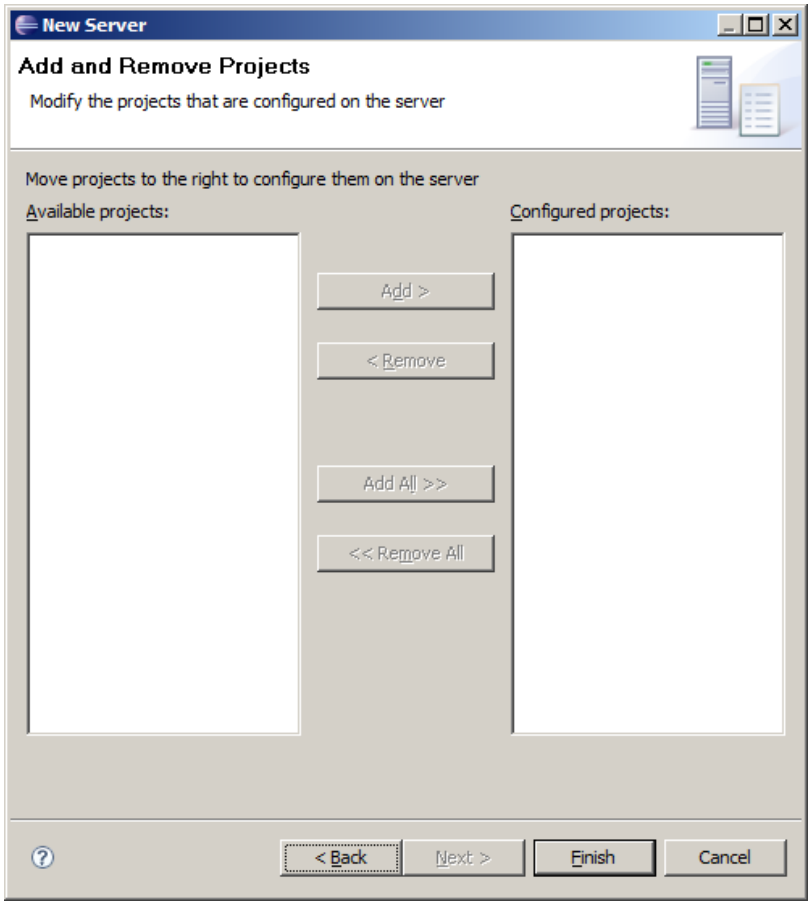


Çalışabileceğimiz bir server (application server) oluşturmak için Servers paneli üzerinde iken, sağ tuşa tıklıyoruz. ***New > Server*** menüsü üzerinden, yeni JBoss server kaydını oluşturuyoruz.

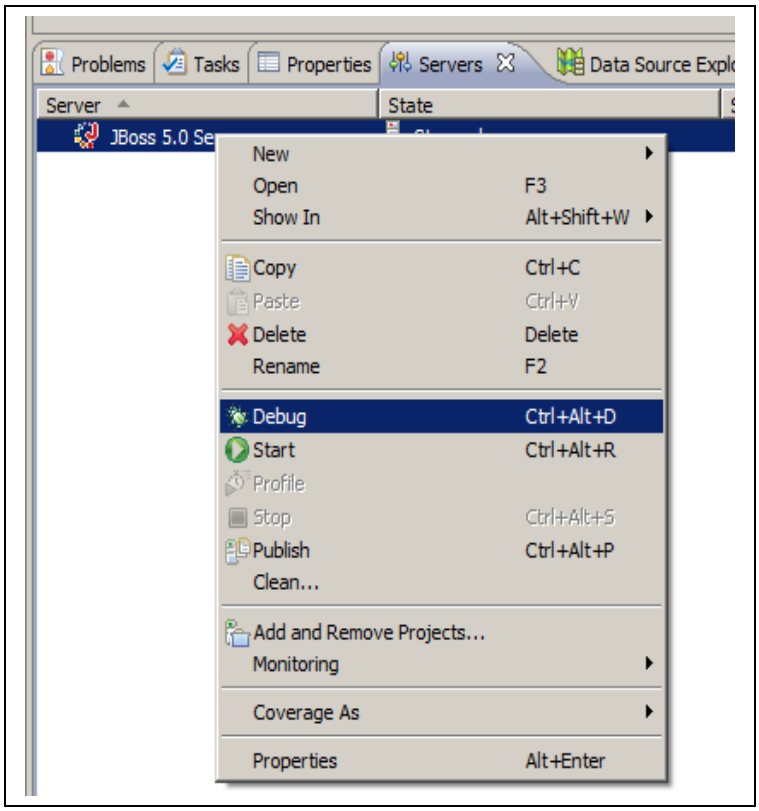

New Server
Define a New Server
Choose the type of server to create
Server's host name: localhost
Download additional server adapters
Select the server type:
type filter text
IBM
JBoss
JBoss, a division of Red Hat
JBoss AS 3.2
JBoss AS 4.0
JBoss AS 4.2
JBoss AS 5.0
JBoss Deploy-Only Server
JBoss Application Server 5.0
Server name: JBoss AS 5.0 at localhost
< Back
Next >
Finish
Cancel


New Server
JBoss Runtime Wizard
JBoss 5.0
A JBoss Server runtime references a JBoss installation directory.
It can be used to set up classpaths for projects which depend on this runtime,
as well as by a "server" which will be able to start and stop instances of JBoss.
Name
JBoss 5.0 Runtime
Home Directory
C:_open-source\jboss-5.0.0.GA
Browse...
JRE
jre1.6.0_02
JRE
Configuration
all
default
minimal
standard
web
< Back
Next >
Finish
Cancel

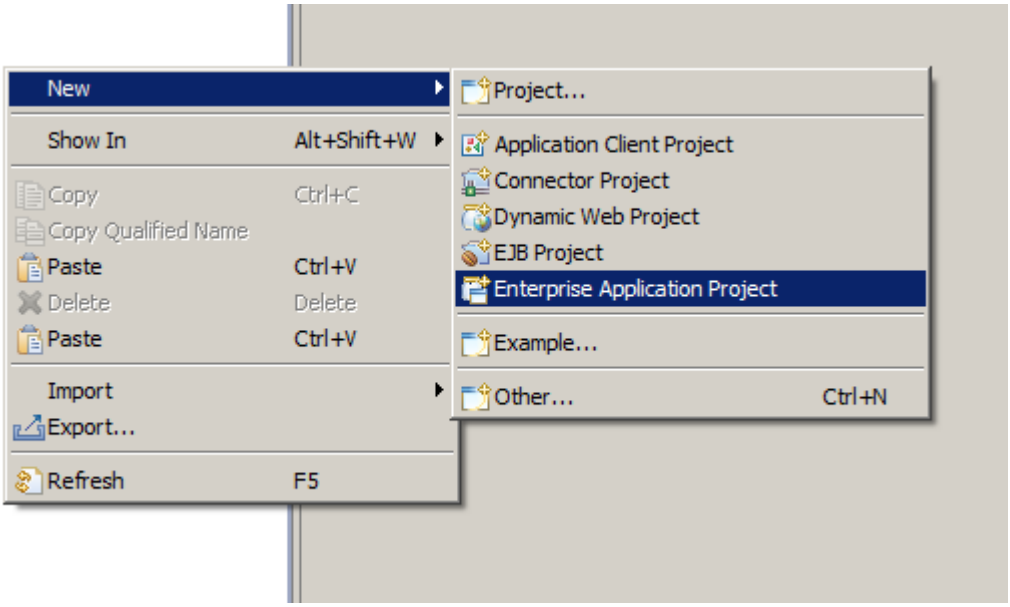

New Server
Create a new JBoss Server
JBoss 5.0
A JBoss Server manages starting and stopping instances of JBoss.
It manages command line arguments and keeps track of which modules have been deployed.
Name
JBoss 5.0 Server
Runtime Information
If the runtime information below is incorrect, please press back, Installed Runtimes...,
and then Add to create a new runtime from a different location.
Home Directory C:_open-source\jboss-5.0.0.GA
JRE C:\Program Files\Java\jre1.6.0_02 (jre1.6.0_02)
Configuration default
Login Credentials
JMX Console Access
User Name
Password
Deployment
Deploy Directory C:_open-source\jboss-5.0.0.GA\server\default\deploy
Browse...
< Back
Next >
Finish
Cancel

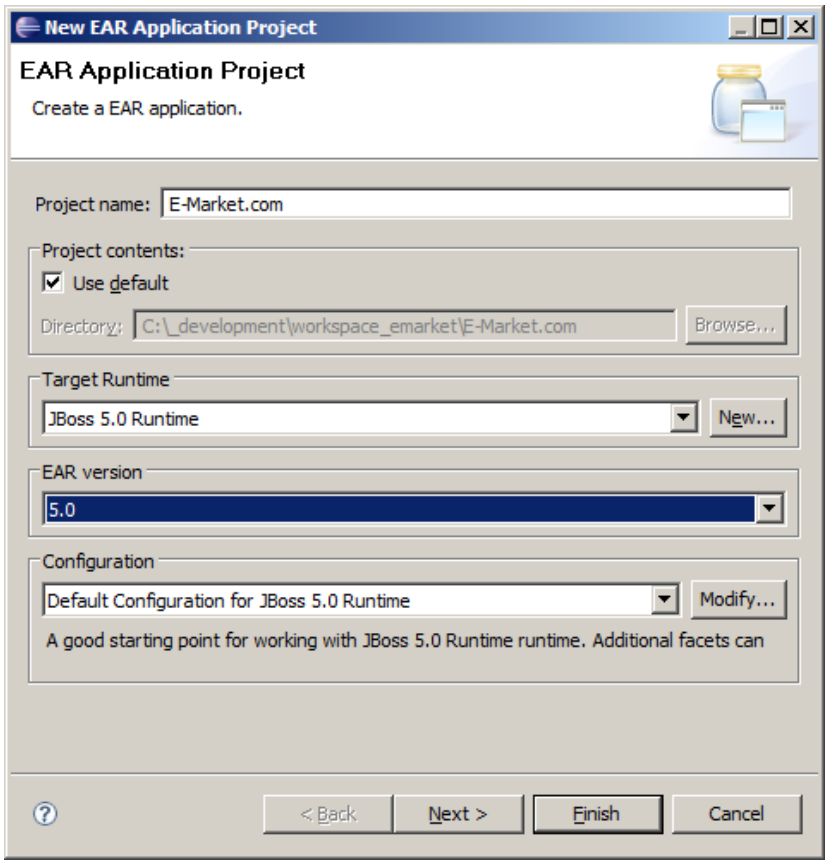

New Server
Add and Remove Projects
Modify the projects that are configured on the server
Move projects to the right to configure them on the server
Available projects:
Configured projects:
Add >
< Remove
Add All >>
<< Remove All
< Back
Next >
Finish
Cancel

Bu işlemlerin ardından Server panelinde (bir sonraki resimde görüldüğü gibi) JBoss 5 Server isminde yeni bir server kaydı oluşacaktır.

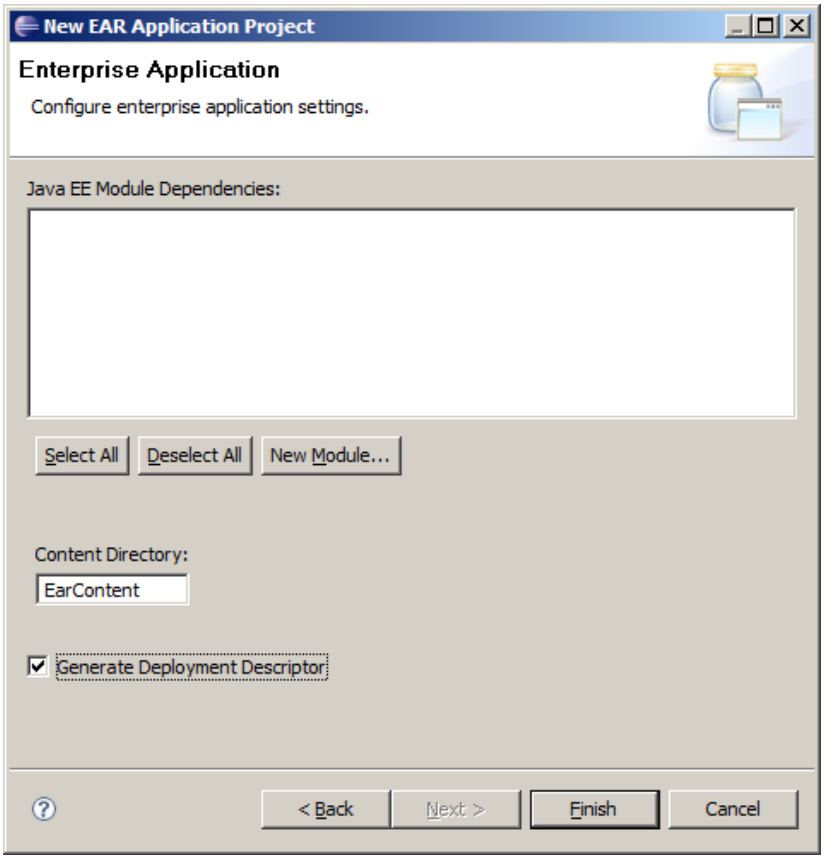


Server hazır ve çalışır durumda. Simdi sıra Java EE projesini oluşturmaya geldi. Project Explorer panelinde sağ tuşa tıkladığımızda bir sonraki resimde yer alan menü oluşur. Buradan ***New > Exterprise Application Project*** opsiyonunu seçerek, Java EE projesini oluşturuyoruz.

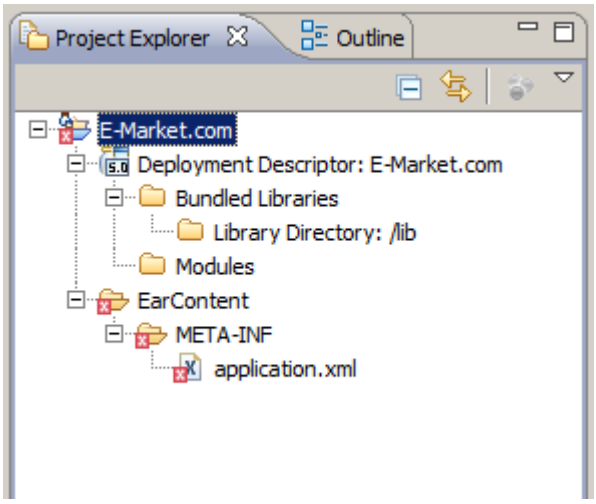




Projeyi E-Market.com olarak isimlendiriyoruz. ***Target Runtime***, daha önce oluşturduğumuz JBoss server konfigürasyonu olan ***JBoss 5.0 Runtime***. Ear versiyonu olarak 5.0 seçiyoruz.

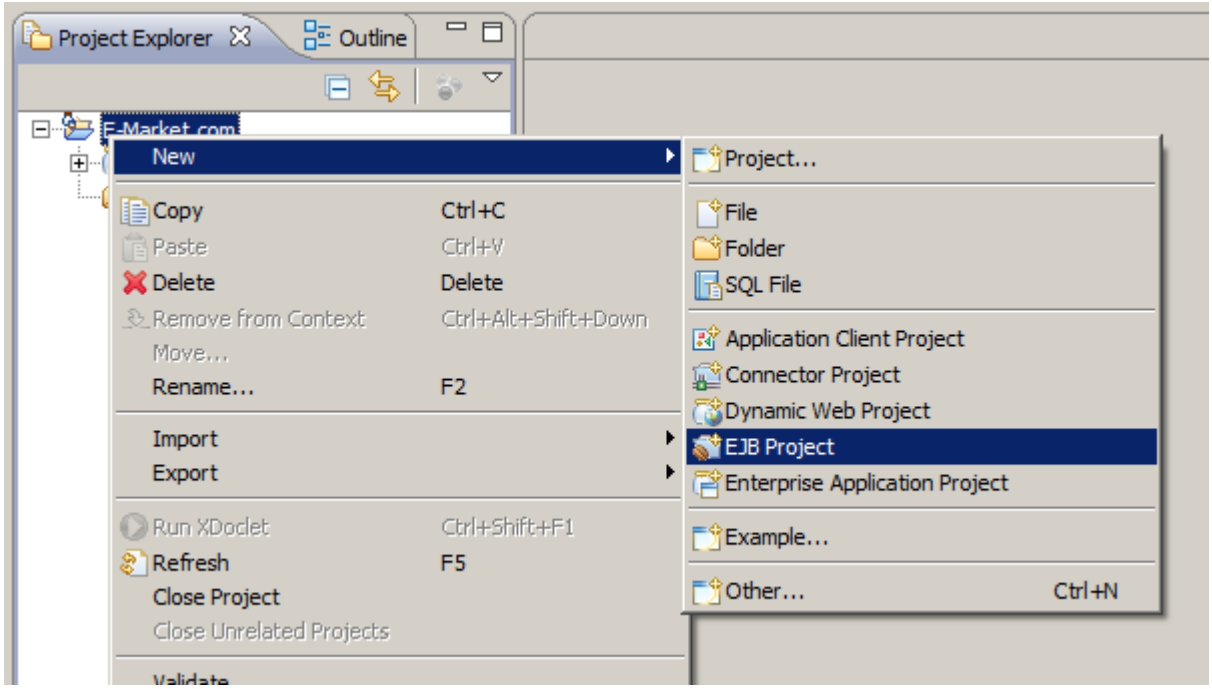


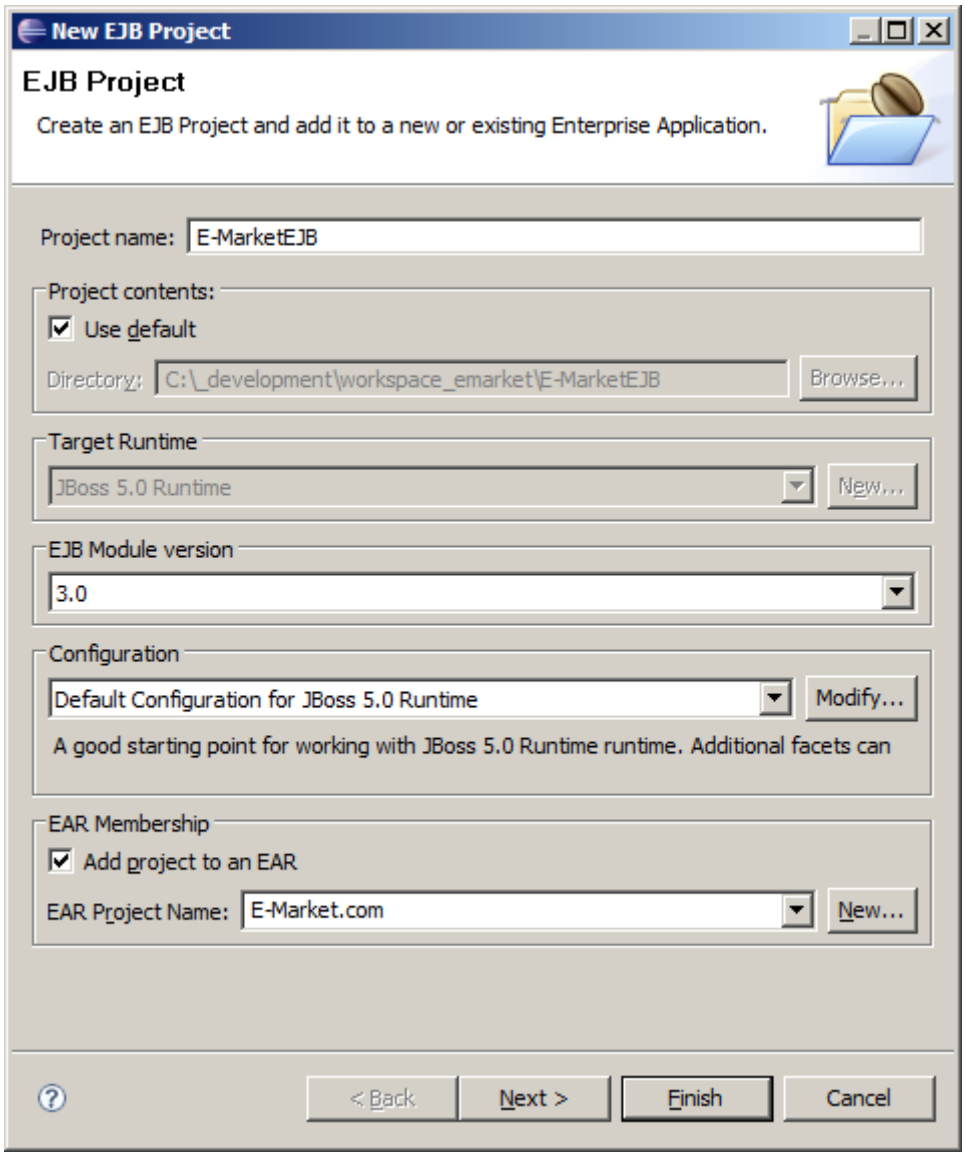
Next ile bir sonraki pencereye geçiyoruz. Bir önceki resimde bu pencere yer almaktadır. Burada ***Generate Deployment Descriptor*** opsiyonu seçmemiz gerekiyor. ***Finish*** ile projeyi oluşturduktan sonra ***Project Explorer*** panelinde E-Market.com isminde yeni bir proje yer alır.



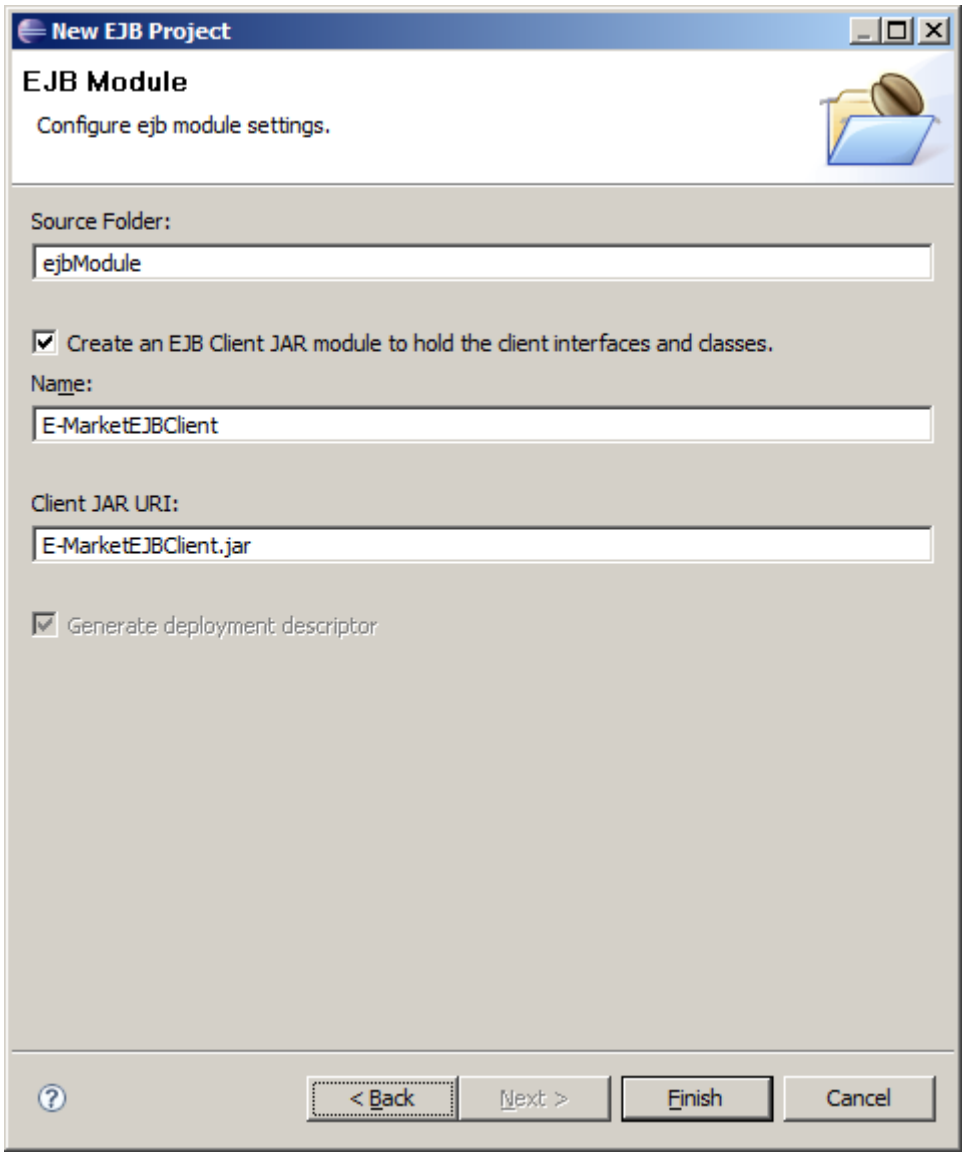
Bu şekilde bir EAR projesi oluşturmuş olduk. Lakin bu boş bir EAR arşiv projesi ve içine JAR ve WAR ihtiva eden modülleri eklememiz gerekiyor. İlk önce bir EJB JAR projesi oluşturuyoruz. Bu proje bünyesinde EJB komponentimiz yer alacak. Daha sonra oluşturacağımız ve EAR arşivinde yer alacak olan web aplikasyonu (WAR arşiv dosyası) EJB komponentlerini kullanarak, işlem yapacak.

Yeni bir EJB projesi oluşturmak için E-Market.com projesinin üzerine sağ tuşla tıklamamız gerekiyor. Aşağıdaki menü oluşacaktır. Bu menüden EJB Project seçeneğini seçerek, EJB projesini oluşturuyoruz.



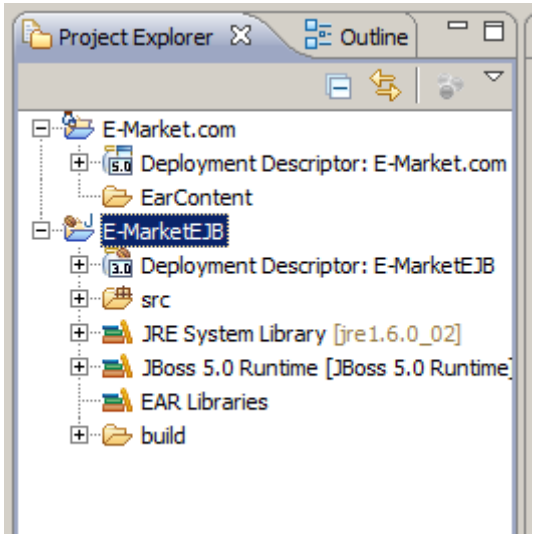


EJB projesi E-MarketEJB ismini taşıyor. ***Target Runtime*** olarak ***JBoss 5.0 Runtime*** kullanıyoruz. ***EJB Module version*** olarak 3.0 seçiyoruz. EJB projesinin bir JAR dosyası olarak EAR arşivine (E-Market.com projesi) dahil edilebilmesi için ***EAR Membership*** bölümünde ***Add project to an EAR*** seçeneğini seçmemiz gerekiyor. Gerekli ayarları yaptıktan sonra ***Next*** butonuna tıklayarak, bir sonraki panele geçiyoruz.

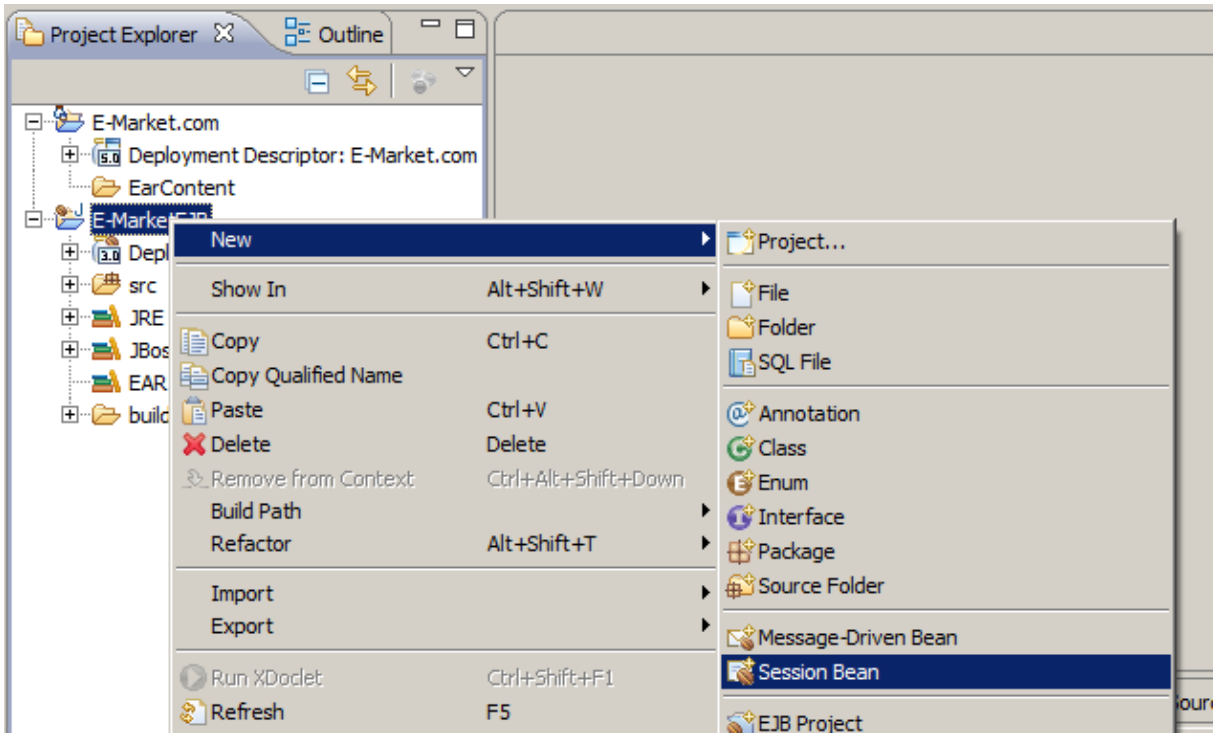


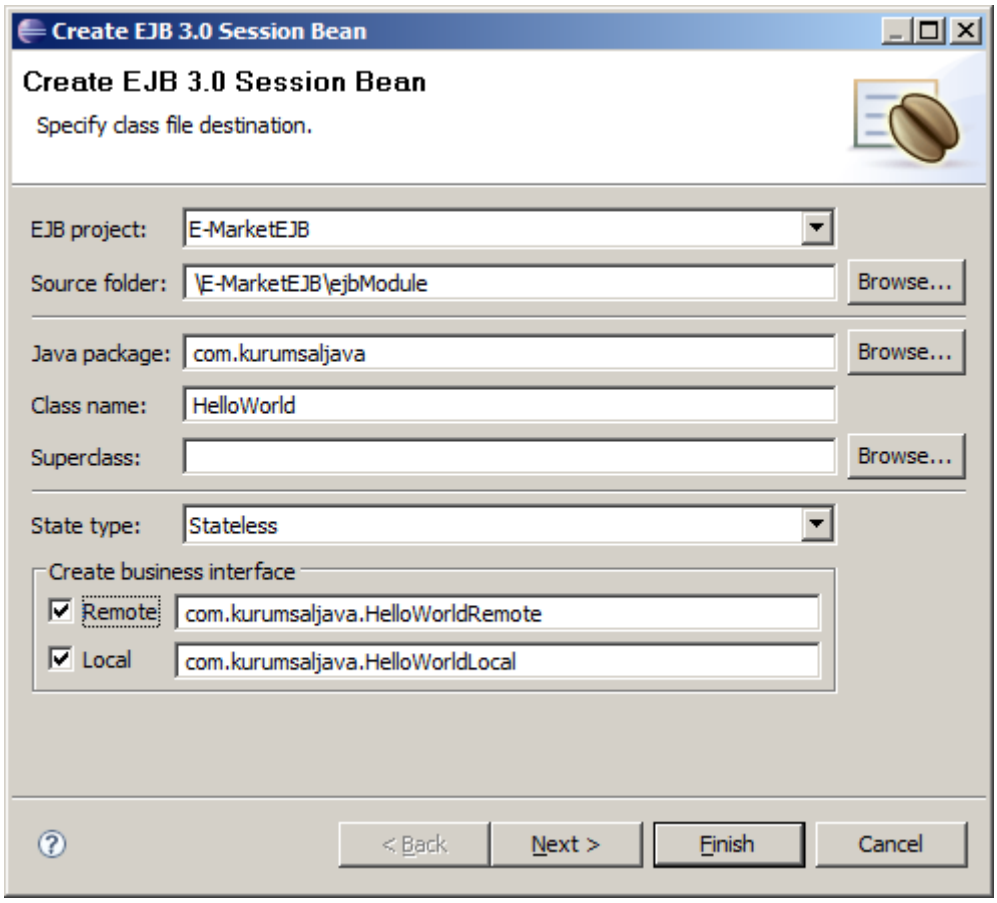
Source Folder olarak ***ejbModule*** ismini tanımlıyoruz. Bir web aplikasyonun ya da başka bir programın EJB komponentlerini kullanabilmesi için, EJB komponentlerin sahip oldukları ***local*** ve ***remote*** interface sınıflarının kendi classpath'ı içinde olması gerekir. Kullanımı kolaylaştırmak için ***local*** ve ***remote*** interface sınıfları bir JAR dosyasına konularak, ortak kullanılmaları sağlanır. Oluşturduğumuz bu JAR dosyası ***EJB Client JAR*** arşivi olarak isimlendirilir. Bir önceki resimde görüldüğü gibi eğer ***Create an EJB Client JAR module to hold the client interfaces and classes*** seçeneğini seçersek, Eclipse otomatik olarak **E-MarketEJBClient** ismini taşıyan yeni bir proje oluşturarak, EJB komponentin sahip olduğu ***local*** ve ***remote*** interface sınıflarını bu projeye dahil edecektir. Ayrıca bu JAR dosyası EAR arşivine dahil edilerek, EAR içinde EJB komponentlerini kullanan diğer modüllerin ***local*** ve ***remote*** interface sınıflarını kullanmaları sağlanır.

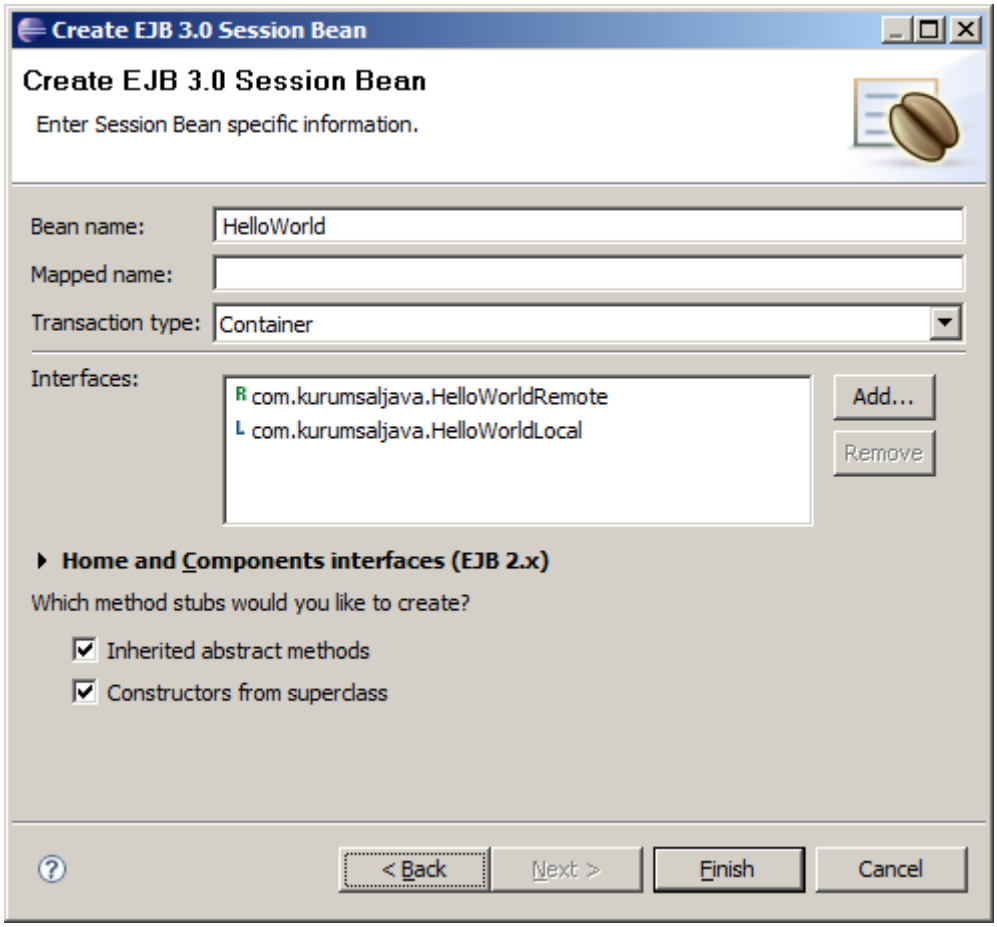
Bu işlemlerin ardından ***Project Explorer*** panelinde aşağıdaki resimde yer alan EAR projesi (modülü) ve EJB projesi (modülü) yer alacaktır.

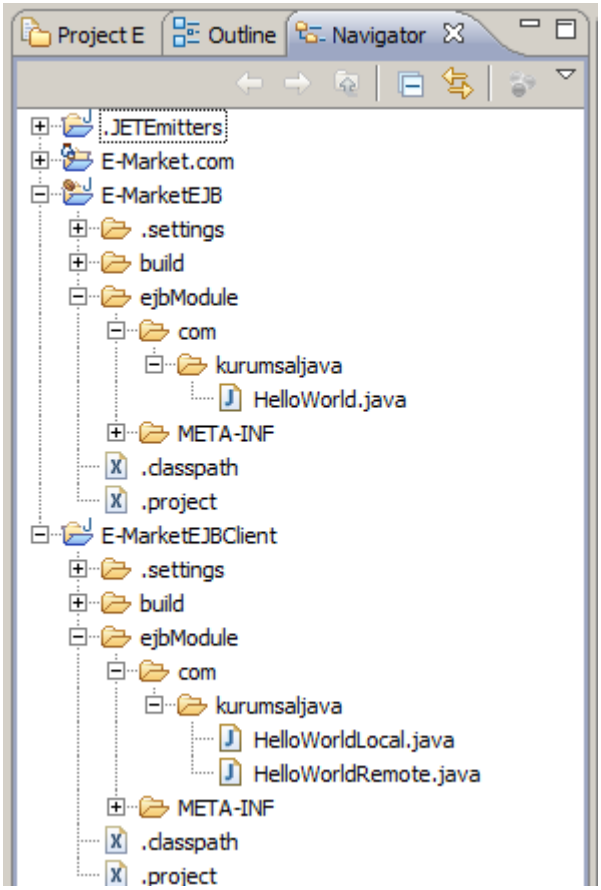


Yeni bir EJB komponenti oluşturmak için, E-MarketEJB projesini seçerek, sağ tuşa tikliyoruz. Oluşan menüdeki ***Session Bean*** seçeneği üzerinden yeni bir EJB komponenti oluşturabiliriz.




Create EJB 3.0 Session Bean
Create EJB 3.0 Session Bean
Specify class file destination.
EJB project:
E-MarketEJB
Source folder:
\E-MarketEJB\ejbModule
Browse...
Java package:
com.kurumsaljava
Browse...
Class name:
HelloWorld
Superclass:
Browse...
State type:
Stateless
Create business interface
Remote
com.kurumsaljava.HelloWorldRemote
Local
com.kurumsaljava.HelloWorldLocal
< Back
Next >
Finish
Cancel


Create EJB 3.0 Session Bean
Create EJB 3.0 Session Bean
Enter Session Bean specific information.
Bean name:
HelloWorld
Mapped name:
Transaction type:
Container
Interfaces:
R com.kurumsaljava.HelloWorldRemote
L com.kurumsaljava.HelloWorldLocal
Add...
Remove
Home and Components interfaces (EJB 2.x)
Which method stubs would you like to create?
Inherited abstract methods
Constructors from superclass
< Back
Next >
Finish
Cancel



EJB komponenti ***HelloWorld*** ismini taşıyor ve ***Session Bean*** tipine sahip. Bir önceki resimde görüldüğü gibi ***HelloWorld*** isimli EJB komponenti ***E-MarketEJB*** projesinin ejbModule dizinine eklendi. Bu komponentin sahip olduğu ***local*** (HelloWorldLocal.java) ve ***remote*** (HelloWorldRemote.java) interface sınıfları E-MarketEJBClient projesinin ejbModule dizinine eklendi.

Daha sonra Wicket ile oluşturacağımız web aplikasyonumuz ***HelloWorld*** isimli EJB komponentimizi kullanacak. Bir EJB komponentinin sahip olduğu bir metodu kullanabilmemiz için, bu metodu HelloWorldRemote.java ve HelloWorldLocal.java interface sınıflarında tanımlamamız gerekiyor. Bunu aşağıdaki şekilde yapıyoruz.

```
package com.kurumsaljava;
import javax.ejb.Remote;

@Remote
public interface HelloWorldRemote
{
      String getText();
}

.....................................................

package com.kurumsaljava;
import javax.ejb.Local;

@Local
public interface HelloWorldLocal
{
```

```
      String getText();

}
```

EJB komponenti bilgisayar ağı içindeki herhangi bir bilgisayardan kullanabilmek için ***HelloWorldRemote*** interface sınıfı ile çalışmamız gerekiyor. Eğer bir aplikasyon EJB komponenti ile aynı aplikasyon serveri içinde ise, o zaman EJB komponenti ***local*** interface üzerinden kullanmak mümkündür. Bu şekilde EJB komponentine erişim, ağ üzerinden gerçekleşmediği, direk olduğu için daha hızlı olacaktır.

HelloWorldRemote ve ***HelloWorldLocal*** interface sınıflarına eklediğimiz yeni metodu ***HelloWorld*** EJB sınıfında implemente etmemiz gerekiyor. Bunu şu şekilde yapıyoruz.

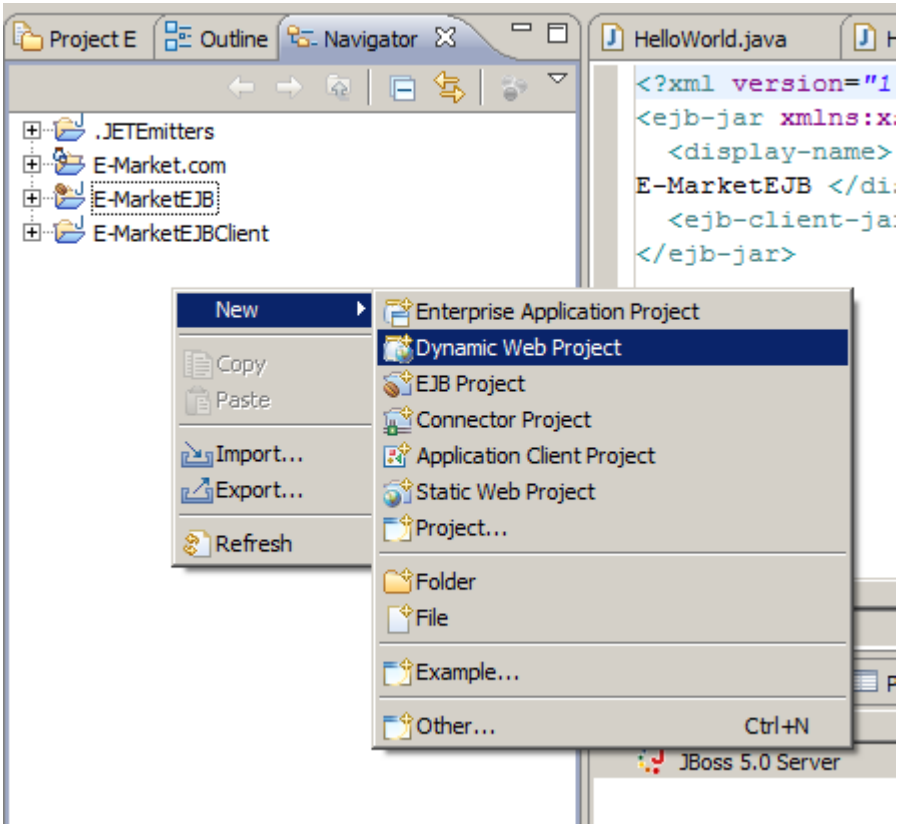
```
package com.kurumsaljava;

import javax.ejb.Stateless;

@Stateless
public class HelloWorld implements HelloWorldRemote, HelloWorldLocal {

    public HelloWorld()
    {
    }

      @Override
      public String getText()
      {
            return "Hello World";
      }

}
```

EJB komponentimiz bu şekilde kullanıma hazır hale gelmiştir. Şimdiye kadar yaptıklarımızı özetleyecek olursak:

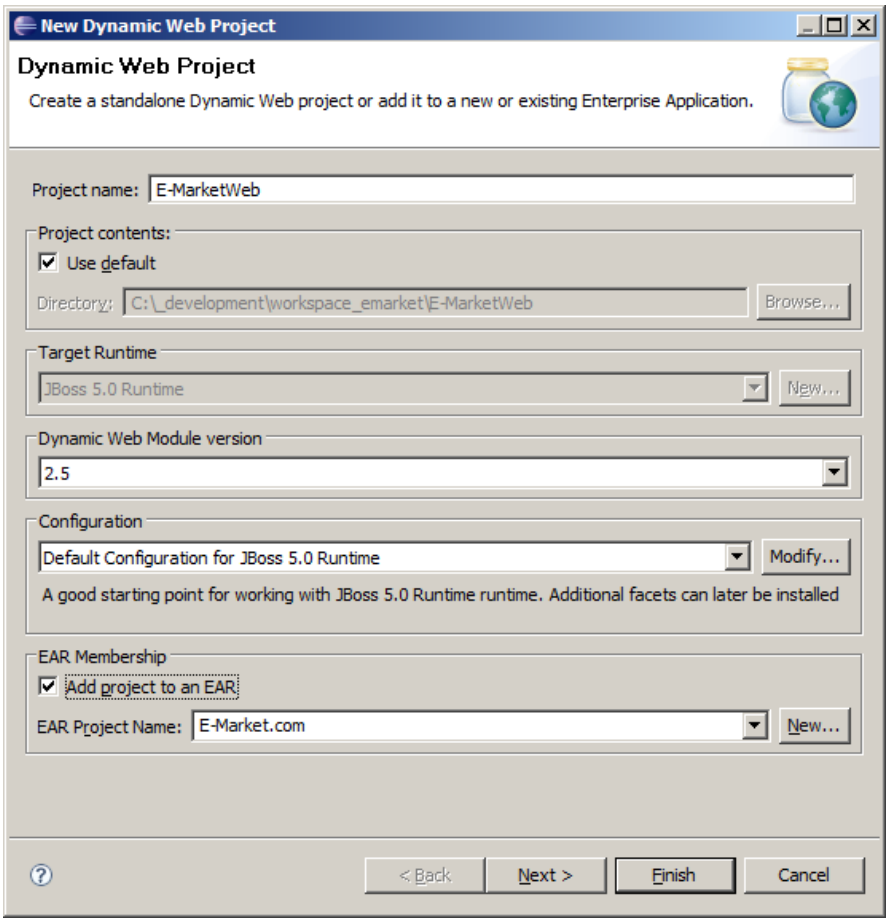
- E-Market.com ismini taşıyan bir ***Enterprise Application Project*** oluşturduk. Bu bir EAR (Enterprise Archive) arşiv projesidir. Bu proje EJB ve web aplikasyonunu komponentlerini ihtiva edecek.
- E-MarketEJB isminde bir ***EJB Project*** oluşturduk. Bu proje bünyesinde HelloWorld EJB komponenti yer almaktadır.
- Eclipse bizim için içinde HelloWorld EJB komponentinin local ve remote interface sınıflarının yer aldığı E-MarketEJBClient isminde bir proje oluşturdu.

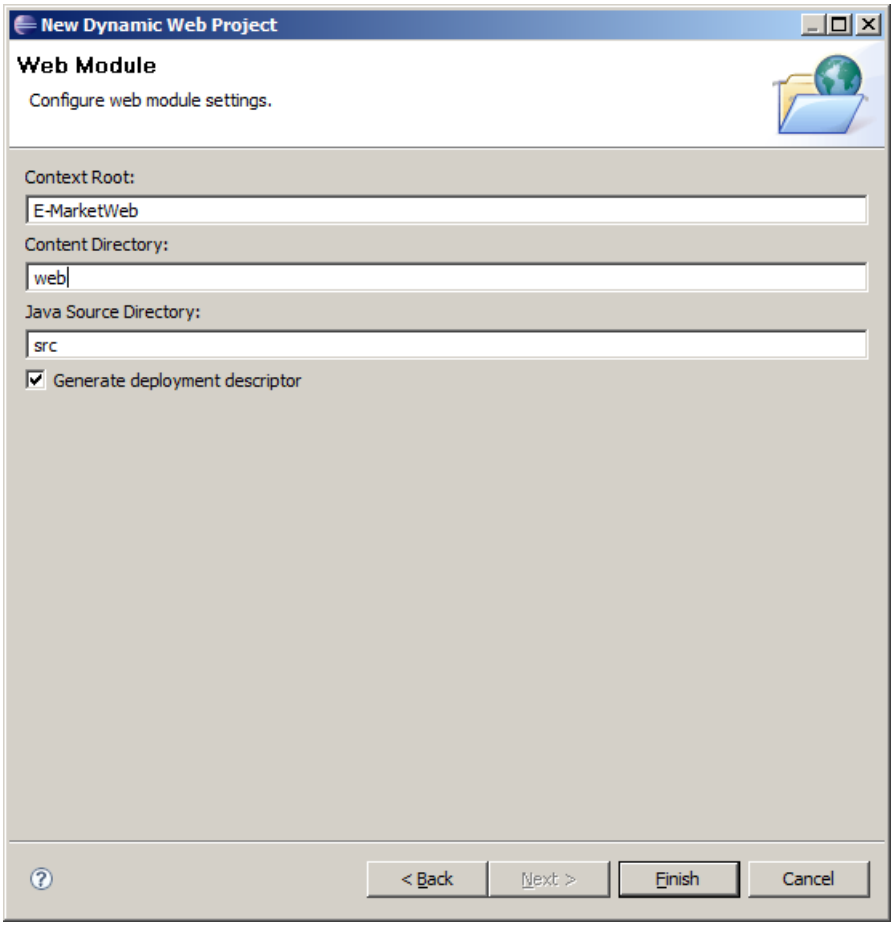
Bu işlemlerin ardından Wicket[1] web frameworkü ile bir web aplikasyonu oluşturmak için kolları sıvıyoruz.

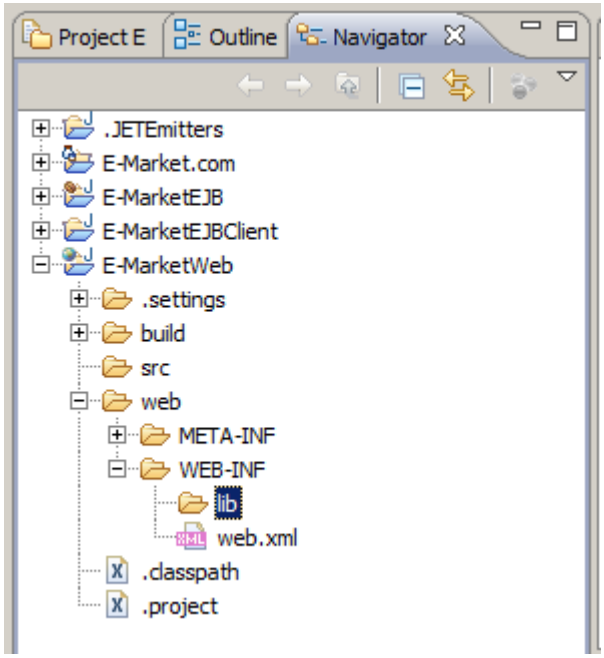
[1] http://www.kurumsaljava.com/2008/12/25/web-framework-gokyuzunde-yeni-bir-yildiz-wicket/



Project Explorer ya da ***Navigator*** paneli içinde sağ tuşa tıklayarak, bir önceki resimde görüldüğü gibi bir web projesi oluşturmak için ***Dynamic Web Project*** opsiyonunu seçiyoruz.

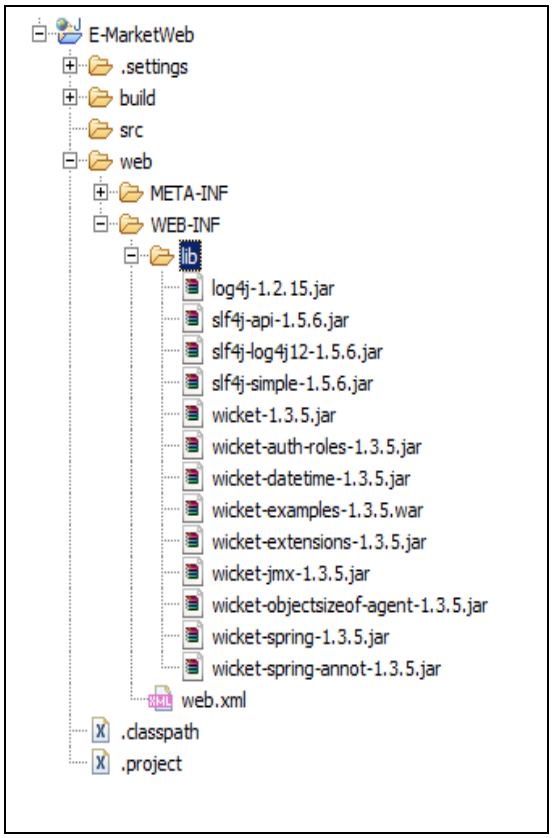

New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: E-MarketWeb
Project contents:
Use default
Directory: C:_development\workspace_emarket\E-MarketWeb
Browse...
Target Runtime
JBoss 5.0 Runtime
New...
Dynamic Web Module version
2.5
Configuration
Default Configuration for JBoss 5.0 Runtime
Modify...
A good starting point for working with JBoss 5.0 Runtime runtime. Additional facets can later be installed
EAR Membership
Add project to an EAR
EAR Project Name: E-Market.com
New...
< Back
Next >
Finish
Cancel


New Dynamic Web Project
Web Module
Configure web module settings.
Context Root:
E-MarketWeb
Content Directory:
web
Java Source Directory:
src
Generate deployment descriptor
< Back
Next >
Finish
Cancel



E-MarketWeb ismini taşıyan web projemiz bir önceki resimdeki yapıya sahiptir. Wicket için gerekli tüm kütüphanaleri WEB-INF/lib dizinine yerleştirmemiz gerekiyor. http://wicket.apache.org adresinden Wicket 1.3 sürümünü edindikten sonra bir sonraki resimde yer aldığı gibi gerekli kütüphaneleri lib dizinine yerleştiriyoruz.

Wicket logging işlemleri için SLF4J ve log4J kütüphanelerini kullanıyor. SLF4J kütüphanesini http://www.slf4j.org/download.html adresinden, log4J kütüphanesini http://logging.apache.org/log4j/1.2/download.html adresinden temin edebilirsiniz.



HelloWorldApplication ismini taşıyan sınıf ile Wicket tabanlı web aplikasyonunu oluşturmaya başlıyoruz. ***HelloWorldApplication*** aşağıdaki yapıya sahiptir.

```
package com.kurumsaljava.ee;

import org.apache.wicket.protocol.http.WebApplication;

public class HelloWorldApplication extends WebApplication
{
    public HelloWorldApplication()
    {

    }


    public Class getHomePage()
    {
        return HelloWorldPage.class;
    }
}
```

HelloWorldApplication web aplikasyonunu başlangıç sınıfıdır. Bu sınıf bünyesinde web aplikasyonun başlangıç sayfası tanımlanır. getHomePage() metodunda ***HelloWorldPage*** ismini taşıyan ilk Wicket sayfasını tanımlıyoruz. Bu sınıfın yapısı aşağıda yer almaktadır.

```
package com.kurumsaljava.ee;

import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.markup.html.basic.Label;

public class HelloWorldPage extends WebPage
{
      public HelloWorldPage()
    {
        add(new Label("message", "Hello World!"));
    }

}
```

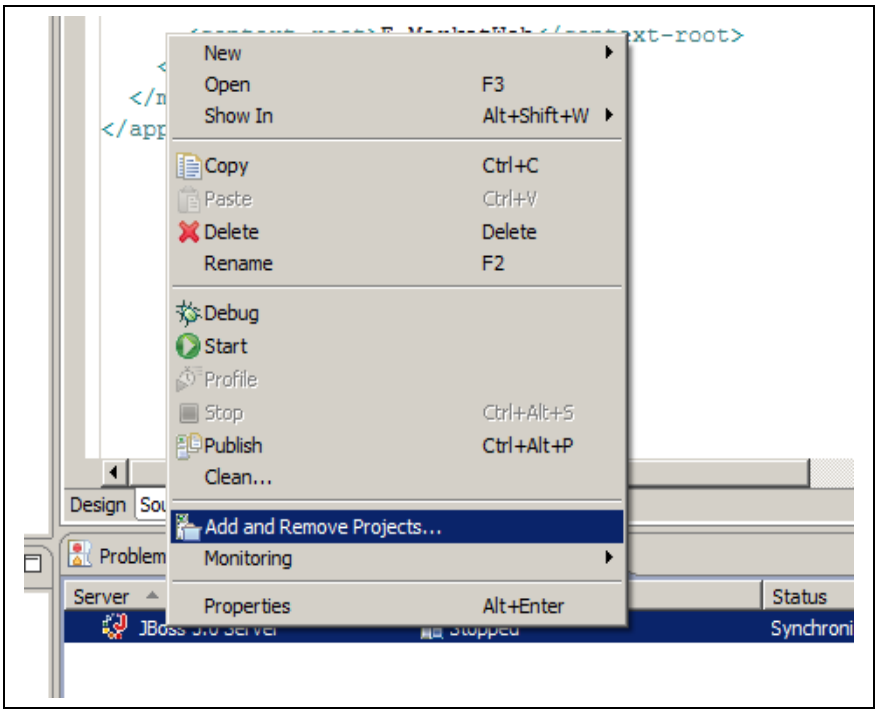
HTML sayfasını HelloWorldPage.html olarak aşağıdaki şekilde tanımlıyoruz.

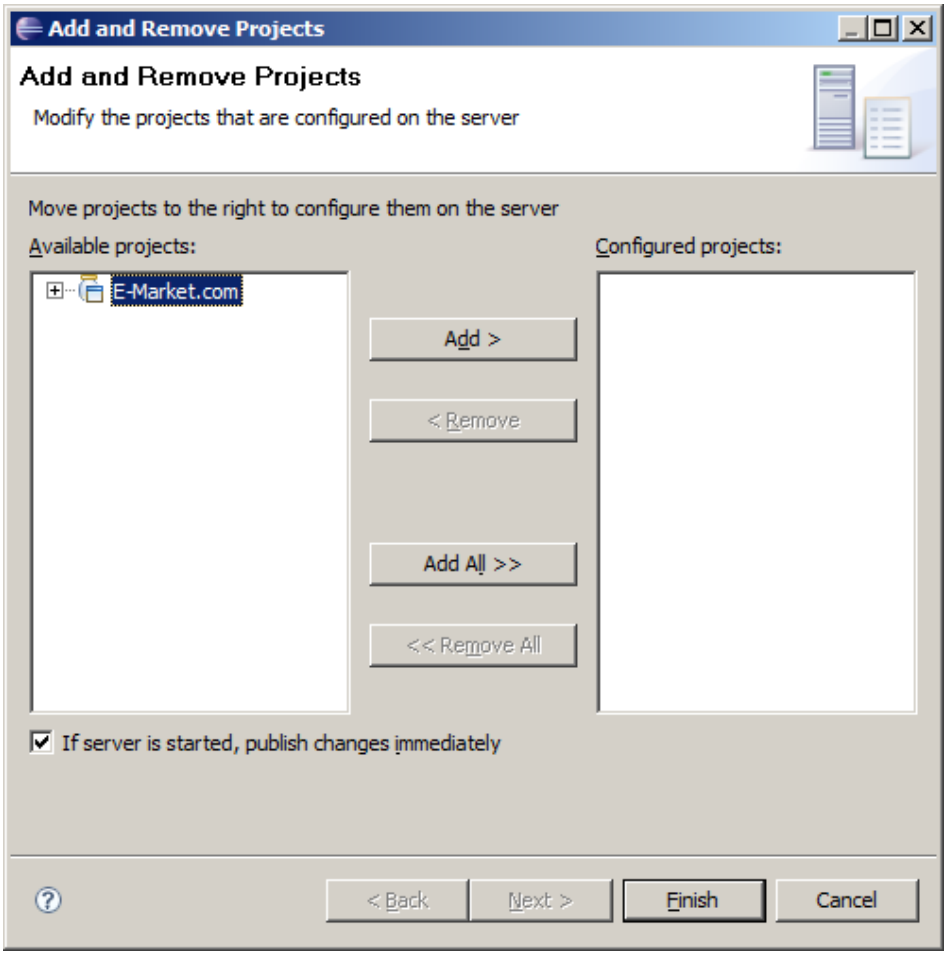
```
<html>
<body>
  <span wicket:id="message" id="message">Mesaj buraya eklemecek</span>
</body>
</html>
```

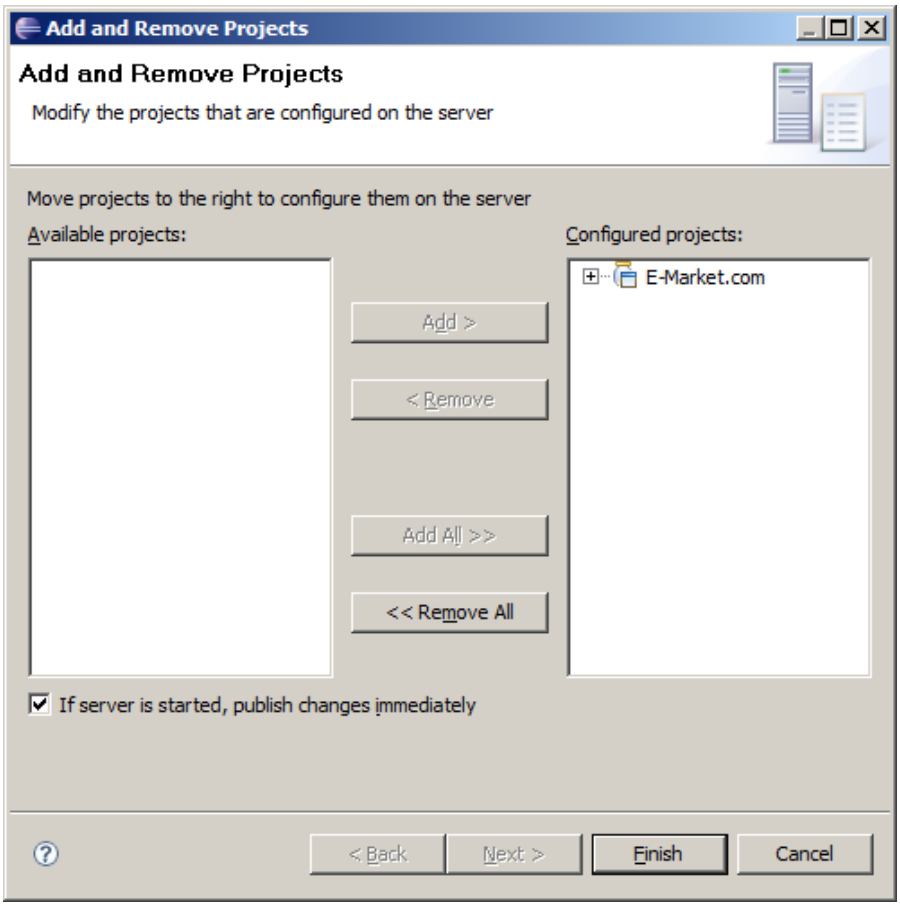
web.xml dosyasında aşağıdaki değişiklikleri yaparak, Wicket web aplikasyonumuzu çalışır hale getiriyoruz.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
  <display-name>E-MarketWeb</display-name>

  <filter>
        <filter-name>HelloWorldApplication</filter-name>
        <filter-
class>org.apache.wicket.protocol.http.WicketFilter</filter-class>
        <init-param>
          <param-name>applicationClassName</param-name>
          <param-value>com.kurumsaljava.ee.HelloWorldApplication</param-
value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>HelloWorldApplication</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

</web-app>
```

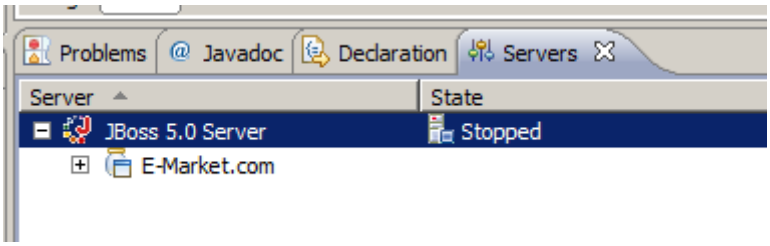
Web aplikasyonumuzda bu şekilde tamamlanmış oldu. EJB komponentini ve web aplikasyonu çalışır hale getirmek için, daha önce oluşturduğumuz JBoss serverine E-Market.com EAR arşivini eklememiz gerekiyor. Bu işlemin nasıl yapıldığı aşağıdaki resimlerde yer almaktadır.

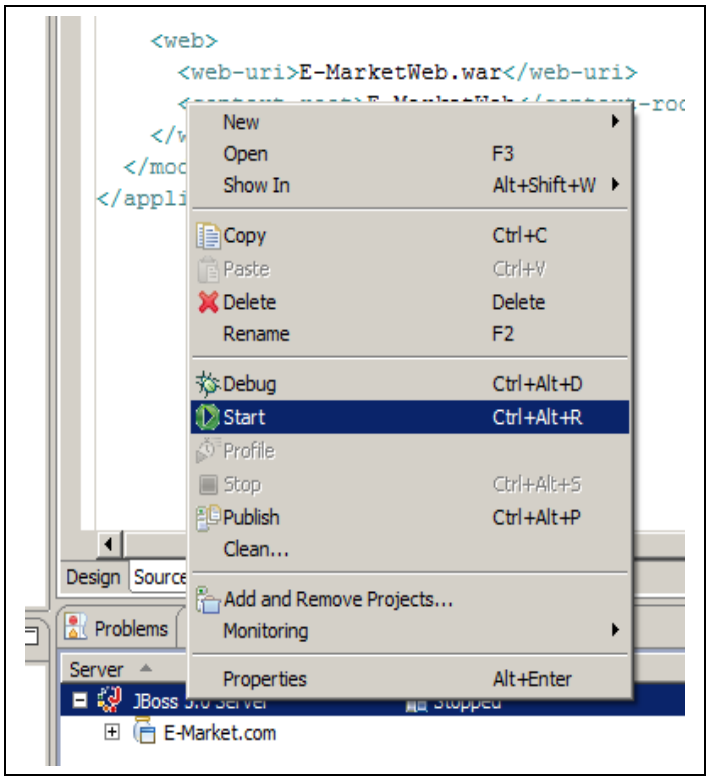

New
Open F3
Show In Alt+Shift+W
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Rename F2
Debug
Start
Profile
Stop Ctrl+Alt+S
Publish Ctrl+Alt+P
Clean...
Add and Remove Projects...
Monitoring
Properties Alt+Enter
Design
Server
Status
Synchroni


Add and Remove Projects
Add and Remove Projects
Modify the projects that are configured on the server
Move projects to the right to configure them on the server
Available projects:
Configured projects:
E-Market.com
Add >
< Remove
Add All >>
<< Remove All
If server is started, publish changes immediately
< Back
Next >
Finish
Cancel



E-Market.com EAR projesini JBoss serverine ekledikten sonra, bir sonraki resimde görüldüğü gibi EAR projesi JBoss serverinde çalışabilir hale gelir.





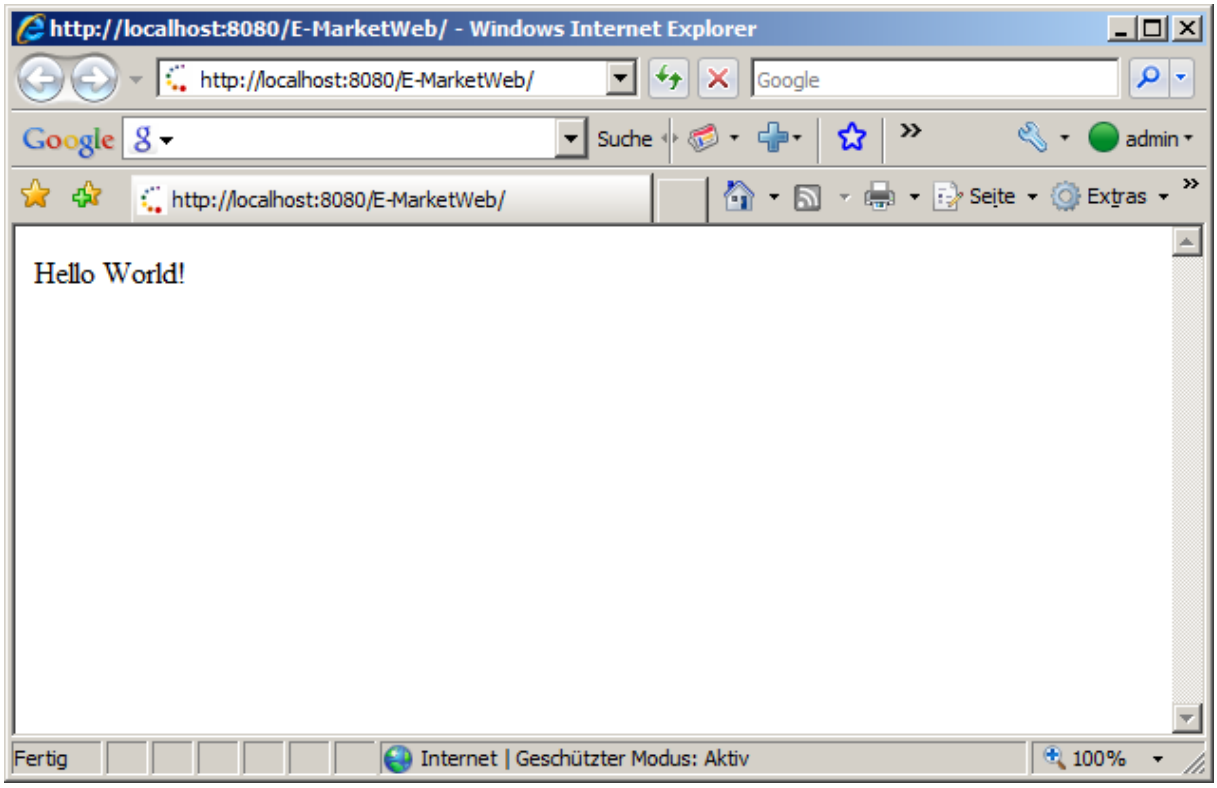
Start menüsü üzerinden E-Market.com EAR projesini aktif hale getirebiliriz. Console panelinde JBoss serveri çalıştıktan sonra, EJB komponentinin çalışır hale geldiğine dair aşağıdaki log yer alacaktır.

```
18:25:20,158 INFO  [JBossASKernel] Created KernelDeployment for: E-
MarketEJB.jar
18:25:20,158 INFO  [JBossASKernel] installing bean: jboss.j2ee:ear=E-
Market.com.ear,jar=E-MarketEJB.jar,name=HelloWorld,service=EJB3
18:25:20,158 INFO  [JBossASKernel]   with dependencies:
18:25:20,158 INFO  [JBossASKernel]   and demands:
18:25:20,158 INFO  [JBossASKernel]  jboss.ejb:service=EJBTimerService
18:25:20,158 INFO  [JBossASKernel]   and supplies:
18:25:20,158 INFO  [JBossASKernel]
      Class:com.kurumsaljava.HelloWorldRemote
18:25:20,158 INFO  [JBossASKernel]  jndi:E-Market.com/HelloWorld/local-
com.kurumsaljava.HelloWorldLocal
18:25:20,159 INFO  [JBossASKernel]  jndi:E-Market.com/HelloWorld/remote
18:25:20,159 INFO  [JBossASKernel]  jndi:E-Market.com/HelloWorld/local
18:25:20,159 INFO  [JBossASKernel]  Class:com.kurumsaljava.HelloWorldLocal
18:25:20,159 INFO  [JBossASKernel]  jndi:E-Market.com/HelloWorld/remote-
com.kurumsaljava.HelloWorldRemote
```

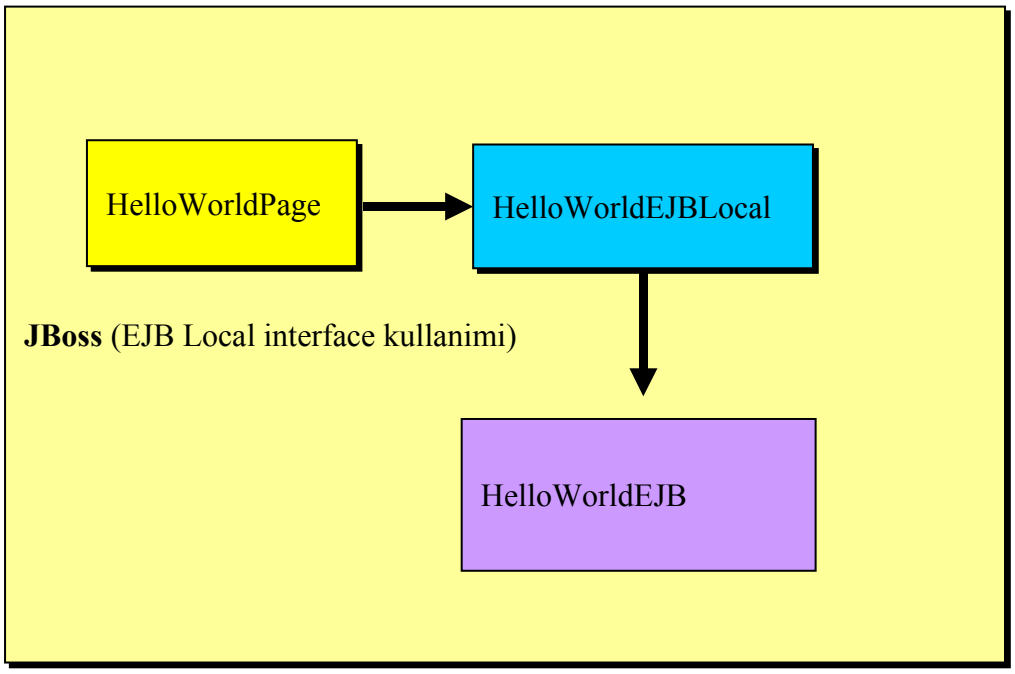
Logda görüldüğü gibi JBoss E-MarketEJB.jar dosyasında bulunan ***HelloWorld*** isimli EJB komponentini çalışır hale getirmiştir (deploy). E-Market.jar arşivi E-Market.com.ear arşivinde yer almaktadır.

```
18:25:20,596 INFO  [TomcatDeployment] deploy, ctxPath=/E-MarketWeb,
vfsUrl=E-Market.com.ear/E-MarketWeb.war
1
```

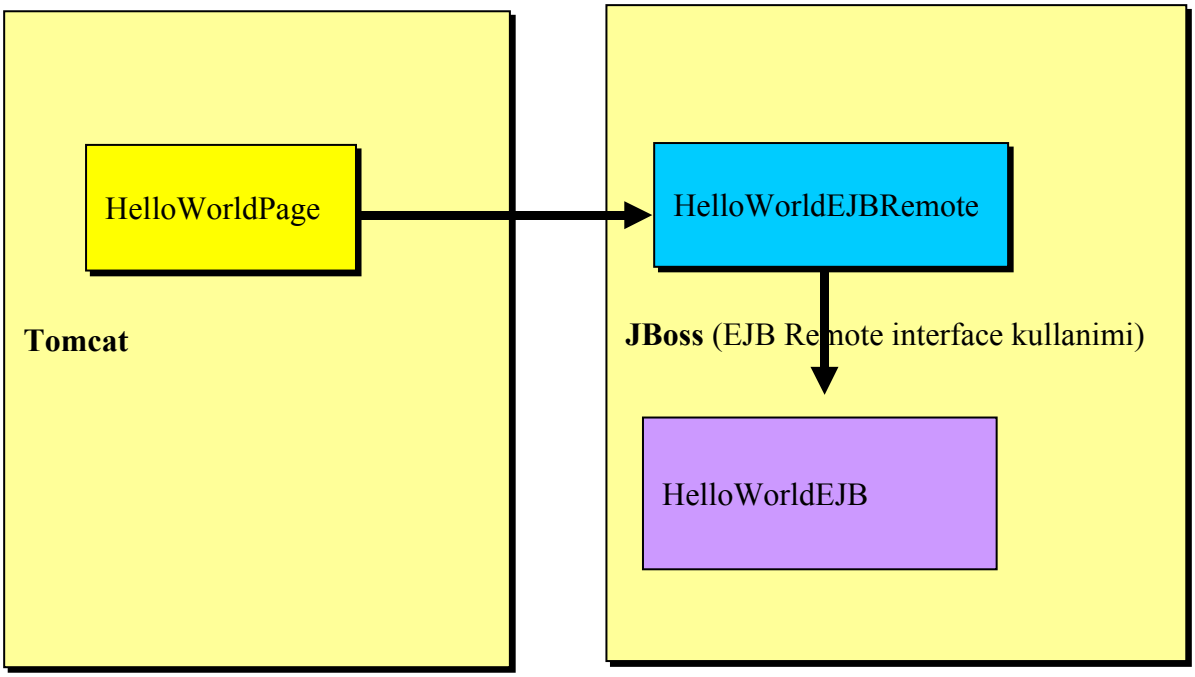
JBoss, EJB komponenti yanı sıra, yine E-Market.com.ear paketinde bulunan E-MarketWeb.war arşivindeki web aplikasyonu çalışır hale getirmiştir. Bir önceki logda görüldüğü gibi web aplikasyonun erişim adresi /E-MarketWeb dir, yani web aplikasyonuna http://localhost:8080/E-MarketWeb adresinden erişebiliriz.



Web aplikasyonumuz çalışır hale geldi ve bir web tarayıcısı üzerinden erişilir durumda. Ekranda yer alan Hello World! ***HelloWorldpage*** sayfasında kullandığımız Label komponentinin değeridir. Şimdi ***HelloWorld*** EJB komponentinin getText() metodunu kullanarak, bu değere ulaşalım. Bunun için ***HelloWorldPage*** sınıfının EJB komponentine local interface üzerinden bağlantı kurarak, gerekli veriyi edinmesi gerekiyor. Web aplikasyon ile EJB komponenti aynı EAR arşivinde ve aynı JBoss serveri üzerinde bulunduğu için, web aplikasyonundaki herhangi bir sınıf EJB komponentin ***local*** interface sınıfını kullanarak, bu EJB komponentinin sahip olduğu metotları kullanabilir. Eğer web aplikasyonu ve EJB komponenti değişik aplikasyon serverlerinde çalışıyor durumda iseler, bu durumda web aplikasyonundaki bir sınıf sadece EJB komponentinin ***remote*** interface sınıfı üzerinden EJB metotlarını kullanabilir.



Bir önceki diagramda görüldüğü gibi EJB komponenti ile aynı aplikasyon server içinde bulunan sınıflar, EJB komponentinin ***local*** interface sınıfı üzerinden EJB komponentin metotlarını kullanabilirler.



Bazı gereksinimlerden dolayı web aplikasyonu ve EJB komponenti değişik aplikasyon serverlerinde konuşlandırılabilirler. Web aplikasyonu için bir Tomcat ve EJB aplikasyonu için bir JBoss aplikasyon serveri değişik sunucular üzerinde çalıştırılır. Bu durumda web aplikasyon bünyesinde bulunan herhangi bir sınıf ***remote*** interface (HelloWorldEJBRemote) üzerinden EJB komponenti kullanabilir. EJB komponenti kullanabilmek için sunucuların aynı ağ içinde olmaları gerekmektedir.

Oluşturduğumuz web aplikasyonu EJB komponenti ile aynı aplikasyon serveri bünyesinde bulunduğu için, EJB komponenti aşağıdaki örnekte yer aldığı gibi ***local*** interface üzerinden kullanabiliriz.

```
package com.kurumsaljava.ee;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.markup.html.basic.Label;
import com.kurumsaljava.HelloWorldLocal;

public class HelloWorldPage extends WebPage
{
      public HelloWorldPage()
    {
        try
        {
                  add(new Label("message", getText()));
            }
        catch (NamingException e)
        {
                  e.printStackTrace();
            }
    }

      private String getText() throws NamingException
      {
            Context context = new InitialContext();
            HelloWorldLocal local = (HelloWorldLocal)
                  context.lookup("E-Market.com/HelloWorld/local");
            return local.getText();
      }

}
```

Web aplikasyonun ve EJB komponentin değişik aplikasyon serverlerinde olduğunu farz edelim. Bu durumda, aşağıdaki şekilde ***remote*** interface sınıfı üzerinden EJB komponentini kullanabiliriz.

```
package com.kurumsaljava.ee;

import java.util.Properties;

import javax.naming.Context;
import javax.naming.NamingException;
import org.apache.wicket.markup.html.WebPage;
import org.apache.wicket.markup.html.basic.Label;
import com.kurumsaljava.HelloWorldRemote;

public class HelloWorldPage extends WebPage
{
      public HelloWorldPage()
```

```
    {
        try
        {
                add(new Label("message", getText()));
            }
        catch (NamingException e)
        {
                e.printStackTrace();
            }
    }

      private String getText() throws NamingException
      {
            Context context = getInitialContext();
            HelloWorldRemote local = (HelloWorldRemote)
                   context.lookup("E-Market.com/HelloWorld/remote");
            return local.getText();
      }

      private Context getInitialContext() throws NamingException
      {
            Properties p = new Properties();
            p.put(Context.INITIAL_CONTEXT_FACTORY,
                        "org.jnp.interfaces.NamingContextFactory");
            p.put(Context.URL_PKG_PREFIXES,
"org.jboss.naming:org.jnp.interfaces");
            p.put(Context.PROVIDER_URL, "jnp://localhost:1099");
            return new javax.naming.InitialContext(p);
      }

}
```

Ağ içindeki herhangi bir EJB komponentini kullanabilmek için, EJB komponentin içinde bulunduğu aplikasyon serverin IP adresini bilmemiz gerekiyor. getInitialContext() metodunda görüldüğü gibi ***PROVIDER_URL*** olarak ***localhost:1099*** yani ***127.0.0.1:1099*** IP adres ve portunu kullanıyoruz. Web aplikasyonumuz EJB komponenti ile aynı aplikasyon serveri bünyesinde olsa bile, remote interface sınıfı üzerinden EJB komponentini kullanabiliyoruz. Ama bu performans açısından doğru bir yaklaşım değil. ***Remote*** interface kullanıldığında iletişim ağ üzerinden gerçekleştiği için, işlem süresi uzamaktadır. Bu yüzden web aplikasyonu ile EJB komponentin aynı aplikasyon serverinde olmaları durumunda ***local*** interface kullanılmalıdır. ***Local*** interface kullanıldığında sanki new operatörü kullanılıp, bir nesne oluşturuluyormuş gibi EJB komponenti kullanılabilir. Bu durumda EJB kullanımı daha performanslı bir hale gelmektedir.

EOF (End of Fun)