

Üst Düzey Programlama

Struts Framework

JSP MODEL 1 ve MODEL 2 Mimarileri

Bu mimariler bir web uygulaması geliştirilirken kullanılan yöntemlerdir.

Bu yöntemler arasındaki temel fark, kullanıcıdan gelen isteğin (request) kim tarafından ve nasıl işleneceğidir.

MODEL 1

JSP sayfası isteğin tüm işlemlerini yapar ve istemciye çıktıyı gösterir.

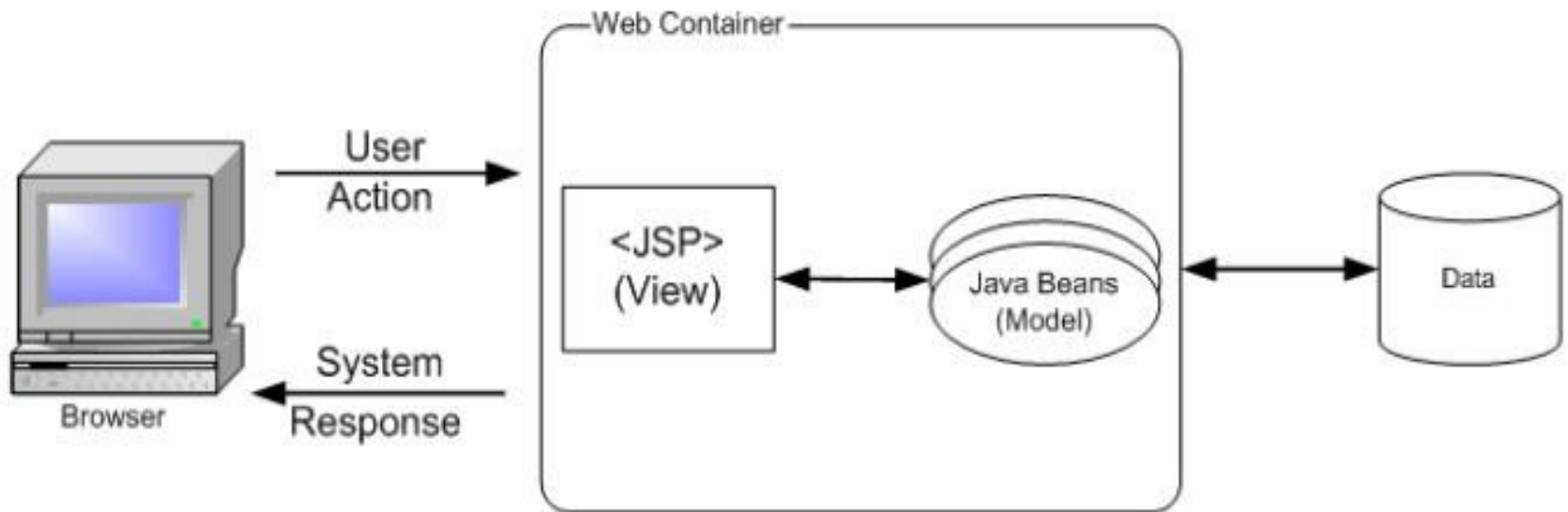


Figure 1-3. JSP Model 1 Architecture

MODEL 1

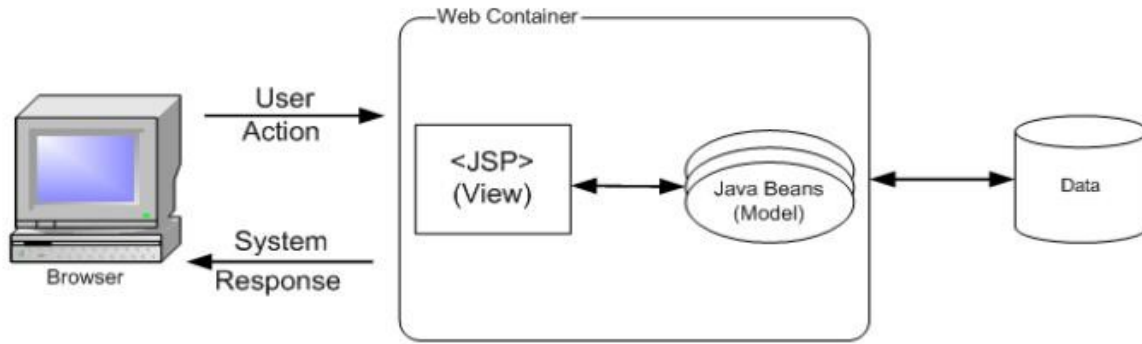


Figure 1-3. JSP Model 1 Architecture

Şekildeki gibi süreçte herhangi bir Servlet kullanılmamaktadır. İstemcinin isteği direkt olarak JSP sayfasına gönderilir. Bu JSP işlemleri yapabilmesi için gerekli olan diğer sınıfları(JavaBean,...) ve servisleri kullanır.

Bir sonraki görüntülenecek olan çıktıyı JSP sayfası kendisi ya da kullanıcıdan gelen bir parametreye göre belirler.

MODEL 2

Bu modelde kullanıcıdan gelen istek bir Servlet tarafından kesilir. Bu servlete genellikle *Kontrolcü Servlet (Controller Servlet)* denilir.

Bu servlet gelen isteği ilk olarak işler ve bir sonraki adımda hangi JSP nin görüntüleneceğini karar belirler.

MODEL 2

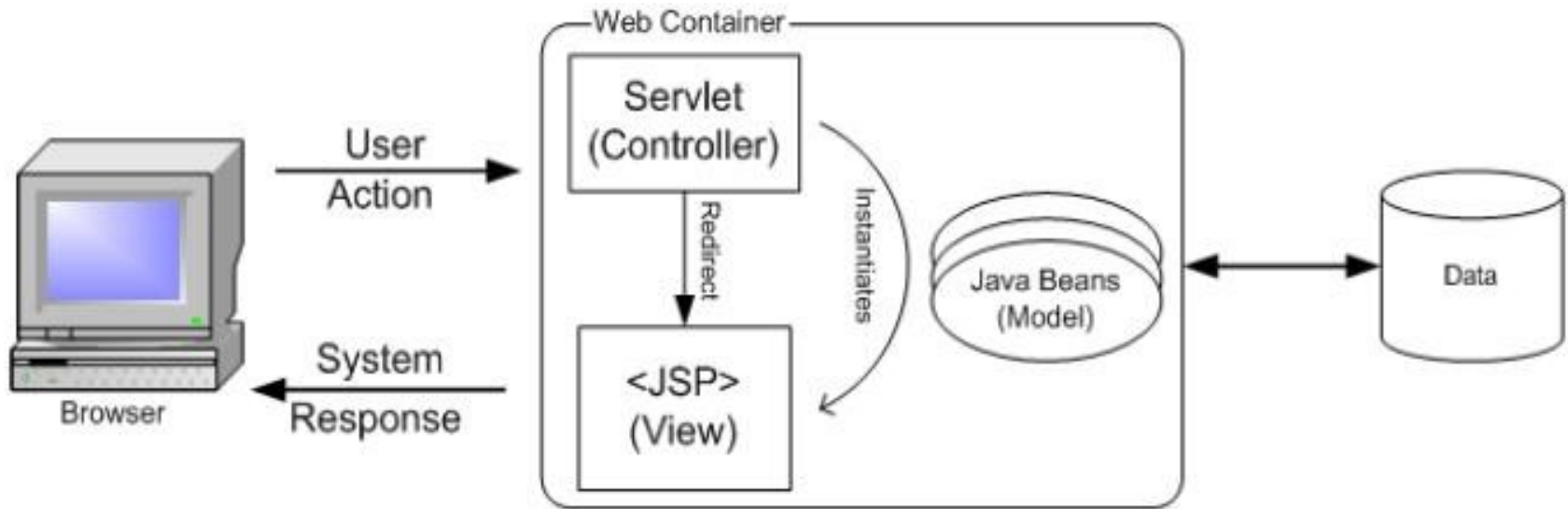


Figure 1-4. JSP Model 2 Architecture

MODEL 2

Bir istemci hiçbir zaman bir JSP ye direkt olarak istek göndermez.

Kontrolcü servlet bir trafik polisi biri görev yapar. İstekleri ilk olarak karşılar,

doğrulama (authentication),
yetkilendirme (authentication),
merkezi günlükleme (logging), ulusallaştırma
(internalization)

gibi işlemleri yerine getirebilir. İşlemler sonunda ilgili JSP ye bu isteği gönderir.

MODEL 2

Model 2 mimarisi ile

iş mantığı (business logic -gerçekten yapılması gerekli olan işler),

sunum (presentation-işlemlerin sonuçlarının ve verilerin görüntülenmesi) ve

isteklerin işlenmesi (request processing) ayrılabilir.

Bu ayırma ***Model-Görünüm-Kontrolcü Model-View-Controller (MVC)*** tasarım şablonudur.

MODEL 2

Bu şekilde web uygulamasındaki bileşenler sorumluluklarına göre ayrılırlar.

Bu şekilde uygulamanın geliştirilmesi ve bakımı daha etkin bir hale gelir.

MVC tasarım şablonunun 3 temel bileşeni:

Model: İş alanındaki bilgiyi tutar. Veri tutar.

Görünüm(View): İş alanındaki bilgileri (model) görüntüler.

Kontrolcü(Controller): Kullanıcının girdisine göre iş akışını yönetir.

MODEL 2

Bu şablonda kullanılan *Model* kısmı farklı olabilmektedir. Örneğin; veritabanında işlem yapan bir uygulamada, sonuç kümesinin (ResultSet) satırlarına karşılık gelen sıradan Java Bean ler Model olarak kullanılır.

Otomatik olarak veritabanından bu Java Bean lere dönüşüm yapan çatılar(framework) bulunmaktadır.

Gerçekleştirilen bu işleme **Nesne-İlişkisel Eşleme (Object-to-Relational Mapping : ORM)** denilmektedir.

- TopLink*
- CocoBase*
- Hibernate*
- ...

MODEL 2

Daha karmaşık uygulamalarda EJB sunucularında bulunan EJB'ler Model olarak kullanılabilmektedir.

Görünüm (View) olarak html, JSP kullanılmaktadır.

Kontrolcü(Controller): Genellikle bir JavaServlet'tir.
Görevleri:

- 1.) İstemciden gelen isteklerin yolunu keser.
- 2.) İsteği alarak gerekli işlemleri yapacak işlemlere (business operation) çevirir.
- 3.) Yapılacak işlemleri kendisi gerçekleştirir ya da bir işleyiciyi(handler) temsilci olarak seçer ve işi yaptırır.
- 4.) İstemciye gösterilecek bir sonraki görünüme karar verir.
- 5.) İstemciye bu görünümü döndürür.

MODEL 2

Tüm istek ve cevaplar kontrolcüden geçer. Bu durumda uygulamaya yeni bir özellik ekleneceğinde tüm JSP lere bu özelliğin eklenmesi yerine sadece Kontrolcüye eklenir.

Struts bir çatı(framework)dır ve **MVC** tasarım şablonunun gerçekleştirilmesini sağlayan birçok sınıf ve yapılandırma dosyası içerir.

STRUTS GÖZDEN GEÇİRME

- MVC tasarım şablonu**na uygun bir çatı(framework) dır.
- Kendine ait olan JSP etiketleri** mevcuttur. Bu etiketler uygulamada kullanılır.
- Merkezi bir yapılandırma(configuration) dosyası** bulunmaktadır. Bu dosya xml biçimlidir. Yapılandırma dosyası ile Java kodunda değişiklik yapılmadan uygulamanın çeşitli davranışları farklılaştırılabilir.
(**struts-config.xml**)
- Form Bean**'ler kullanılarak kullanıcı formlarından gelen veriler tek bir nesnenin özelliklerinde saklanır. Veriler bu nesne özelliklerine otomatik olarak aktarılır. Ayrıca bu nesnelerin gelişmiş özellikleri mevcuttur.

STRUTS GÖZDEN GEÇİRME

-**Bean etiketleri** , <jsp:useBean ...> ve <jsp:getProperty ...> etiketlerine eş, daha basit etiketleri bulunmaktadır. (<bean:write ...> gibi)

-**HTML Form** etiketleri vardır.

-**Form alanlarının geçerlilik kontrolü(validation):**

Form alanlarının istenilen biçimde girilip girilmediğinin test edilmesini sağlayan yerleşik mekanizmaları vardır.

Değerler girilmemiş ya da hatalı girilmiş ise hata mesajları görüntülenmesi yapılır. Bu geçerlilik kontrolü istemcide ya da sunucuda yapılabilir.

Örnek struts siteleri :

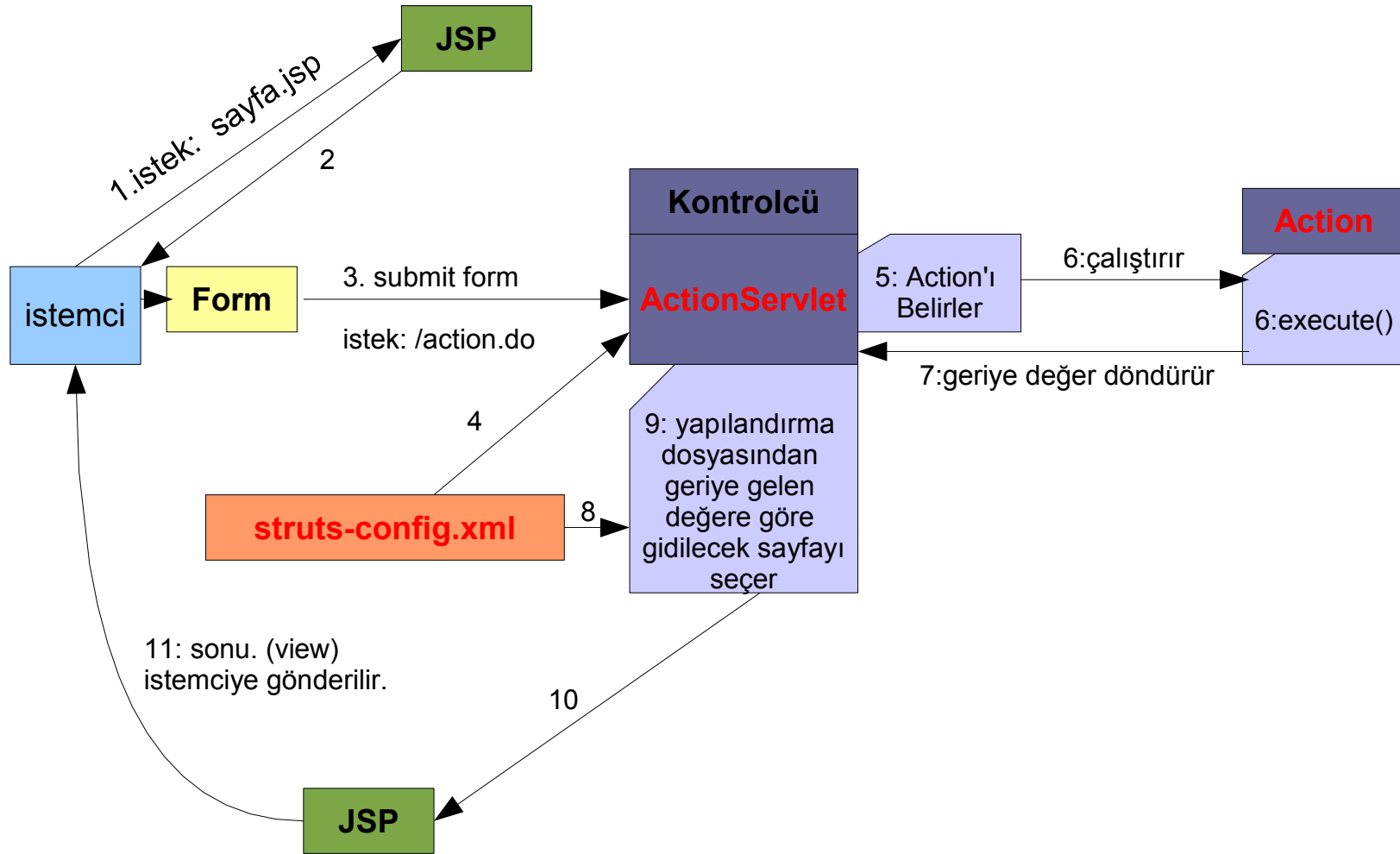
<http://www.enterprise.com>

<http://travel.travelocity.com>

<http://www.bp.com>

<http://www.mastercard.com>

STRUTS AKIŞ KONTROLÜ



STRUTS GÖZDEN GEÇİRME

ActionServlet: Kontrolcü sınıfıdır. **struts-config.xml** yapılandırma dosyasını uygulama çalışmaya başladığında okur.

Bu dosyadan, gelen isteklere karşılık hangi Action sınıfın seçileceği, kullanıcıdan gelen istek bir formdan geliyorsa bu form bilgilerinin hangi Form Bean'e konulacağı, Action sınıfından geri dönen değere göre nereye yönlendirilme yapılacağı gibi bilgileri alır.

Action: Belirli bir işi yerine getirir. ActionServlet bu nesneleri ihtiyaç halinde otomatik olarak oluşturur. Daha önceden oluşturulmuş var ise bu nesneyi kullanır. Action sınıfında bulunan execute() metodu Action sınıfın işlerinin yapıldığı metottur ve ActionServlet tarafından otomatik çalıştırılır.

STRUTS GÖZDEN GEÇİRME

ActionForm: Kullanıcıdan gelen form bilgilerinin doldurulduğu Java Bean sınıfıdır. ,

Java Bean'den farklı olarak validate(), reset() gibi metotları vardır.

validate() metodu form verileri nesne içerisindeki değişkenlere doldurulduktan sonra otomatik olarak ActionServlet tarafından çalıştırılır.

Bu metotta oluşan hatalara göre farklı işlemler yapılır.

STRUTS GÖZDEN GEÇİRME

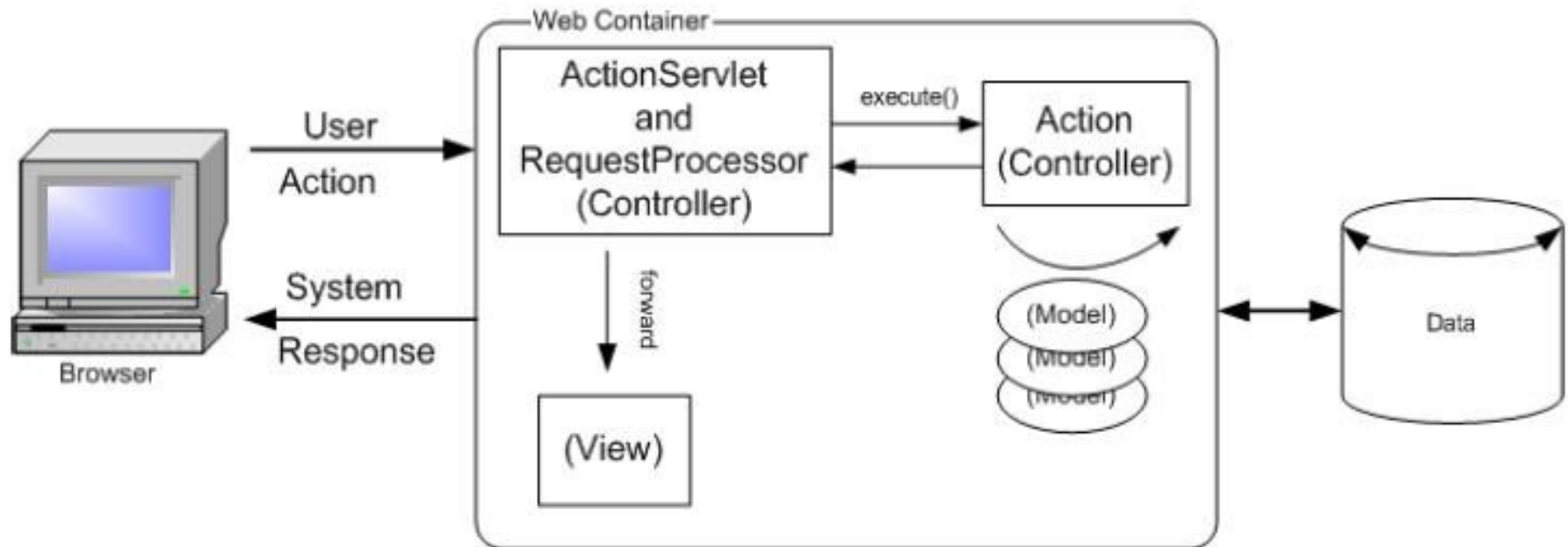


Figure 3-6. The execute() method of the Action class is called by the primary controller

STRUTS ÇALIŞMA ADIMLARI

- 1.) Kullanıcı bir form isteğinde bulunur. html formu ya da `<html:form>` struts etiketi ile oluşturulmuş bir form dur.
- 2.) Bu form `action_adı.do` şeklinde bir adrese gider. Bu adres değeri `struts-config.xml` dosyası içerisindeki bir Action sınıfı ile eşleştirilmiştir.
- 3.) Bu ayar dosyasından tespit edilen Action sınıfı ile ilişkili bir `ActionForm` olup olmadığına bakılır.
- 4.) İlişkili bir `ActionForm` var ise, belirtilen scope da bu nesneden var mı diye kontrol edilir. Var ise kullanılır yok ise yeni bir nesne oluşturularak bu scope a eklenir.

STRUTS ÇALIŞMA ADIMLARI

- 5.) ActionForm nesnesinin reset() metodu çalıştırılır.
- 6.) Kullanıcı formundan gelen bilgiler bu form nesnesinin elemanlarına doldurulur.
- 7.) ActionForm nesnesinin validate() metodu çalıştırılır.
- 8.) validate() metodundaki hata durumuna göre forma geri dönülür ve hata mesajları gösterilir ya da bir sonraki adıma geçilir.
- 9) Tespit edilen Action sınıfına ait olan **yönlendirme (forward)** bilgileri **ActionMapping** nesnesine depolanır.
Bu nesnede **forward adı=gidilecek yer** şeklinde kayıtlar bulunur.

STRUTS ÇALIŞMA ADIMLARI

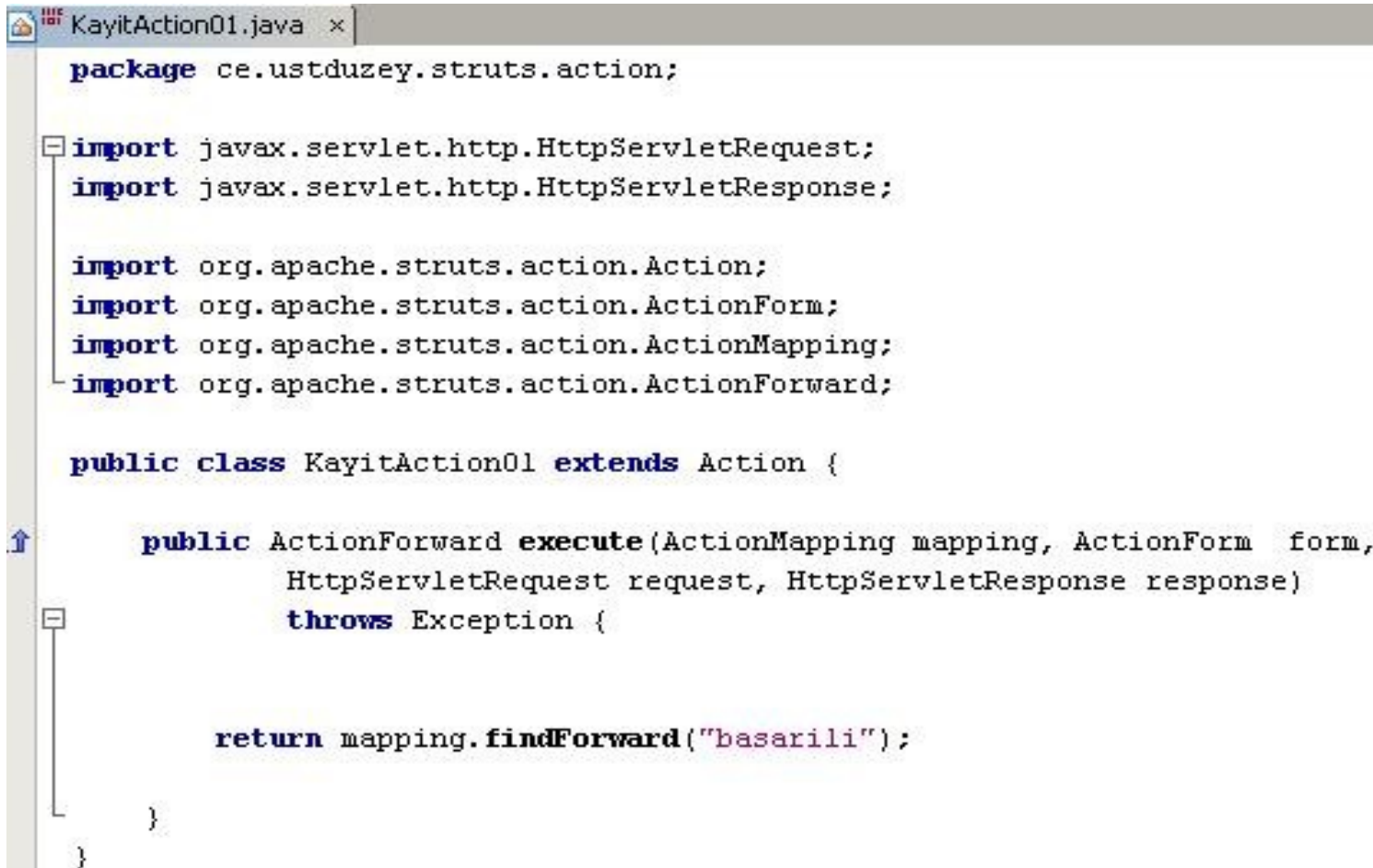
10.) ActionServlet, seçmiş olduğu Action sınıfının execute() metodunu ActionMapping, ActionForm nesneleri ile birlikte request ve response nesnelerini de parametre olarak göndererek çalıştırır.

11.) Action metodu çalışması sonucunda ActionServlet' e bir yönlendirme (forward) bilgisi gönderilir.

12.) ActionServlet belirtilen yönlendirme bilgisine göre yönlendirme işlemini gerçekleştirir ve sonuç görünüm(view) kullanıcıya gönderilir.

Örnek

```
kayit1.jsp x
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Yeni Hesap Kayıt Sayfası</title>
  </head>
  <body>
    <h1>Yeni Hesap Kayıtı</h1>
    <form action="kayit1.do" method="POST">
      <table>
        <tr>
          <td>Email Adresi :</td>
          <td><input type="text" name="email"></td>
        </tr>
        <tr>
          <td>Parola:</td>
          <td><input type="password" name="parola"></td>
        </tr>
        <tr>
          <td colspan="2"><input type="submit" value="KAYIT OL"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```



```
package ce.ustduzey.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionForward;

public class KayitAction01 extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {

        return mapping.findForward("basarili");
    }
}
```

onay.jsp x

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>ONAY JSP</title>
</head>
<body>
    <h1>Başarılı Şekilde Kayıt Oldunuz</h1>
</body>
</html>
```



```
struts-config.xml x
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
  <global-exceptions>
  <global-forwards>
  <action-mappings>
    <action
      path="/kayit1"
      type="ce.ustduzey.struts.action.KayitAction01">
      <forward name="basarili" path="/sonuclar/onay.jsp" />
    </action>
    <action
      path="/kayit2"
      type="ce.ustduzey.struts.action.KayitAction2">
      <forward name="basarili" path="/sonuclar/onay.jsp" />
      <forward name="hatali_parola" path="/sonuclar/hatali_parola.jsp" />
      <forward name="hatali_email" path="/sonuclar/hatali_email.jsp" />
    </action>

    <action path="/Welcome" forward="/welcomeStruts.jsp"/>
  </action-mappings>
</struts-config>
```

Her action için bilgiler ayrı ayrı tutulur.

Bu action a ait yönlendirme bilgisi

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Yeni Hesap Kayıt Sayfası</title>
  </head>
  <body>

    <h1>Yeni Hesap Kayıtı</h1>
    <form action="kayit2.do" method="POST">
      <table>
        <tr>
          <td>Email Adresi :</td>
          <td><input type="text" name="email"></td>
        </tr>
        <tr>
          <td>Parola:</td>
          <td><input type="password" name="parola"></td>
        </tr>
        <tr>
          <td colspan="2"><input type="submit" value="KAYIT OL"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

```
package ce.ustduzey.struts.action;
```

```
+ import ...
```

```
public class KayitAction2 extends Action {
```

```
    public ActionForward execute(ActionMapping mapping, ActionForm form,  
        HttpServletRequest request, HttpServletResponse response)  
        throws Exception {
```

```
        String email = request.getParameter("email");  
        String parola = request.getParameter("parola");
```

```
        if ((email==null) || (email.trim().length()<3) ||  
            (email.indexOf("@")==-1)){  
            return mapping.findForward("hatali_email");  
        }else if((parola==null) || (parola.trim().length()<6)){  
            return mapping.findForward("hatali_parola");  
        }
```

```
        return mapping.findForward("basarili");
```

```
    }
```

```
}
```

```
hatali_email.jsp x|
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Geçersiz Email</title>
  </head>
  <body>
    <h1>Geçersiz email adresi</h1>
    girilen adres email@email biçimine uymalıdır
    Lütfen tekrar deneyin <a href="kayit2.jsp">KAYIT</a>
  </body>
</html>
```

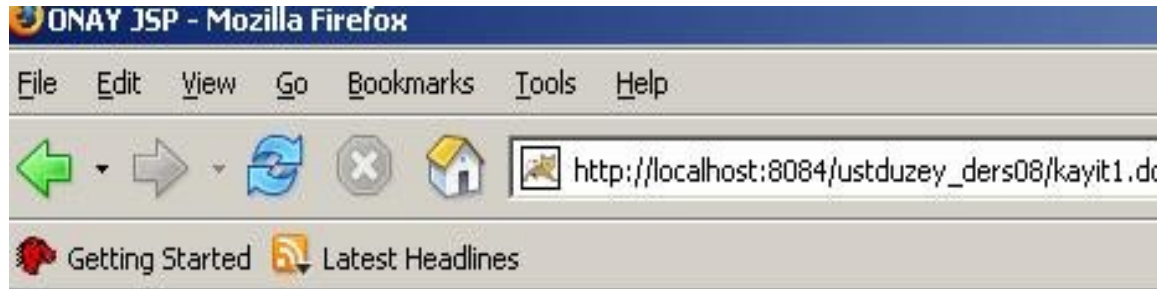
```
hatali_parola.jsp * x|
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
|
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1>Geçersiz parola</h1>
    Girilen parola en az 6 karakter olmalıdır.
    Lütfen tekrar deneyin <a href="kayit2.jsp">KAYIT</a>
  </body>
</html>
```



Yeni Hesap Kayıtı

Email Adresi :

Parola:



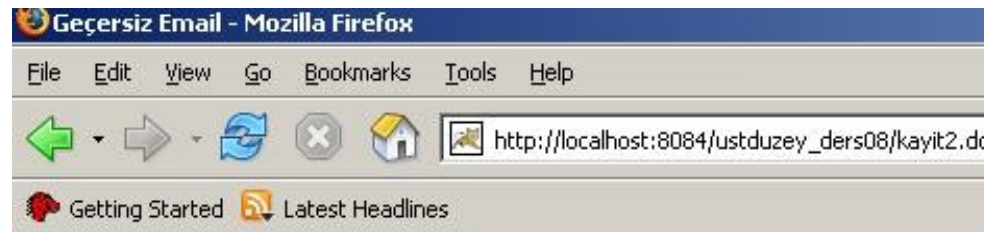
Başarılı Şekilde Kayıt Oldunuz



Yeni Hesap Kayıtı

Email Adresi :

Parola:



Geçersiz email adresi

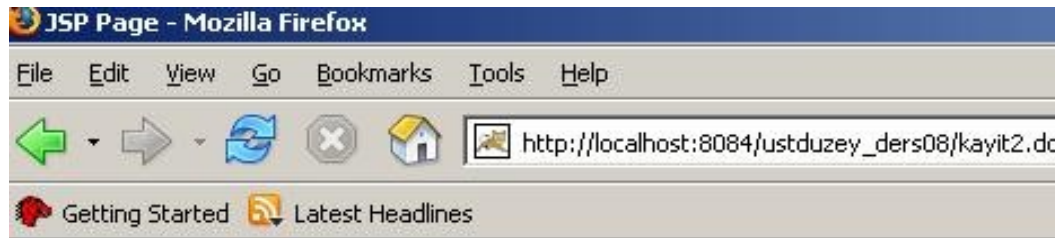
girilen adres email@email biçimine uymalıdır Lütfen tekrar deneyin [KAYIT](#)



Yeni Hesap Kayıtı

Email Adresi :

Parola:



Geçersiz parola

Girilen parola en az 6 karakter olmalıdır. Lütfen tekrar deneyin [KAYIT](#)

Üst Düzey Programlama

Struts Framework