

Modüler UI Kodlamak

Burak Bayramlı

Bu belge, \LaTeX ile üretilmiştir

UI Kod Parçaları

- Görsel olmayan Java kodları gibi, Web Beans UI (XHTML) sayfa kodları da modüler şekilde kullanılabilir.
- Niye? Belki bir görsel parçaya pek çok sayfada sürekli ihtiyacımız var... Meselâ bir “en iyi müşteriler” listemiz olabilir, ve bu listeyi birkaç değişik sayfada göstermek istiyoruz.

Modüler UI ve Web Beans

Web Beans'de modüler kodlama yapmak için üç yöntem mevcut:

- Her sayfada aynı görülecek XHTML kod parçası dahil etmek (`ui:include`)
- Her dahil edişte değişik görülebilecek “parametre geçme” yöntemi ile dahil etmek (`s:decorate`)

- Tüm sayfayı şablon bazlı geliştirmek ve göstermek (`ui` → `:define`, `ui:composition`)

Statik Kod Dahili

Statik Kod Dahili

Bu durumda kodlama oldukça basit. Meselâ eğer müşteri listemiz içerik olarak her dahil edildiği yerde değişmiyor ise, bu sayfayı statik olarak dahil edebiliriz. Dahil edilen dosya (diyelim `included.xhtml`)

```
<ui:composition xmlns= ... >
```

```
  <h:dataTable ..  
    <h:outputText ..  
</h:dataTable>
```

```
</ui:composition>
```

Dahil eden ise şöyle olabilir (diyelim `included.xhtml`)

```
<html ..  
  <div id="id3">  
    <ui:include src="/included.xhtml"/>  
  </div>  
</html>
```

Dinamik Kod Dahili

Dinamik Kod Dahili

Eğer dahil ettiğimiz XHTML parçasının geçilen belli bazı parametrelere göre değişmesini istiyorsak, o zaman kullanım şöyle değişecektir. Sayfa `included.xhtml`:

```
<ui:composition xmlns= .. >
  ..
  <ui:insert name="param1"/>
  ...
</ui:composition>
```

Sayfa `including.xhtml`:

```
<s:decorate template="/included.xhtml">  
  <ui:define name="param1">  
    [değişik olan tanım buraya]  
  </ui:define>  
</s:decorate>
```

UI Şablonu Kullanmak

UI Şablonu Kullanmak

- Bir Web sitesinde genellikle her sayfa için aynı kalan bir “yapı” vardır.
- Bu yapı, meselâ, üstte geniş UI bloğu (topbar), ve sol yanda, içinde bazı bağlantılar içeren ikinci bir UI bloğu (sidebar) şeklinde olabilir.
- Bu yapı çoğunlukla sayfadan sayfaya değişmeyecektir, fakat her sayfa, içerik alanı denilen orta kısma değişik

UI görüntüleri basacaktır.

- Şablon bazlı UI kodlaması, değişmeyen yerleri tek bir şablon dosyası içinde tanımlayarak, sadece değişen yerlerin her sayfa içinde kodlandığı geliştirme şeklidir.
- Bu durumda dahil *eden*, değişmeyen şablon, dahil edilen ise, değişen, genellikle daha az kod içeren içerik kısmıdır.

Şablon Tanımlamak

Şablon tanımı için bir şablon dosyası oluştururuz. Bu dosya Web Beans'de genelde `template.xhtml` diye isimlendirilir. Bu dosya normal XHTML, JSF ibareleri içerir.

Değişmesi gereken yerler ise `ui:insert` ile işaretlenip, boş bırakılır.

Şablon Tanımlamak

```
<html xmlns=...>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Test</title>
    <link href="css/screen.css" rel="stylesheet" type="text/css" />
  </head>
  <form>
    <div id="topBar"/>
      ...
      <ui:insert name="content12"/>
      ...
    </div>
  </h:form>
</div>
</html>
```

Şablon Kullanmak

Bu tanımdan sonra artık her sayfa bu dosyayı şablon (template) olarak gösterip, boş bırakılan yerlere kendi içeriğini koyabilecektir. Bunu yapacak her sayfa, `ui:composition` ibaresi ile başlamak zorundadır. Değişik içerik `ui:define` ile şablona bildirilir.

Şablon Kullanmak

```
<ui:composition xmlns= ...
                template="template.xhtml">
  <ui:define name="content12">
    <h:outputText ...
  </ui:define>
</ui:composition>
```

Geliştirme Süreci

Geliştirme Süreci

- Çoğunlukla programcılar uygulamaya başladıkları ilk sayfa üzerinde gerekli düzenlemeleri bir yerde yapar.
- Daha sonra *ikinci* sayfanın geliştirilmesi gerektiğinde, bir yeniden düzenleme (refactoring) tekniği kullanılarak, ortak kodların o tek sayfadan çıkartılıp şablon yapısına geçirilmesi gündeme gelir.

Görevler

Test - Windows Internet Explorer

http://localhost:8080/kitapdemo/home.seam

Links Akbank İstanbul Deniz Otobüsleri A.Ş. Cars

Test

garage 1 [Edit](#)
garage 2 [Edit](#)



Plaka
Tanım
Mevcut mu?
Olcu
Garaj

34 TTD 2202 örnek description

- XHTML/CSS dersinde ve seçilebilir URL listelerini işlediğimiz derste geliştirdiğimiz uygulamaları birleştirerek bazı ekler yapacağız.
- Bu eklerle garaj listesinin tek başına bir sayfa olarak değil, soldaki sidebar içinde çıkmasını istiyoruz.
- Görsel pozisyonlama hakkında bir not: Bir `<div>` bloğunu sayfa içinde direk kordinat vererek yerleştirmek için CSS'te

```
#id123 {  
    position: absolute;
```

```
top: ...px;  
left: ..px;  
}
```

ibaresini kullanmalısınız. Yâni her `<div>` kendi padding ayarlarını belirleyebildiği gibi, kendi kesin yerini de belirleyebilir: Dikkat: Bizim örneğimiz yerini belirleyen `<div>` bloğu, içerik bloğunu belirleyen “alt `<div>`” içinde olmalı, sidebar’ı tanımlayan dış `<div>`’de değil.

- İlk önce `list.xhtml` adlı bir dosyada garaj listesini gösterelim, ve bu dosyayı `home.xhtml`’den dahil ederek gerekli `<div>` ayarları ile sol sidebar’da gösterelim.

- Daha sonra yeni bir `dlist.xhtml` dosyası yazalım, bu dosya da garaj listesini gösterecek, fakat dahil ederken geçilen bir parametreye göre değişik başlıklar gösterebilecek. Bunu bir önceki listenin hemen altına koyabilirsiniz. Hatta bu dahil etmeyi iki kez yapın, her iki seferde de başlık değişik bir şey olsun. Yâni işiniz bittiğinde garaj listesi solda üç kez gösteriliyor olacak.
- Bunlar bittikten sonra bile herhangi garaj üzerinde “edit” tıklaması yapıldığında görsel yapımızdan oldukça uzak çoğu boş bir sayfaya geldiğimizi görüyoruz. Şimdi bu sayfanın da üst ve sol görsel blokları aynen göstermesini

sağlayalım (ki garaj listesini gösteren sidebar buna dahildir)

- Bunun için `template.html` adlı bir şablon yaratıp, `home.html`'deki ortak yerleri tanımlayan yerleri bu şablona atalım. Ortadaki içerik kısmını sayfaların kendisine bırakım. Bundan sonra `garage.html` ve `home.html` dosyaları sadece içeriği tanımlayacaktır. Şimdi bir garaj üstünde “edit” tıklaması yapalım ve yapımızın artık değişmediğini görelim.
- Not: UI refactoring yapmak biraz uğraştırabilir, birkaç

hata yapılması da normaldir. Başladığınız noktaya dönebilmek için `home.xhtml`'in orijinal hâlinin bir yerde kopyasını tutabilirsiniz.

Son