JSF Conversion

Data Conversion

- UI Component değerinin **String – Java nesne** şeklinde **karşılıklı dönüşümü**ne conversion adı verilir
- UI bileşenlerinin değerleri **sunucundan tarayıcıya** gönderilirlen **Java nesne – String dönüşümü** gerçekleşir
- Değerlerin **tarayıcıdan sunucuya** gönderilmesi (submit) sırasında da **String – Java nesne** dönüşümü gerçekleşir

Data Conversion

- Örneğin, Date tipinde bir property değeri sunucudan istemciye yyyy/mm/dd formatında bir String olarak gönderilir
- Tarayıcı üzerinde edit edilen String tarih değeri sunucuya gönderildiğinde, örneğin "2004/11/04" String değeri Java Date nesnesine çevrilir
- Bu **çevrimi yapan** nesnelere **Converter** adı verilir

Data Conversion

- JSF, UI bileşenin bind edildiği property tipi primitif, BigInteger, BigDecimal ise değeri **otomatik dönüştürür**
- Diğer tiplerde (Date vb dahil) Converter tanımlanması gerekir
- JSF **javax.faces.convert.Converter** arayüzünü implement eden standart converter'lar sunar
- **Custom converter** da yazılıp kullanılabilir

Standart JSF Converter Tipleri

- BigDecimalConverter
- BigIntegerConverter
- BooleanConverter
- CharacterConverter
- ShortConverter
- ByteConverter
- DateTimeConverter
- DoubleConverter
- FloatConverter
- LongConverter
- IntegerConverter
- NumberConverter

Standart JSF Converter Tipleri

```
<h:inputText value="#{userBean.birthDate}">
   <f:convertDateTime pattern="dd/MM/yyyy" type="date"/>
</h:inputText>
```

```
<h:inputText value="#{userBean.weight}">
   <f:convertNumber minFractionDigits="2"/>
</h:inputText>
```

NumberConverter ve **DateTimeConverter**'ın tag'leri de mevcuttur
Bu tag'ler yardımı ile conversion işleminin detayları belirlenebilir

Standart JSF Converter Tipleri

```
<h:inputText id="age" value="${userBean.age}"
                converter="javax.faces.Integer" />
```

```
<h:inputText id="age" value="${userBean.age}">
   <f:converter converterID="javax.faces.Integer" />
</h:inputText>
```

Custom Converter

```
@FacesConverter("entityConverter")
public class EntityConverter implements Converter {
    private EntityService entityService;
    public EntityService getEntityService() {
        return entityService;
    }
    @Override
    public Object getAsObject(FacesContext context,
                              UIComponent component, String value) {
        String[] tokens = value.split(":");
        try {
            return entityService.getEntity(
                    Class.forName(tokens[0]),Long.parseLong(tokens[1]));
        } catch (NumberFormatException | ClassNotFoundException e) {
            throw new ConverterException(e);
        }
    }
    @Override
    public String getAsString(FacesContext context,
                              UIComponent component, Object value) {
        return value.getClass().getName() +
                    ":" + ((BaseEntity)value).getId();
    }
}
```

Custom Converter Kullanımı

```
<h:selectManyListbox value="${vet.specialties}"
                     converter="entityConverter">
   <f:selectItems var="specialty"
      value="${vetDetailBean.availableSpecialties}"/>
</h:selectManyListbox>
```

```
<h:selectManyListbox value="${vet.specialties}">
   <f:selectItems var="specialty"
      value="${vetDetailBean.availableSpecialties}"/>
   <f:converter converterId="entityConverter"/>
</h:selectManyListbox>
```

İletişim

www.harezmi.com.tr

www.java-egitimleri.com

info@harezmi.com.tr

info@java-egitimleri.com

@HarezmiBilisim

@JavaEgitimleri