

GUI Birimleri

Burak Bayramlı

Bu belge, \LaTeX ile üretilmiştir

Tüm birimler

Bir Web uygulaması için gereken tüm form birimlerini nasıl kodlayacağımızı göreceğiz.

- Inputbox
- Command button
- Checkbox
- Radio box

- Dropdown list
- Çoklu checkbox
- Image

Örnek

Cars - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080

Wizaa Cars Bilgidata.com Azureus



License Plate

Description

Available?

Size

Garage

Available?	License Plate	Description
●	34 TTD 2202	ornek description
●	dfgd	fgdsg

Hatırlatma

Web Beans bir POJO'yu alıp bu POJO'nun değerlerini bir form'a koymayı, ya da formda kullanıcının girdiği değerleri alıp otomatik olarak POJO'yu güncellemeyi arka planda kendisi halleder.

Checkbox

Checkbox

XHTML

```
<h:selectBooleanCheckbox id="prop1"  
    value="#{obj.prop1}"/>
```

Java POJO

```
public class Xyz {  
    ...  
    private Boolean prop1;  
}
```

Dropdown List - Statik Veri

Dropdown List - Statik Veri

Statik ve çok dilliliği desteklemek

XHTML

```
<h:selectOneMenu value="#{obj.prop1}" id="prop1">
  <f:selectItem itemLabel="#{messages.key1}"
    itemValue="a"/>
  <f:selectItem itemLabel="#{messages.key2}"
    itemValue="b"/>
  <f:selectItem itemLabel="#{messages.key3}"
```

```
                itemValue="c"/>  
</h:selectOneMenu>
```

Java

```
public class Xyz {  
    ...  
    private String prop1;  
    ..  
}
```

message.properties

```
key1=Key 1 Tanimi  
key2=Key 2 Tanimi
```

key3=Key 3 Tanimi

Dropdown List - Dinamik Veri

Dropdown List - Dinamik Veri

XHTML

```
<h:selectOneMenu value="#{xyzService.obj1}">
  <f:selectItems value="#{xyzService.objList}"/>
</h:selectOneMenu>
```

Java Action Bean

```
public List<SelectItem> getObjList()
{
    List<SelectItem> result =
```

```
        new ArrayList<SelectItem>();

// boş bir SelectItem ekleyin burada

Query query = ..

for (Garage g : query.getResultList())
{
    result.add(new SelectItem(g.getProp1(),
                              g.getProp2));
}

return result;
```

```
}
```

```
String ObjectValue;
```

```
public void setObjectValue(String x) { ... }
```

```
public String getObjectValue() { ... }
```

```
...
```

Dropdown List - Açıklama

- `selectOneMenu` ve `selectItems` etiketlerine bir liste vermek gerekiyordu. Bu liste arka planda o sayfaya servis eden Action Bean'den gelecektir. Bu normal! Çünkü veriye yazma, button hareketlerini karşılamak gibi işlemleri zaten o yapıyordu.
- Seçimden sonra listeden “seçilen” objenin yine Bean üzerinde set edilmesi de mantıklı. Ki böyle oluyor.

- Böylece liste gösterilip, seçim yapıp sayfa buttonu tıklendiğinde, o tıklamayı karşılayan action metodu, içindeki “seçilmiş” obje referansta bir obje olduğunu farz edebilecektir.

Dropdown List

```
public String add()
{
    // objectValue String'i kullanarak gereken objeyi find
    // ile yükle

    // obj1 üzerinde set et

    em.persist(obj1);
}
```

Imaj Göstermek

Imaj Göstermek

```

```

İmaj Göstermek

İmaj göstermenin ikinci bir yolu, `<h:graphicImage>` etiketidir. Bu etiket, aynen normal HTML `img` komutunda olduğu gibi `width` ile genişlik ölçüsü alabilir, `value` ile yeri belirtilen imaj dosyasını ekrana basabilir.

Fakat ek olarak bir özelliği daha var. Aslında genel olarak tüm `<h: komutları için geçerli, bu etiketler içinde ek bir rendered="#{obj.prop1 == [DEĞER]}" gibi bir ifade kullanmak mümkündür. Bu ifadeye göre, sadece ve`

sadece `obj.prop1 DEĞER`'e eşit ise o etiket gösterilecektir.

Karşılaştırma operatörü Java dilindeki operatörler ile aynıdır, `>`, `<`, `vs.` gibi operatörlerin hepsi kullanılabilir.

h:dataTable Tanımları

h:dataTable Tanımları

Her kolonun başında bir kolon tanımı olmasını istiyorsak,

```
<h:dataTable value=".." var="x">
  <h:column>

    <f:facet name="header">
      <h:outputText value="#{messages.key1}"/>
    </f:facet>

    #{x.prop1}
  </h:column>
```

```
<h:column>  
  <f:facet name="header">  
  ...  
</h:column>  
  
</h:dataTable>
```

Çoklu checkbox

Çoklu checkbox

Eğer bir checkbox *seti* arasından bir ya da daha fazlasını seçmemiz gerekiyor ise, o zaman çoklu checkbox kullanmamız gerekiyor.

XHTML

```
<h:dataTable value="#{objList}" var="x">  
  ...  
  <h:column>  
    <h:selectBooleanCheckbox value="#{objSelections[x]}" />
```

```
</h:column>  
</h:dataTable>
```

Java Action Bean

```
..  
@Out(required=false)  
Map<Xyz, Boolean> objSelections =  
    new HashMap<Xyz, Boolean>();  
...  
public String delete() {  
    for (Car item : objSelections.keySet()){  
        Boolean selected = carSelections.get(item);  
        if (selected) {
```

```
        // seçilen obje ile işlem yap
    }
}
// listeyi tekrar oluştur

return "/home.xhtml";
}
```

Görevler

- Resmini gördüğümüz ekranı tasarlayacağız. Gereken alan isimleri o resimde belirtiliyor.



- Aynen resimde olduğu gibi en üstte bir resim olacak.
- Size için `properties` dosyasından gelen `Small`, `Medium` ve `Large` olan üç seçenek koyun. Bu seçilince database o alan için `s`, `m` ve `l` kodları yazılsın.
- Bir arabanın garaj ile bağlantısı kurulmalı. Bunun için gereken garaj listesi database'den alınacak, yani bean üzerindeki bir metot bunu Hibernate üzerinde alacak.
- Araba eklendikten sonra garaj ilişkisi düzgün kurulmuş olmalı. Bunun örneğini daha önce görmüştük.

- Araba silmek için yeni bir button ekleyelim. Bu ikinci button EJB üzerindeki `delete` adlı metodu çağırсын. Silinecek arabalar checkbox ile seçilmeli.
- Araba listesinde her arabanın başında `available` öğesine uygun şekilde bir top ikonu gösterilmelidir. Eğer bu alan `false` ise bir kırmızı ikon (`images/red.jpg`), eğer `true` ise yeşil bir ikon (`images/green.jpg`) ekrana basılmalı.
- Bütün alanlar Türkçe ve İngilizce'de tanımlarla gelmeli.

- Tavsiye: Birimleri teker teker ekleyip, testini yapıp bir sonraki birime geçmek.

Import'lar

```
import java.io.Serializable;
import java.util.List;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.bilgidata.kitapdemo.pojo.Car;
import org.bilgidata.kitapdemo.pojo.Garage;
import org.jboss.seam.annotations.Destroy;
import org.jboss.seam.annotations.Factory;
```

```
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Logger;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;
import org.jboss.seam.annotations.datamodel.DataModel;
import org.jboss.seam.core.FacesMessages;
import org.jboss.seam.log.Log;
import javax.faces.model.SelectItem;
import java.util.ArrayList;
import static org.jboss.seam.ScopeType.SESSION;
```

Import'lar

```
import org.bilgidata.kitapdemo.pojo.*;  
import javax.faces.model.SelectItem;  
import javax.ejb.Local;  
import java.util.List;
```

Son