

# Mesajlar ve Çok Dil Desteđi

**Burak Bayramlı**

Bu belge,  $\text{\LaTeX}$  ile üretilmiştir

# Yabancı diller desteği

Bir uygulamanın yabancı dilleri desteklemesinden bahsedilince, birkaç değişik konu devreye girer:

- Aynı site, Mozilla/Explorer dili değişince tüm sayfa dilini değiştirsin
  - ★ Hata mesajları
  - ★ Form tanımları (labels)
  - ★ Büyük anlatımlar

- Yabancı karakterler veri tabanından okunup oraya yazılabilir

# Veri tabanı

- Veri tabanının meselâ Türkçe karakterleri desteklemesi için Java EE bağlamında en alta tabaka olan JDBC sürücüsünün “encoding” ayarı değişmelidir.
- -ds.xml dosyası veri tabanı ayarımızı yapıyor
- Bu dosyada <connection-url> ayarına useUnicode=true&characterEncoding=UTF-8 eklenmelidir. Yâni encoding, UTF-8 olarak tanımlanmalıdır.

# Veri tabanı

Artık form'larınız veri tabanına doğru karakterde yazılacak ve oradan okunacaktır.

# Mesajlar

- Genel bağlamda kullanıcıya mesajlar, Seam FacesMessages objesi ile iletilir.
- Bu obje, Seam Action seviyesinde yaratılır.

```
@In(create=true)
```

```
private FacesMessages facesMessages;
```

- `facesMessages.add("Mesaj")` ile mesajı mesaj listesine ekleriz.

- Bu çağrı aynı EJB metodu içinde bile birkaç kez işletilebilir.
- Daha sonra toplanan mesajlar, yönlendirildiğiniz sayfa (xhtml) üzerinde `<h:messages globalOnly="true" />` tanımı ile gösterilebilir.
- Bu etiket, yani hata mesajları, genelde göze çarpacak bir renk ile işaretlenir (meselâ kırmızı: `<font color="#FF0000">...</font>`)

## Çok dil ayarı

Eğer kullanıcı tarayıcı dilini değiştirdiğinde o dilden mesajlar görmesini istiyorsak, `faces-config.xml` şunları ekleriz:

```
<faces-config>
  ...
  <application>
    <message-bundle>messages</message-bundle>
    <locale-config>
      <default-locale>tr</default-locale>
```

```
    <supported-locale>en</supported-locale>  
    <supported-locale>tr</supported-locale>  
  </locale-config>  
</application>
```

...

```
</faces-config>
```

## Çok dil ayarı

resources/WEB-INF/web.xml dosyasında tüm Http request ve response nesneleri üzerinde encoding ayarının UTF-8'e çevrilmesi gerekiyor.

```
<filter>  
  <filter-name>  
    RequestCharacterEncodingFilter  
  </filter-name>  
  <filter-class>  
    vs..vs...RequestCharacterEncodingFilter
```

```
</filter-class>
<init-param>
  <param-name>requestCharacterEncoding</param-name>
  <param-value>UTF-8</param-value>
</init-param>
</filter>

<filter-mapping>
  <filter-name>
    RequestCharacterEncodingFilter
  </filter-name>
  <url-pattern>*.seam</url-pattern>
</filter-mapping>
```

## Mesajlar nerede tutulur

Bu yapılmca messages baz ismi üzerinden, İngilizce için `messages_en.properties`, Türkçe için `messages_tr.properties`, vs.. şeklindeki dosyalar içinde *o dile has* mesajları koyacağımızı beyan etmiş oluyoruz.

## Mesajlar nerede tutulur

Peki `_tr` ya da `_en` eklentileri nereden geliyor? Bu kelimeler tarayıcınız tarafından kullandığı dile göre JBoss'a her çağrıda otomatik olarak zaten gönderilmektedir.

## Mesajlar nerede tutulur

Üstte ismi verilen `properties` dosyaları, `isim=değer` şeklindeki basit dosyalardır. İsim için genellikle (convention) İngilizce değişken ismine benzeyen bir anahtar ismi vermektir, değer için ise o mesajın o dildeki gereken metni koyulur. Meselâ `messages_en.properties`'de

```
insert.error=There was an error while inserting object
```

`messages_tr.properties`'de

```
insert.error=Araba kaydini eklerken hata olustu
```

## Ek

Böylece Türkçe mesajlar basılacaktır. Veri tabanını da hallettik, fakat veri tabanından değil `properties` dosyasından gelen Türkçe mesajlar *içindeki* Türkçe karakterler hala problem oluyor. Yâni “kayidini” yerine “kayıldını” yazabilmek istiyoruz.

## native2ascii

Bunu gerçekleştirmek için düzgün karakterler ile yazılmış bir dosyayı `native2ascii` ile işlememiz gerekiyor. Bu program “ı” yerine `\u00c5\u0178` kodunu koyuyor ve JBoss bu karakteri doğru karaktere çevirebiliyor. İçinde tüm mesajlarımızın olduğu `messages_tr.properties` dosyasını “`native2ascii messages_tr.properties`” diyerek bir çırpıda işleyebiliriz.

# Tamam

- Database ayarı çok dile hazır
- Properties dosyaları gerekli yerlerde ve isimlerde
- Properties dosyaların içinde Türkçe karakterler ascii kodlarını taşıyor
- .. şimdi bu mesajları kullanalım.

# Sayfalarda

Artık, sayfalarımızda her dilde değişecek tanımlar basmak için `<h:outputText value="#{messages.key1}"/>` ile `messages_x.properties` dosyası içindeki `key1` anahtarının *o dildeki* değerine erişebilmiş oluyoruz.

# Mesajlarda

Çok dilli bir mesaj ekleyebilmek için

```
facesMessages.add("#{messages['key3']}");
```

gibi bir ibare kullanabiliyoruz.

# Görevler

- SeamHibSimpleTr projesini alın. Bu projede gerekli yerlere ayarları ekleyerek yabancı karakter desteğini gerçekleştirin. `cars-ds.xml`, `persistence.xml`, `web.xml`, `faces-config.xml` gereken yerlerde değişmeli.
- Standard Car nesnesini giren kodunuzu buraya getirin.
- `messages_tr.properties` ve `messages_en.properties` dosyalarında İngilizce ve Türkçe mesajlarınızı koyun.

- `home.xhtml` içinde plaka ve tanım verileri girilsin. Ek olarak, şimdi bu alanların tanımlarını koyacağız, bunlar İngilizce ve Türkçe ayrı olarak ekrana gelebilmeli.
- `em.persist`, aynı anahtar'dan aynı iki veri girilince `Exception` atma yeteneğine sahiptir. Bu `Exception`'ı `catch` edin ve `catch` içinde iken iki hata mesajı yazın. Biri, hatalı olan objenin `licensePlate`'ini belirten İngilizce mesaj, öteki çok dilli ama her hata için aynı olan bir mesaj olmalı.
- Bu mesaj `home.xhtml` içinde kırmızı renk ile en üstte

gelecek şekilde gösterilmeli.

- Test için tanımı Türkçe karakterlerden oluşan bir araba girin. Daha sonra Internet Explorer'dan tarayıcı dilini değiştirin. Form alanlarının tanımı değişiyor mu?

# Import'lar

```
import java.util.List;
import javax.ejb.Remove;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.jboss.seam.annotations.Destroy;
import org.jboss.seam.annotations.In;
import org.jboss.seam.annotations.Name;
import org.jboss.seam.annotations.Scope;
import org.jboss.seam.core.FacesMessages;
```

```
import org.jboss.seam.annotations.Logger;  
import org.jboss.seam.log.Log;  
import java.io.Serializable;  
import org.jboss.seam.annotations.datamodel.DataModel;  
import org.jboss.seam.annotations.Factory;
```

# Son