Kurumsal Java.com

Java Enterprise Architecture

Web Framework Gökyüzünde

Yeni bir Yıldız!

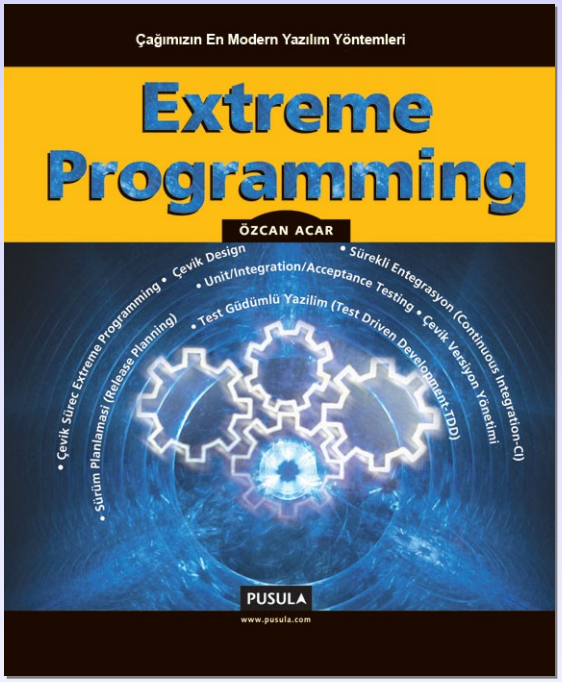
Özcan Acar
acar@unitedinter.net
http://www.ozcanacar.com
http://www.kurumsaljava.com

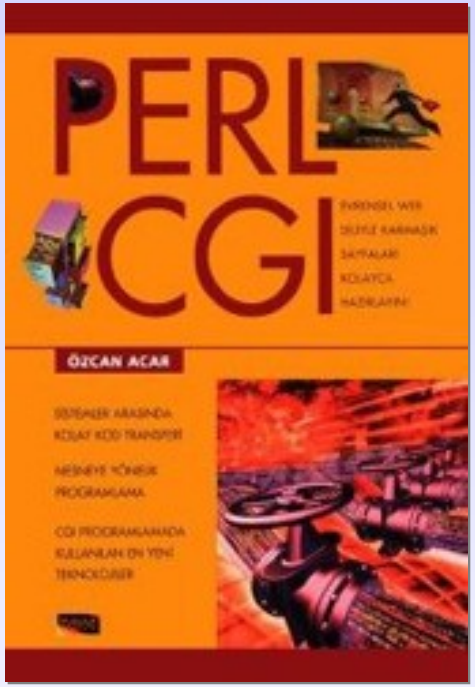
Wicket'in Varoluş Nedeni

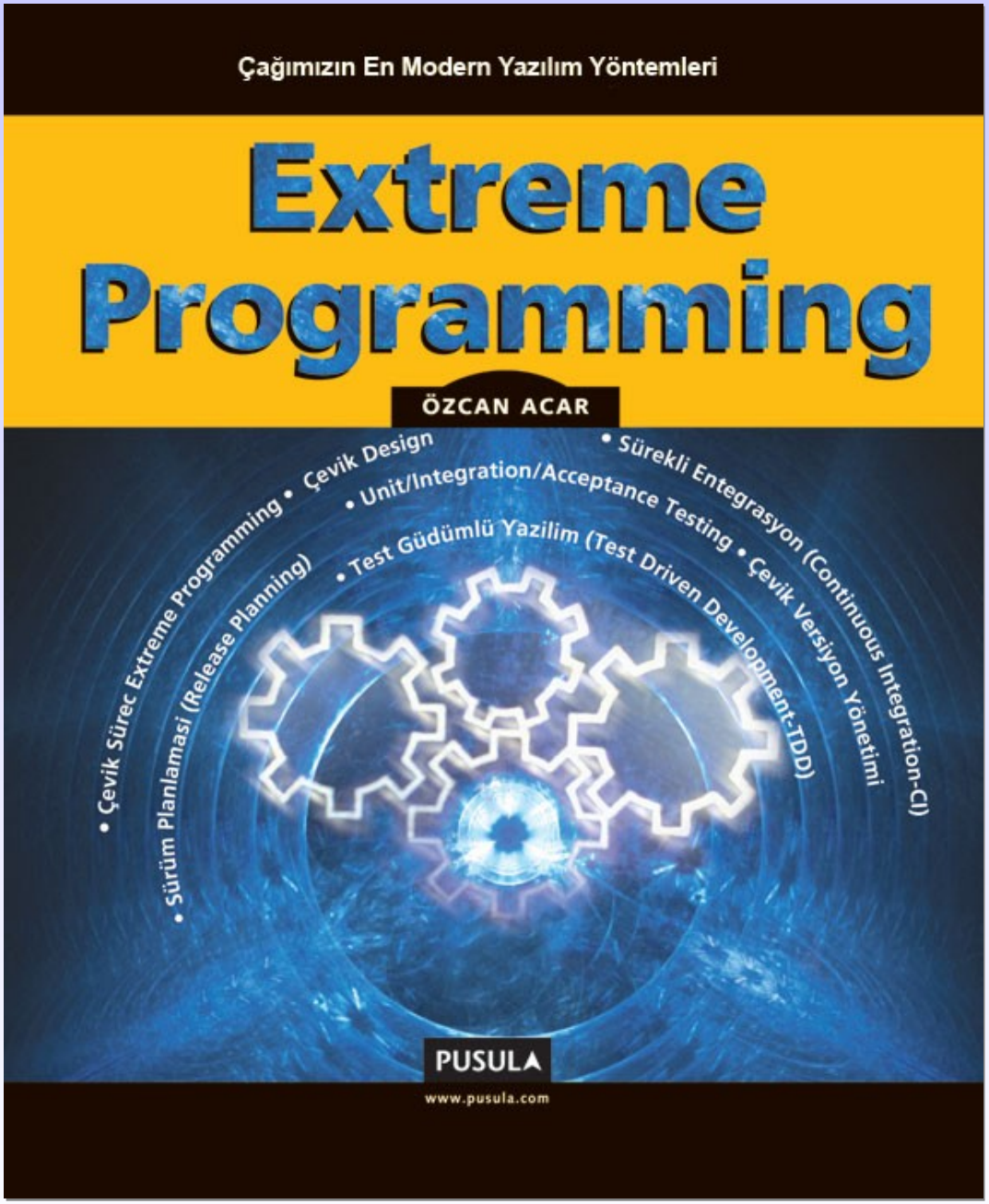
Özcan Acar Hakkında

```
public class OezcanAcar
{
      public static void main(String[] args)
      {
            Acar oezcan = new Acar();
            oezcan.setBirthday("18.07.1974");
            oezcan.setBirthplace("Izmir");
            oezcan.setJob("Bilgisayar Mühendisi");
           oezcan.setPassion("J2EE");
      }
}
```

Özcan Acar Hakkında







Sunumun İçeriği

- API anlatmaya gelmedim! Wicket'i anlamak için web programcılığında gerekli soyut konseptler üzerine yoğunlaşacağız.
- Bir tezim var: 2 boyutlu web programcılığı yapıyoruz. Neden?
- Java ile web programcılığının tarihçesi
- Wicket tanıtımı
- Wicket ile Web Komponentleri

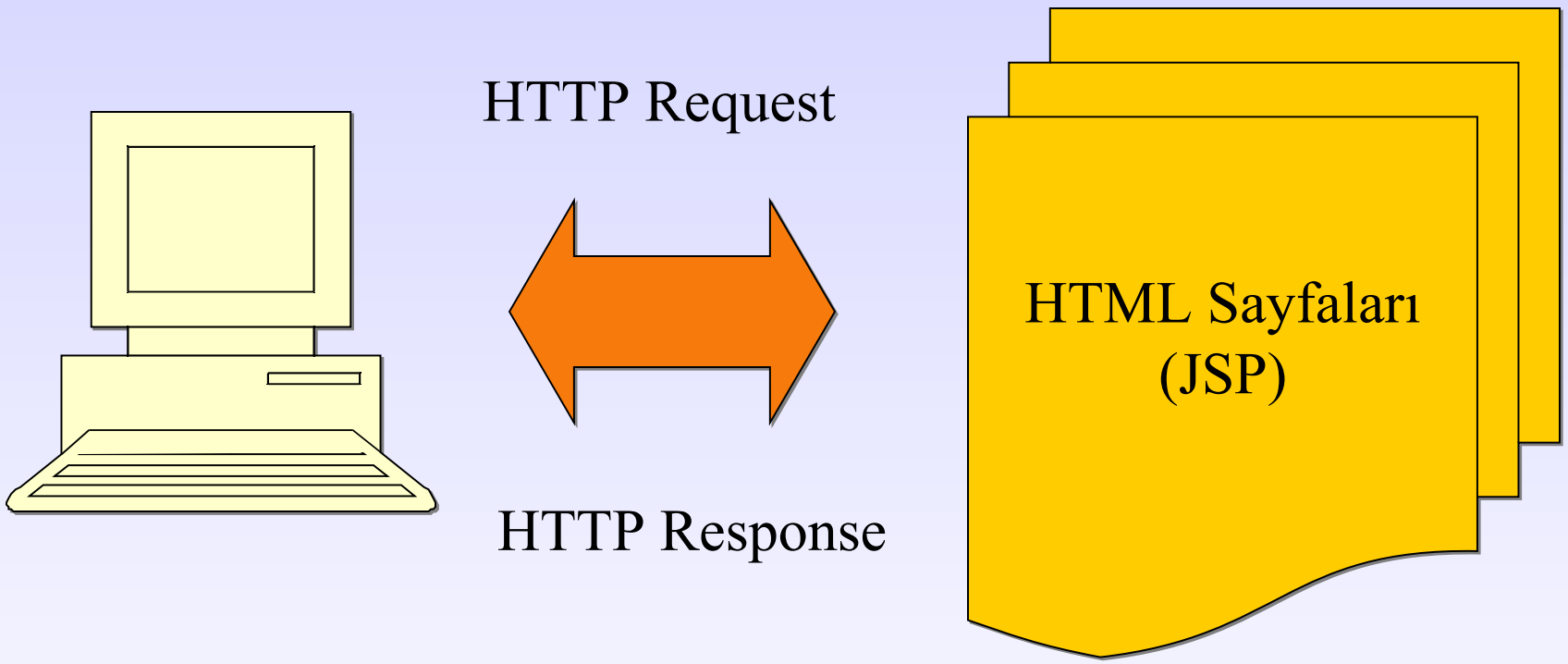
Giriş

Kimler daha önce Java teknolojilerini kullanarak web programcılığı yaptı?

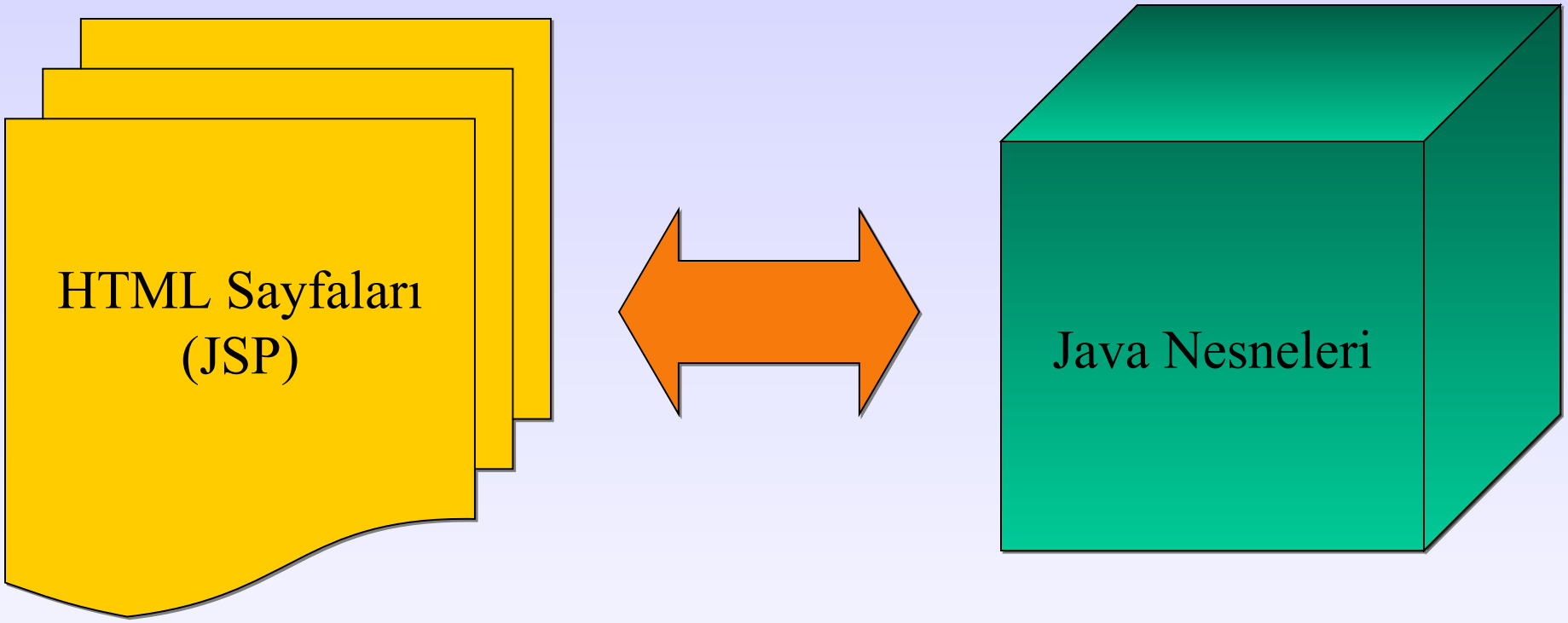
Web Programcılığı

- Web programcılıgı klasik maasaüstü programcılığından farklıdır.
- Yazılan kod dinamik olarak web tarayıcısı için gerekli HTML kodunu oluşturur.
- Web aplikasyonu için yazılan kodun aynı anda birden fazla kullanıcıyı desteklemesi lazım (Multithreading).
- Web aplikasyonun değişik türdeki bilgisayar, işletim sistemi ve web tarayıcısını desteklemesi gerekir.

Web Aplikasyonu



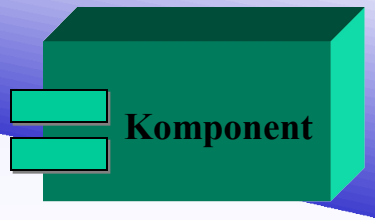
Java Web Aplikasyonu



Tez: 2 boyutlu web yazılımı yapıyoruz! 3. boyut neresi?

2 boyutlu
web programcılığı
Controller,
Model, View,
JSTL

3 boyutlu
web programcılığı
Gerçek Java
Komponentleri



Beklentilerimiz

- Web programcılığı da olsa nesneye yönelik programcılığı uygulamak istiyoruz.
- Sadece Java dilinde yazılım yapmak istiyoruz. JSTL ya da EL gibi web'e özel bir yazılım tekniğini öğrenmek zorunda kalmak istemiyoruz.
- Yazılımcı olarak amacımız bakımı kolay ve her yeni müşteri gereksinimi ile kolayca genişletilebilir yazılım sistemleri geliştirmektir. Kullandığımız araçlar bunu desteklemeli.
- Kodun tekrar kullanılabilmesi (reuse) önemlidir. Web programcılığında, tekrar kullanılabilen Java komponentleri oluşturmak zor bir iş haline gelmemeli.

Time To Change

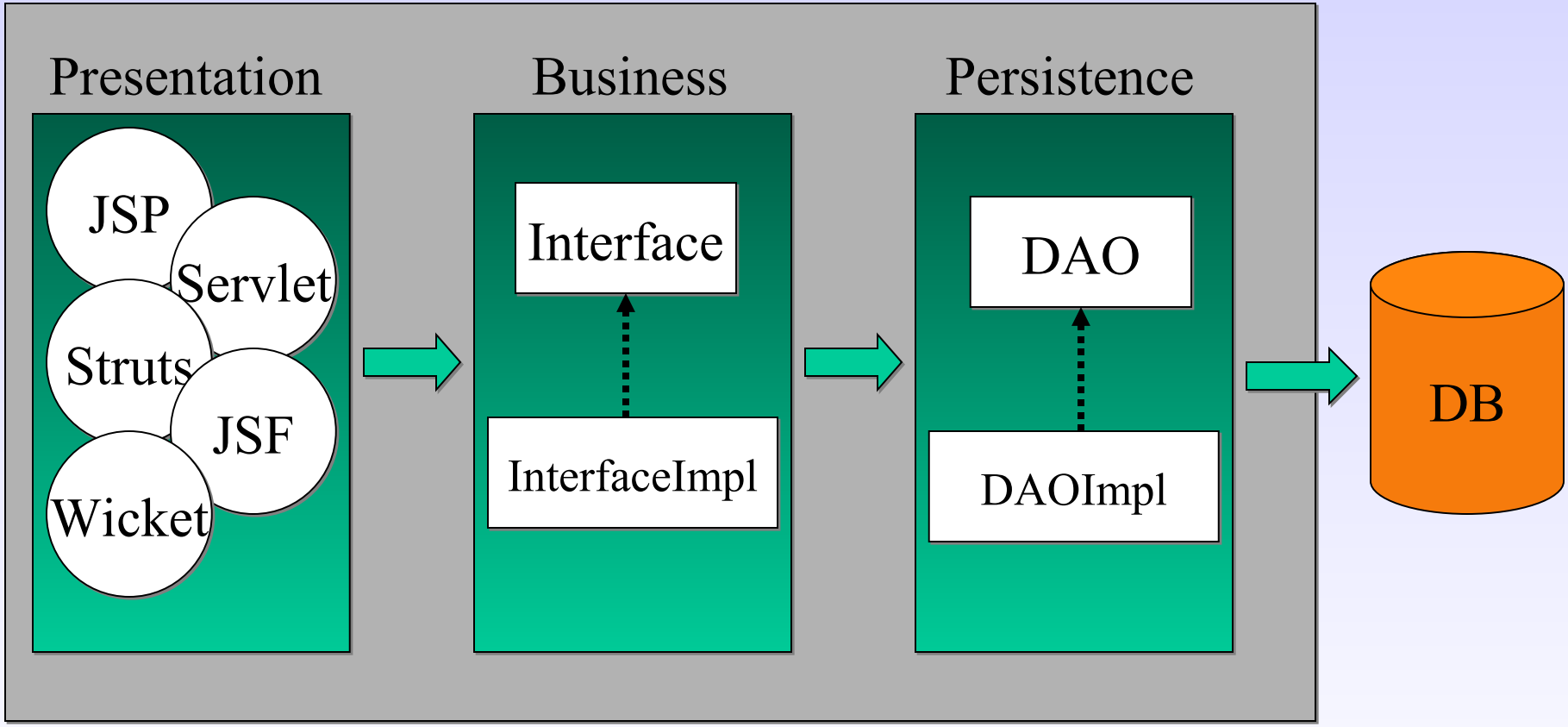
- Java ile web programcılığı hakkında bildiklerinizi unutun!
- Wicket ile web programcılığına 3. boyut geliyor!
- Wicket web komponentleri

Konsept

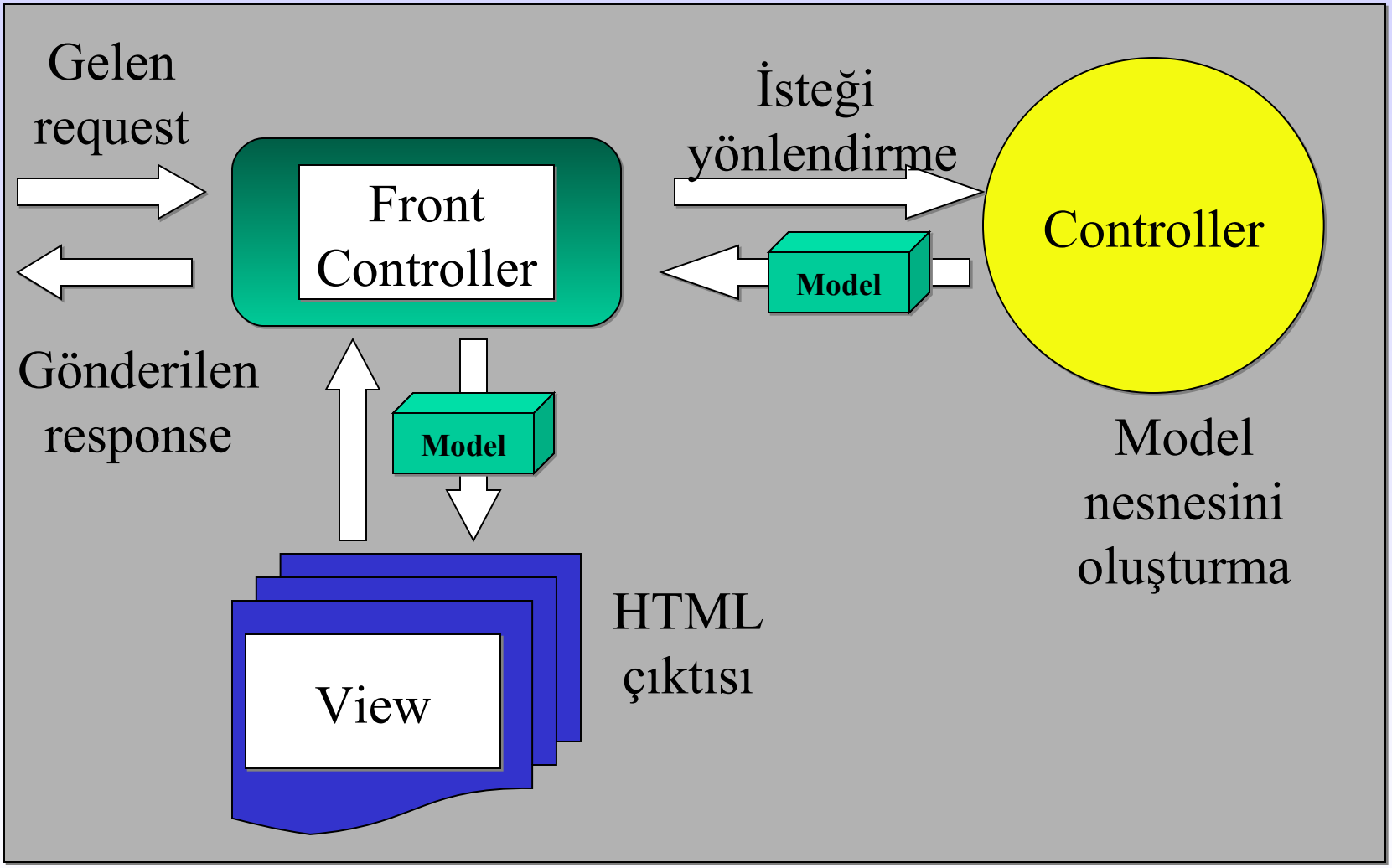
- İdeal bir web projesi mimarisi nasıl olmalı?
- MVC Design Pattern
- Komponent nedir?
- Web komponent nedir?

İdeal Bir Web Proje Mimarisi

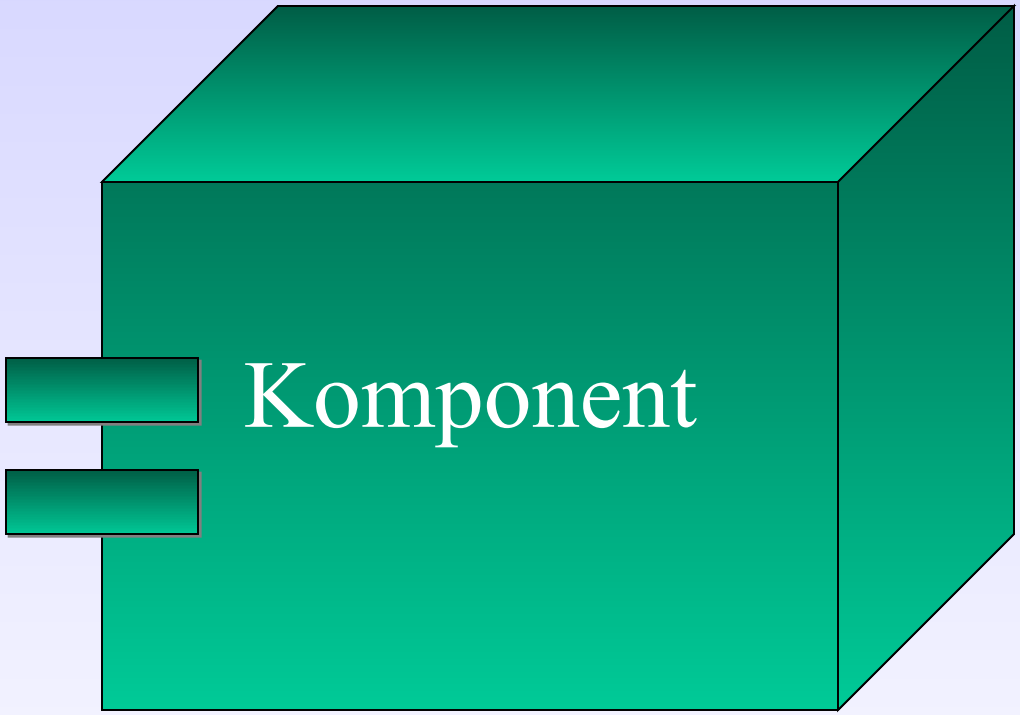
Üç Katmanlı Mimari



MVC Design Pattern

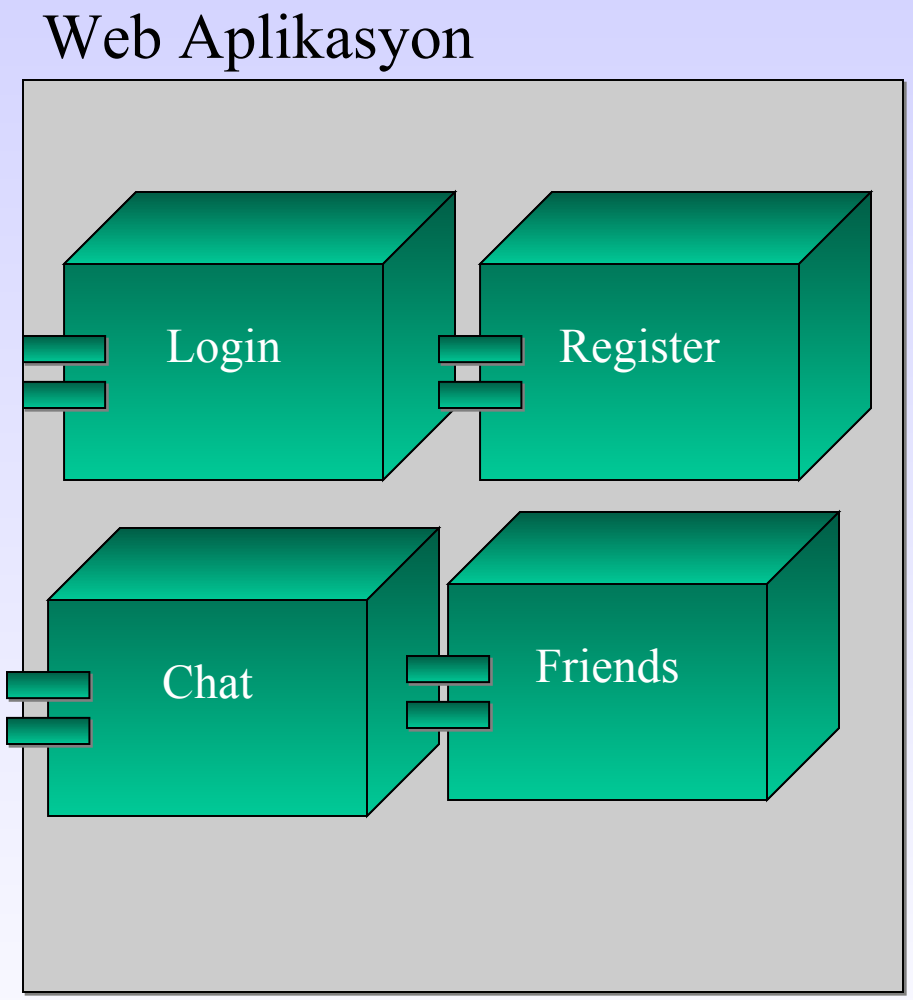


Komponent



- Tekrar kullanılabilir kod birimi.
- İç dünyasını gizler.
- Tanımlanmış interface sınıflar üzerinden kullanılır. Örneğin Facade.
- Jar dosyasında dağıtılabilir.
- Kullanıldığı ortama göre konfigüre edilebilir.

Web Komponent



- Bir web aplikasyonda konfigürasyon yapılarak kullanılan modül.
- Diğer komponent özelliklerine sahiptir.
- Jar dosyası olarak aplikasyona dahil edilebilir.

Java İle Web Programcılığın Tarihçesi

- Servlets
- JSP
- JSTL
- MVC Web Frameworks (Struts, Spring MVC)
- User Interface Component Model Frameworks (JSF)

Servlets

- Web programcılığı yapılabilen ilk Java teknolojisi.
- Sun tarafından tanımlanmış bir standart (API). İhtiva ettiği interface'ler Tomcat gibi bir application server tarafından implemente edilir.
- Servlet web tarayıcısı üzerinden gelen kullanıcı istediğini cevaplayıp, HTML sayfası oluşturan bir Java nesnedir.
- Aplikasyon serveri (Tomcat, Resin) bünyesinde çalışır. Aplikasyon server HTTP protokolü ile Servlet sınıfı arasındaki köprüdür.
- ***javax.servlet.http.HttpServlet sınıfını genişleten her sınıf Servlet haline gelir.***
- ***Servlet web aplikasyonları URL tabanlıdır (http://domain.com /servlet/SomeServlet). web.xml konfigürasyon dosyasında hangi URL den hangi Servlet sınıfının sorumlu olduğu tanımlanır.***

Servlet Multithreading

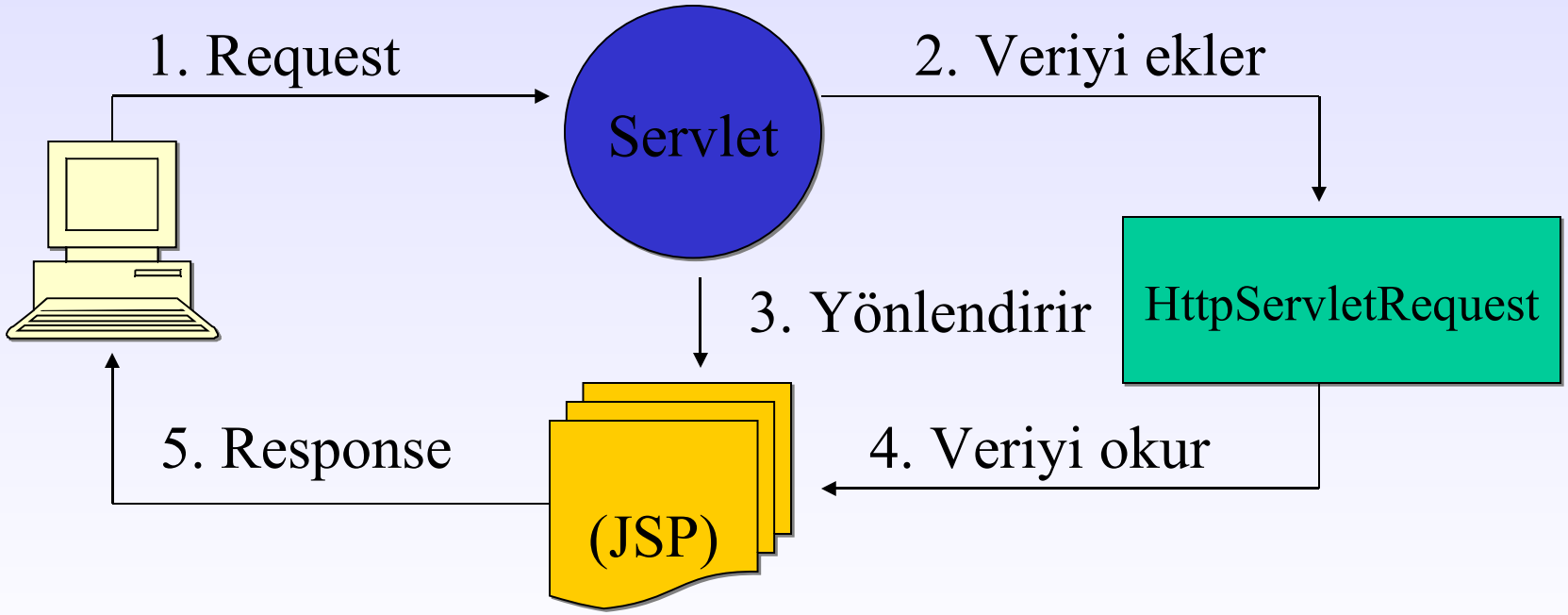
```
public class HelloWorldServlet extends HttpServlet
{
    private int counter = 0;
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
                       throws ServletException, IOException
    {
        counter ++;
        response.setContentType("text/html");
        ServletOutputStream output = response.getOutputStream();
        output.print("<h1>Sayac: " + counter +"</h1>");
    }
}
```

Servlet Teknolojisinin Dezavantajları

- HTML kodu Servlet sınıfında oluşturulduğundan, Servlet sınıflarının bakımı ve geliştirilmesi zordur!
- Geniş kapsamlı Servlet aplikasyonları çok kompleks bir yapıya dönüşebilir. Programcıların, aplikasyonun her bölümünü aynı seviyede anlamaları zorlaşabilir.
- Her yeni müşteri gereksiniminin getirdiği değişiklikler, bakım ve geliştirme sürecini zora sokabilir.
- HTML kodu programcılar tarafından geliştirilmek zorundadır. Web design yapan ekibin Java bilgisi olmadan Servlet sınıfları üzerinde modifikasyon yapmaları mümkün değildir.
- Servlet aplikasyonlarında HTML kodu da üretildiği için kodun bol olduğu sınıflar oluşur.

JSP – Java Server Pages

- Servletler tarafından template olarak kullanılmak üzere geliştirilmiş web teknolojisi. Bir standart.
- Application server JSP sayfalarını compile esnasında Servlet sınıflarına dönüştürür.



JSP – Java Server Pages

```
<html>
<body>
<%
    for(int i=0; i<5; i++)
    {
        out.print("<h1>Merhaba Dünya</h2>);
    }
%>
</body>
</html>
```

JSP Teknolojisinin Dezavantajları

- JSP Servlet teknolojisi ile beraber kullanıldığında JSP ve Servlet arasında nesnesel bir ilişki yoktur. Oluşturulan HTML kodunda yer alan linkler üzerinden ihtiyaç duyulan Servlet sınıfları adreslenir. Servlet bir JSP nesne referansına erişemediği için ***HttpServletRequest*** üzerinden JSP için gerekli verileri oluşturur.
- Kompleks JSP sayfalarında Java kodunun kullanılması, JSP sayfalarının bakımını güçleştirir.
- JSP sayfalarnda değişiklik yapmak isteyen web design ekibinin Java biliyor olması gerekir. Web designcıların yaptıkları en ufak bir değişiklik bile web aplikasyonu çalışmaz hale getirebilir.

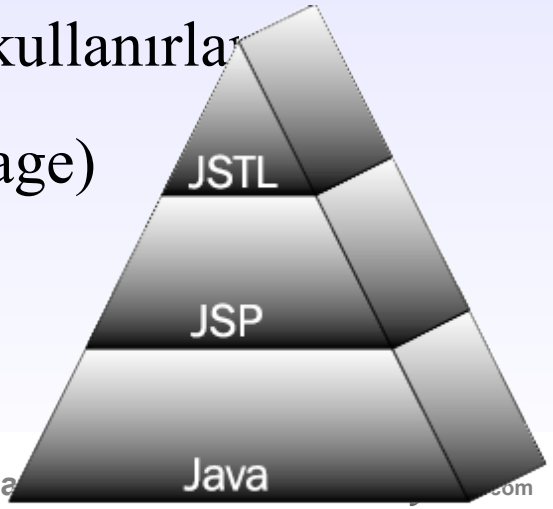
JSP Teknolojisinin Dezavantajları

- Java kodu kullanılan JSP sayfalarının HTML editörlerinde gösterim problemi oluşmaktadır.
- JSP sayfalarında Java kodunu kullanılması hata tespit ve debugging işlemlerini zorlaştırmaktadır.
- Test güdümlü (Test Driven Development) JSP yazılımı bir application server (Tomcat) kullanmadan mümkün değildir (HttpUnit).

JSTL – Java Standard Tag Library

- Yeni bir web komponent modelini ihtiva eder.
- JSP sayfasında Java kodu yerine, tag olarak bilinen ve server tarafında bir Java sınıfında yer alan koda denk gelen birimler kullanılır. Tagler komponenttir.
- Web design ekibi Java ögrenmek zorunda kalmadan JSP sayfaları oluşturabilir.
- Tagler XML syntax benzeri bir notasyon ile kullanırla
- Verilere ulaşmak için EL (Expression Language) kullanılabilir.

```
<c:out value="${username}"/>
```



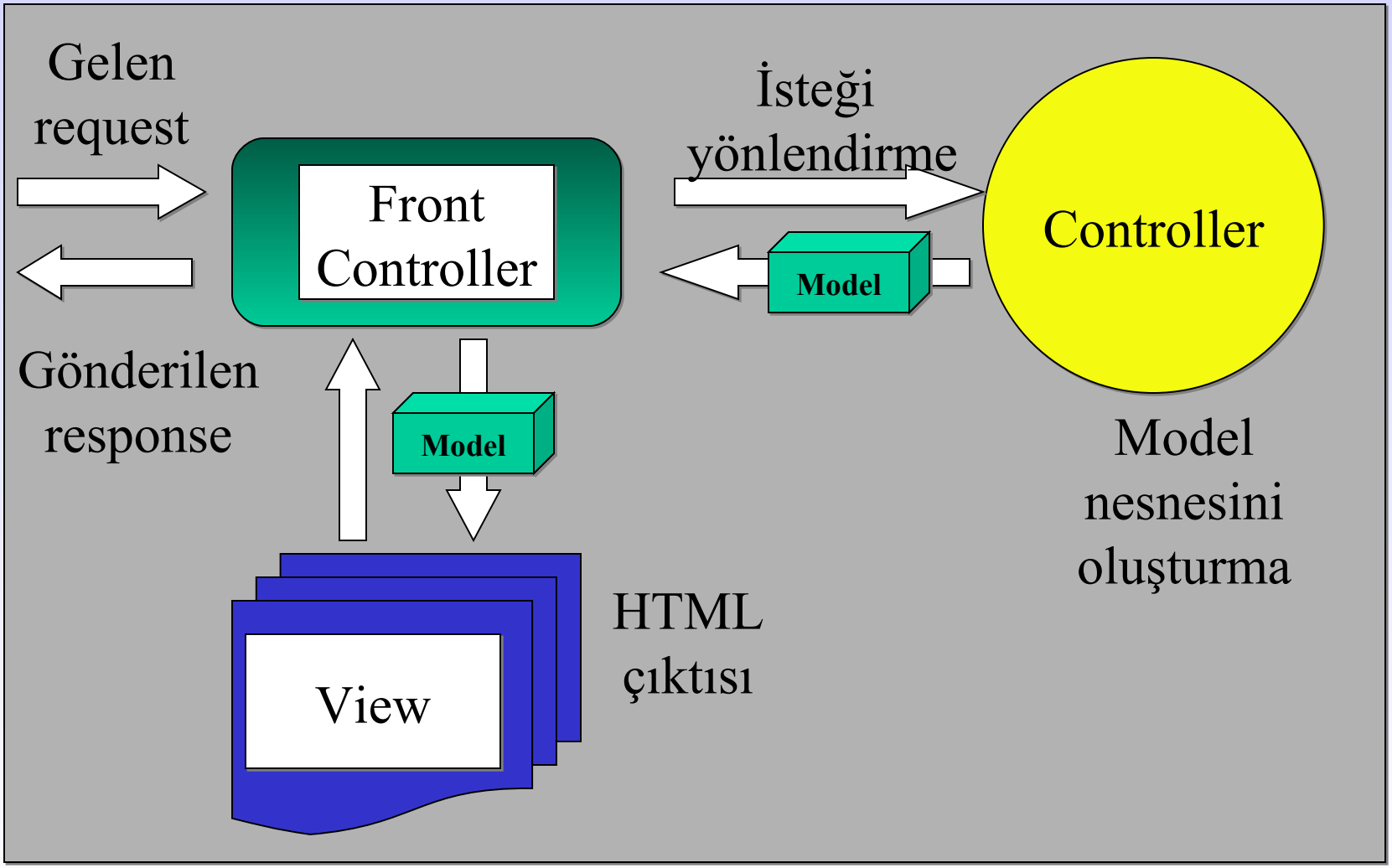
JSTL – Java Standard Tag Library

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<p><h1>Müsteri Isimleri</h1></p>
<c:forEach items="${addresses}" var="address">
   <c:choose>
      <c:when test="${not empty address.lastName}" >
        <c:out value="${address.lastName}"/><br/>
      </c:when>
   <c:otherwise>
      N/A<br/>
  </c:otherwise>
</c:choose>
```

JSTL – Java Standard Tag Library

- JSTL kullanılmış olsa bile JSP teknolojisi ile geniş kapsamlı web programları oluşturmak kolay değildir. JSP sayfalarında HTML gösterimi yanı sıra navigasyon ve verilerin validasyonu (validation) gibi kompleks işlemlerin yapılması gerekmektedir. Bu işlemlerin JSP sayfalarında yapılması, JSP sayfalarının bakımını ve geliştirilmesini zora sokmaktadır. Navigasyon ve validasyon gibi işlemlerin JSP sayfalarının dışında, başka bir mekanizma kullanılarak yapılması gerekmektedir. Bu ihtiyacı karşılamak amacıyla Struts, Spring MVC ve WebWork gibi web frameworkler doğmuştur.

MVC Frameworks



MVC Frameworks

Struts

webwork

tapestry

Spring 2.0

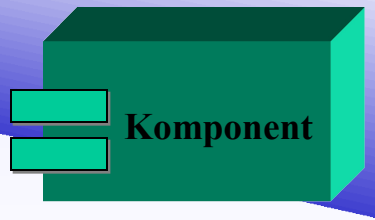
User Interface Component Model Frameworks

- JSF (Java Server Faces), Tapestry gibi web frameworkleri komponent modeline sahiptir.
- HTML arayüzü oluşturmak için Java komponentleri kullanılır.
- Komponentleri HTML sayfalarında kullanabilmek için JSTL, EL tarzı programlama teknikleri kullanılır.

Tez: İki boyutlu web yazılımı yapıyoruz! 3. boyut neresi?

2 boyutlu
web programcılığı
Controller,
Model, View,
JSTL

3 boyutlu
web programcılığı
Gerçek Java
Komponentleri



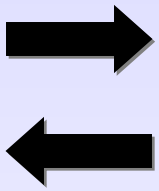
Neden 2. boyutta kalındı?

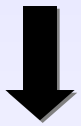
JSP Sayfası

```
<%@ taglib prefix= “c“
      uri=..../>
```

Controller Sınıfı

```
public class Controller
{
}
```





XML Konfigürasyon

```
<config>
...
</config>
```

Model Sınıfı

```
public class Model
{
}
```

Neden 2. boyutta kalındı?

- Mevcut web frameworklerde gerçek anlamda Java komponent oluşturulması ve kullanımı mümkün değil.
- Bize komponent modeli olarak satılan bir takım JSTL tag kütüphaneleri ve JSF den tanıdığımız user interface komponentleri
- Tekrar kullanılanabilir web komponentleri oluşturmak mümkün değil.

Kısaca Wicket

- Açık Kaynaklı (Open Source) web framework
- Komponent tabanlı (Component Based)
- HTML Markup kodunu ve Java kodunu ayırır.
- JSTL ya da EL (Expression Language) gibi bir programlama tekniğini gereksiz kılar. Basit Java (POJO – Plain Old Java Object) sınıfları kullanır.
- Çok az konfigürasyonu gerekli kılar. XML konfigürasyon dosyası yoktur.
- Java ve HTML'i bilenler kısa zamanda Wicket'i ögrenebilir.
- Spring ile birebir entegre edilebilir.

Kısaca Wicket

- WicketTester sınıfı ile test güdümlü yazılımı ve akseptans (acceptance – onay/kabul) testlerinin oluşturulmasını destekler.
- Javascript kodu yazmadan Ajax desteği sağlar.

Wicket Konseptleri

- **Application**
- Page
- Komponentler
- Modeller
- Paneller

Application

- Web aplikasyonun başlangıç noktasıdır (Main Entry Point).
- Giriş sayfasını (Home Page) tanımlar.
- Spring ile entegrasyonun gerçekleştiği (`SpringComponentInjector`) yerdir.

Application

web.xml konfigürasyonu

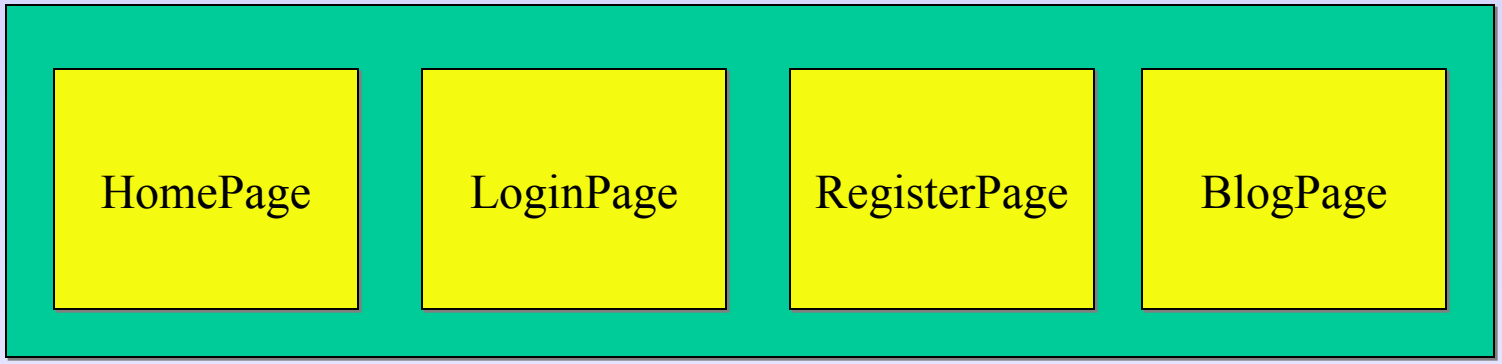
```
<filter>
    <filter-name>wicket</servlet-name>
    <filter-class>
        org.apache.wicket.protocol.http.WicketFilter
    </filter-class>
    <init-param>
        <param-name>applicationClassName</param-name>
        <param-value>
            com.company.MyApplication
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</filter>
```

Wicket Konseptleri

- Application
- **Page**
- Komponentler
- Modeller
- Paneller

Page

Application




```
public abstract class HomePage extends WebPage
{
    public SmartWicketPage()
    {
        super();
        buildGui();
    }

    public abstract void buildGui();
}
```

Wicket Konseptleri

- Application
- Page
- **Komponentler**
- Modeller
- Paneller

Komponentler

- Label
- MultiLineLabel
- Button
- SubmitLink
- TextField
- CheckBox
- TextArea
- ListChoise
- RadioChoise
- Image
- Link
- Tree
- Panel
- Loop
- Border
- Fragment
- PageLink
- ExternalLink
- Palette
- CheckGroup
- DropDownChoise
- RadioGroup
- Select
- ImageMap

Komponentler ve Markup

HTML sayfalarında Wicket komponentleri ***wicket:id*** ile kullanılır.

HTML

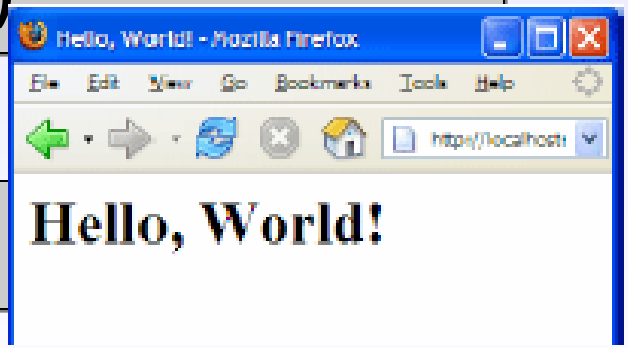
```
<h1 wicket:id=  msg>
        Komponent tarafından değiştirilir
</h1>
```

Java Kodu

```
new Label(  msg, "Hello, World!");
```

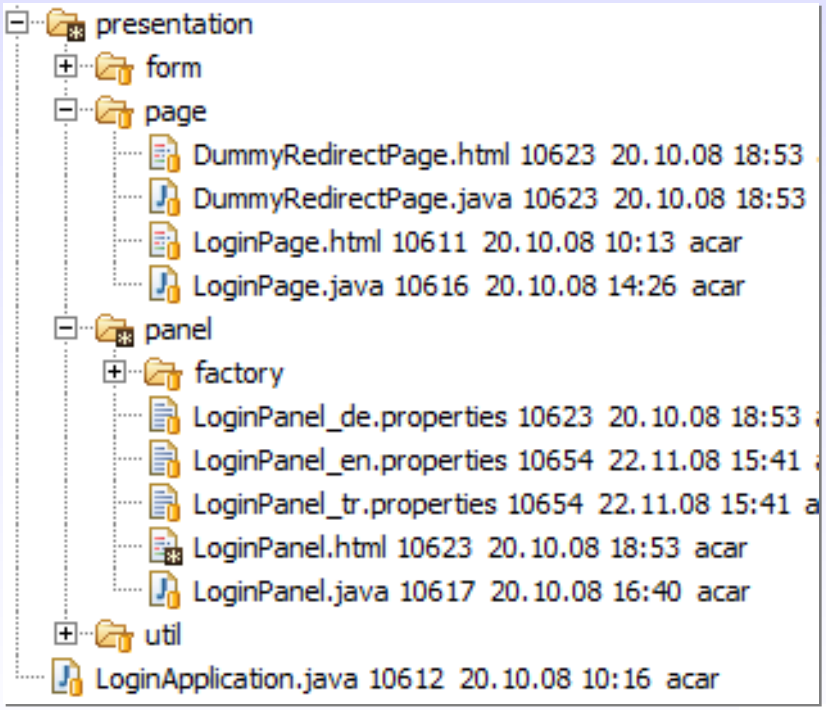
Ekran Çıktısı

```
<h1>Hello, World!</h1>
```



Komponentler ve Markup

- Page ve Panel gibi komponentlerin kendi HTML dosyaları vardır.
- Komponentler ve sahip oldukları HTML sayfaları aynı Java package içinde yer alır.



Merhaba Dünya Örneği

```
<html>
<head>
<title>Hello World!</title>
</head>
<body>
<h1 wicket:id="msg">Mesaj buraya ekleniyor</h1>
</body
<html>
```

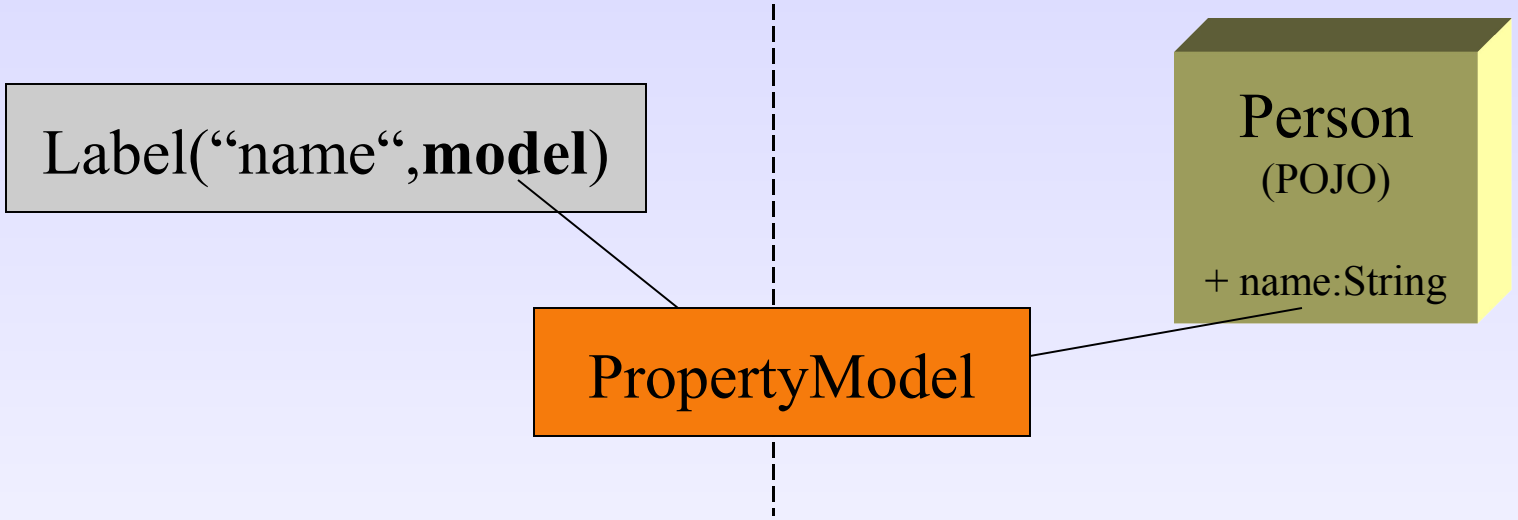
```
package helloworld;

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

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

}
```

Wicket Konseptleri

- Application
- Page
- Komponentler
- **Modeller**
- Paneller

Modeller

Wicket komponentleri ile veri taşıyıcı Java sınıfları bir araya getirmek için kullanılırlar.




```
new TextField("name",
                new PropertyModel(getPerson(), "name"));
```

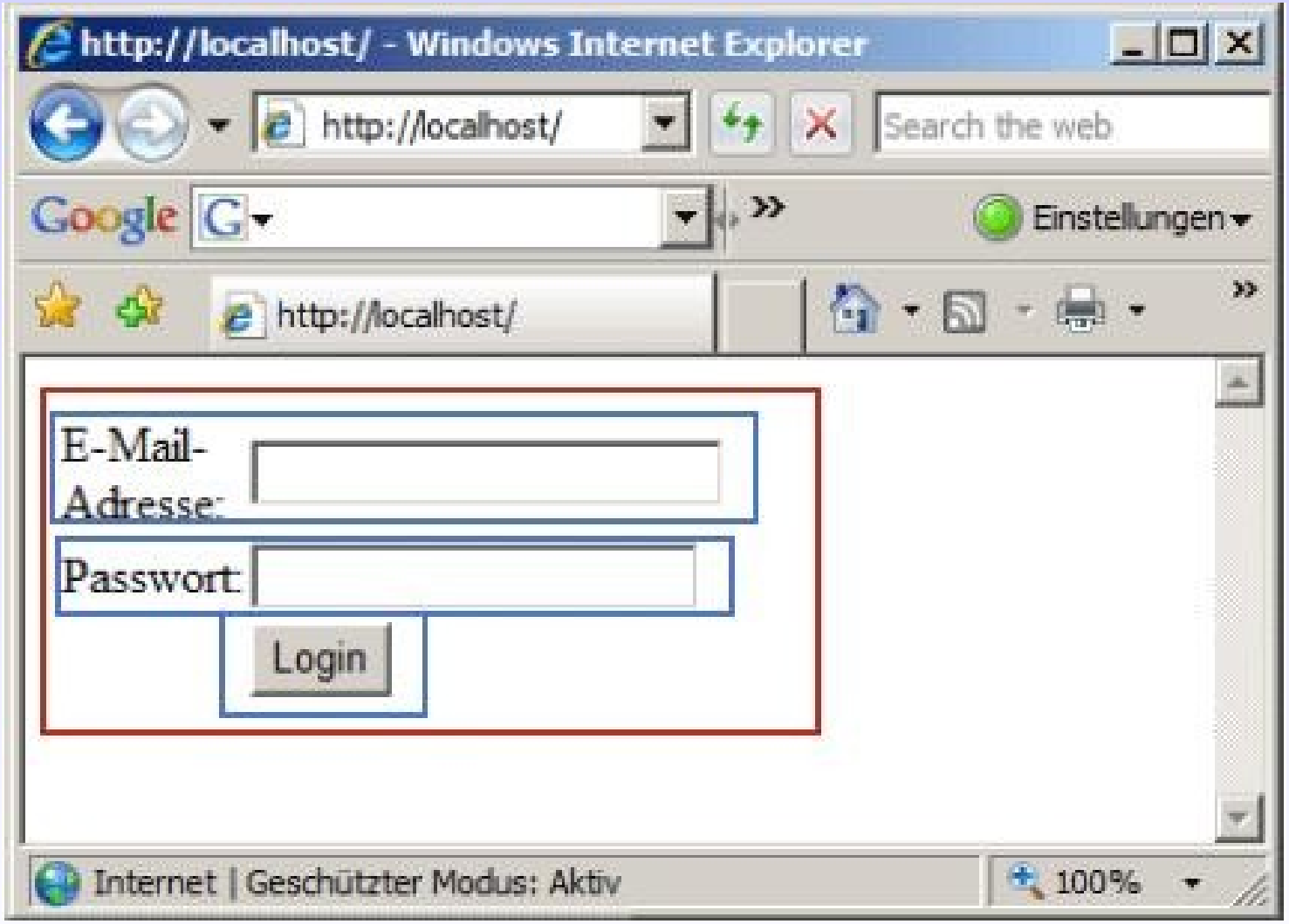
Wicket Konseptleri

- Application
- Page
- Komponentler
- Modeller
- **Paneller**

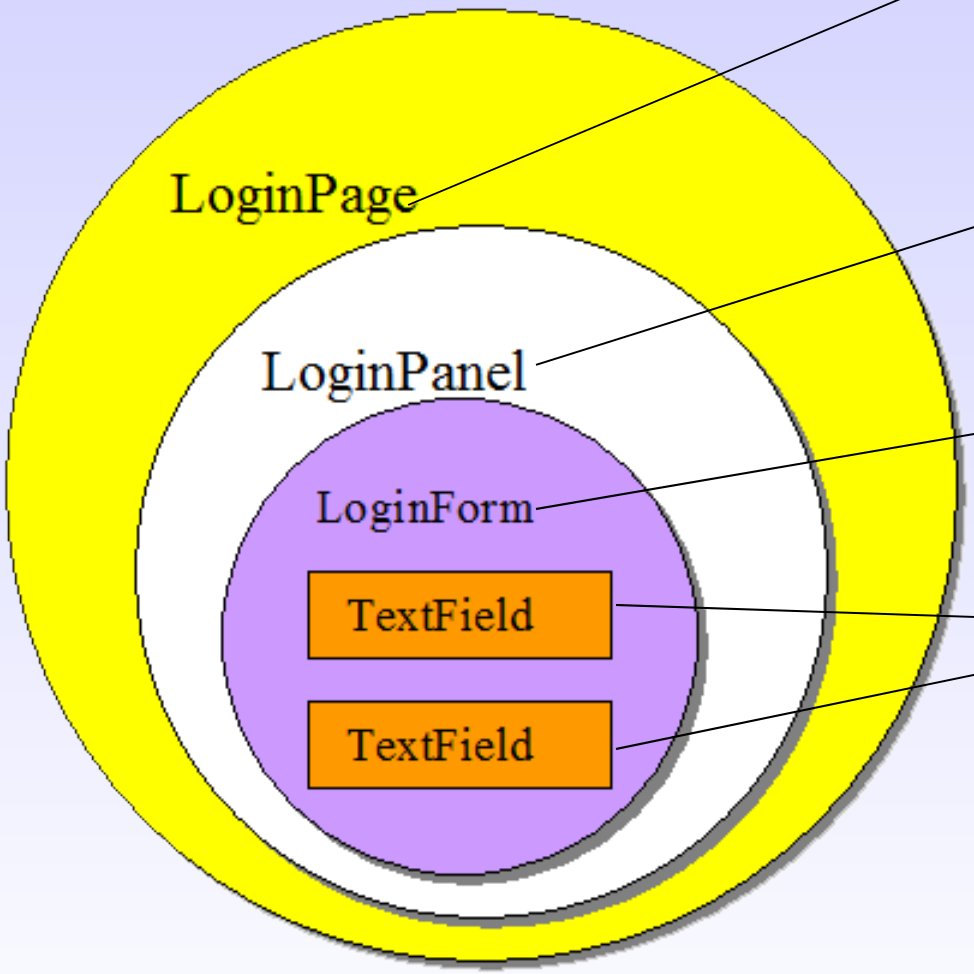
Paneller

- Birden fazla wicket komponentini gruplamak için kullanılır.
- Kendi HTML dosyası vardır.
- Paneller kullanılarak tekrar kullanılabilir web komponentleri oluşturulabilir.

Login Web Komponenti



Login Web Komp



LoginPage.java
LoginPage.html

LoginPanel.java
LoginPanel.html

LoginForm

TextField

PropertyModel

Account

Login Web Komponenti

LoginPage.java

```
import org.apache.wicket.markup.html.WebPage;

import smart.web.login.presentation.panel.LoginPanel;

public class LoginPage extends WebPage
{
    public LoginPage()
    {
        super();
        add(new LoginPanel("loginpanel"));
    }
}
```

Login Web Komponenti

LoginPage.html

```
<html>
<head/>

<body>
<div wicket:id="loginpanel"/>

</body>
</html>
```

Login Web Komponenti

LoginPanel.java

```
import org.apache.wicket.markup.html.panel.Panel;
import smart.web.login.presentation.form.LoginForm;

public class LoginPanel extends Panel
{
    public LoginPanel(final String arg0)
    {
        super(arg0);
        add(new LoginForm("loginform"));
    }
}
```

Login Web Komponenti

LoginPanel.html

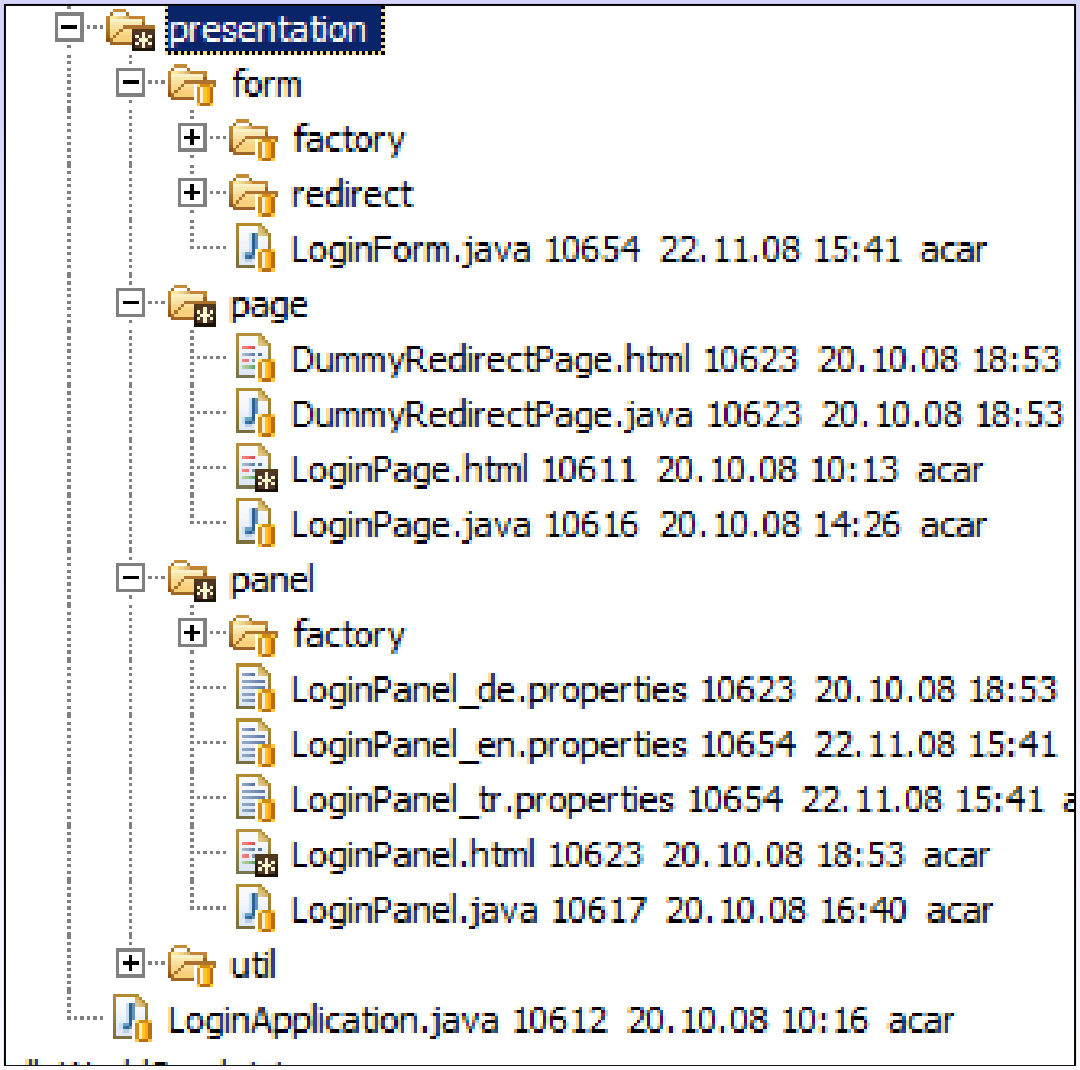
```
<wicket:panel>
<form wicket:id="loginform">
  <table width="100%" cellspacing="1">
  <tr>
    <td width="100%" class="text11" colspan="3">
      <div wicket:id="feedback"></div></td>
  </tr>
  <tr>
    <td width="13%" >
      <wicket:messagekey="login.email">Email</wicket:message>:
    </td>
    <td width="82%">
       <span wicket:id="email.border">
           <input size="20" maxlength="50„ wicket:id="email"
                  class="inputfield" />
        </span>
    </td> ..........
</wicket:panel>
```

Login Web Komponenti

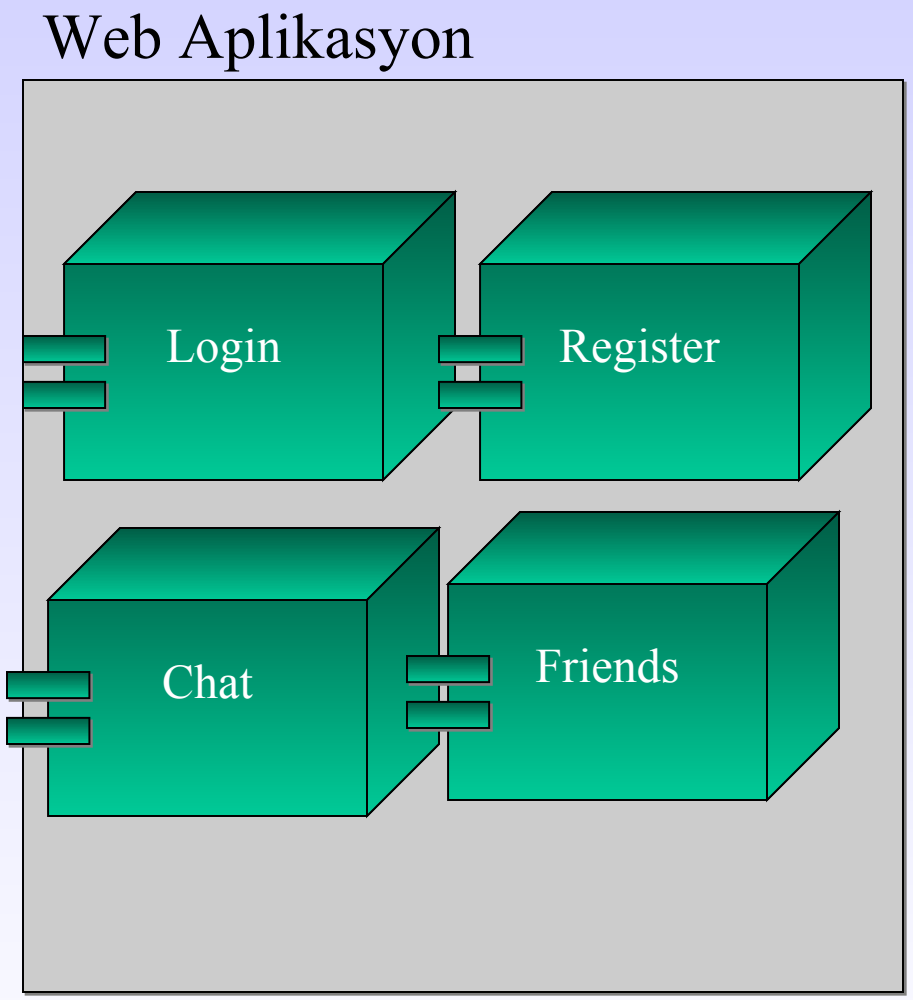
LoginForm.java

```
public class LoginForm extends Form
{
    public LoginForm(final String pId)
    {
        super(pId);
        add(new TextField("email",
                new PropertyModel(getAccount(), "email")).
                        setRequired(true));
        add(new PasswordTextField("password",
                 new PropertyModel(getAccount(), "password")).
                         setRequired(true));
    }

    protected final void onSubmit()
    {
        LoginResult result =
                  manager.login(getAccount().getEmail(),
                         getAccount().getPassword());

        // LOGIN OK
        if(result.getStatus() ==
                  LoginManagerStatusCodes.LOGIN_OK.getValue()) .......
}
```

Login Web Komponenti

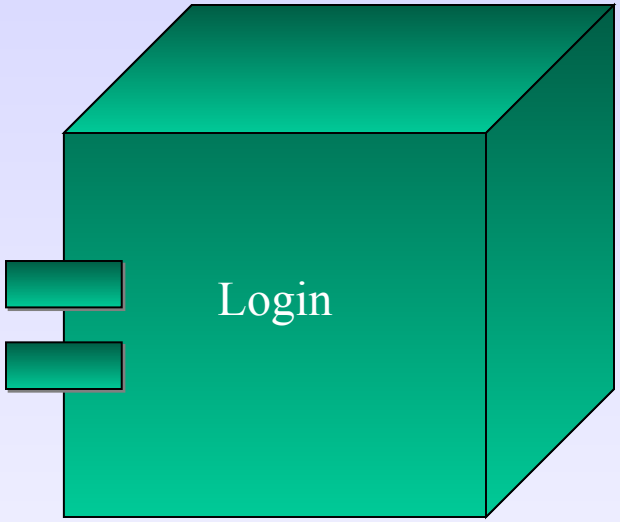


Web Komponent



- Bir web aplikasyonda konfigürasyon yapılarak kullanılan modül.
- Diğer komponent özelliklerine sahiptir.
- Jar dosyası olarak aplikasyona dahil edilebilir.

Web Komponent



Login komponentini bir **Jar** dosyası haline getirerek, istediğimiz bir projede kullanabiliriz.

Java

```
add(new LoginPanel("loginpanel"));
```

HTML

```
<div wicket:id="loginpanel"/>
```

SON

İlginiz için teşekkür ederim.

Kurumsal Java.com