Using Standard Converters

Introduction

- The JavaServer Faces implementation provides a set of Converter implementations that you can use to convert component data.

Introduction

- The standard Converter implementations, located in the javax.faces.convert package, are as follows:

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

Introduction

- A standard error message is associated with each of these converters. If you have registered one of these converters onto a component on your page, and the converter is not able to convert the component's value, the converter's error message will display on the page.

Introduction

- Data conversion is the process of converting, or transforming, one data type into another. JSF will provide implicit conversion when you map a component's value to a managed bean property of a Java primitive type or Object types of BigDecimal and BigInteger. Alternatively JSF provides a set of standard converters that may be explicitly specified using the converter attribute of a UIComponent tag. Let's take a look at both the implicit and explicit conversion techniques.

Implicit Conversion

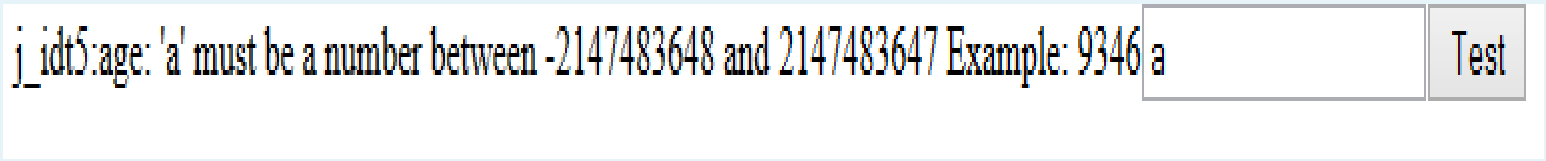
- Managed Bean

```
@ManagedBean
@Named(value = "myBean")
@SessionScoped
public class MyBean {

    private int age;

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public MyBean() {
    }

}
```

Implicit Conversion

- JSF:try to set string in text field will cause error message to be displayed...

```
<h:body>
    <h:form>
        <h:message showSummary="true" for="age"></h:message>
        <h:inputText id="age" value="#{myBean.age}"></h:inputText>
        <h:commandButton value="Test"></h:commandButton>
    </h:form>
</h:body>
```



Explicit Conversion

```
<h:body>
    <h:form>
        <h:message for="age"></h:message>
        <h:inputText id="age" value="#{myBean.age}">
            <f:converter converterId="javax.faces.Integer"></f:converter>
        </h:inputText>
        <h:commandButton value="Test"></h:commandButton>
    </h:form>
</h:body>
```

DateTime and Number Converter tags

- DateTime and Number converter tags supply attributes that may be used for additional data conversion precision.
- Next example shows how you can use DateTime converter with different values

Managed Bean

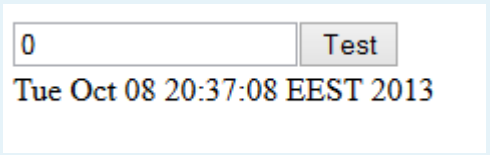
- Note the type in Managed Bean has to be java.util.Date!

```
@ManagedBean
@Named(value = "myBean")
@SessionScoped
public class MyBean {

    private int age;
    private Date date = new Date();

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public MyBean() {
    }
}
```

JSF Page

- No Conversion

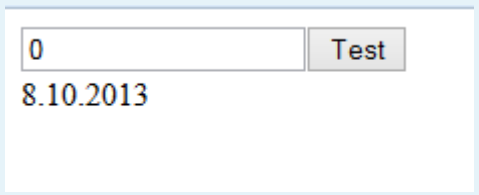
```
<h:body>
    <h:form>
        <h:message for="age"></h:message>
        <h:inputText id="age" value="#{myBean.age}">
            <f:converter converterId="javax.faces.Integer"></f:converter>
        </h:inputText>
        <h:commandButton value="Test"></h:commandButton><br/>
        <h:outputLabel value="#{myBean.date}">
        </h:outputLabel>
    </h:form>
</h:body>
```



JSF Page

- With Conversion

```
<h:body>
    <h:form>
        <h:message for="age"></h:message>
        <h:inputText id="age" value="#{myBean.age}">
            <f:converter converterId="javax.faces.Integer"></f:converter>
        </h:inputText>
        <h:commandButton value="Test"></h:commandButton><br/>
        <h:outputLabel value="#{myBean.date}">
            <f:convertDateTime dateStyle="short"></f:convertDateTime>
        </h:outputLabel>
    </h:form>
</h:body>
</html>
```



DateTime Converter

- dateStyle attribute possible values are:
  - default ->Jan 1, 2006 10:05:30 AM
  - short->1/1/06 10:05:30 AM
  - medium->Jan 1, 2006 10:05:30 AM
  - long-> January 1, 2006 10:05:30 AM
  - full-> Sunday, January 1, 2006 10:05:30 AM

DateTime Converter

- pattern attibute can be set to following values:
  - yyy. MM. dd 'at' HH:m
  - EEE, d MMM yyyy HH:mm:ss Z
  - MM/dd/yy
  - hh 'o''clock' a, zzzz
  - yyyy.MM.dd 'at' HH:mm:ss z
- Example:

```
<f:convertDateTime type="both" pattern="MM/dd/yyyy h:mm a"
dateStyle="short" timeStyle="medium" />
```

Number Converter

- In JSF, “f:convertNumber” is a standard converter, which converts String into a specified “Number” format. In addition, it’s also used as a validator to make sure the input value is a valid number.

Number Converter Example

```
@ManagedBean
@Named(value = "myBean")
@SessionScoped
public class MyBean {

    private int age;
    private Date date = new Date();
    private double amount = 2.90012d;

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public MyBean() {
    }
}
```

Number Converter Example

```
</h:head>
<h:body>
    <h:form>
        <h:message for="age"></h:message>
        <h:inputText id="age" value="#{myBean.age}">
            <f:converter converterId="javax.faces.Integer"></f:converter>
        </h:inputText>
        <h:commandButton value="Test"></h:commandButton><br/>
        <h:outputLabel value="#{myBean.date}">
            <f:convertDateTime dateStyle="short"></f:convertDateTime>
        </h:outputLabel><br/>
        <h:outputText value="#{myBean.amount}">
            <f:convertNumber maxFractionDigits="2"></f:convertNumber>
        </h:outputText>
    </h:form>
</h:body>
```

Number Converter Example

```
<h:outputText value="#{myBean.amount}">
    <f:convertNumber pattern="#0.000" />
</h:outputText>
```

0 Test

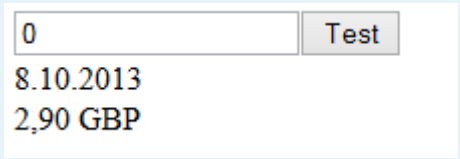
8.10.2013

2,900

Number Converter Example

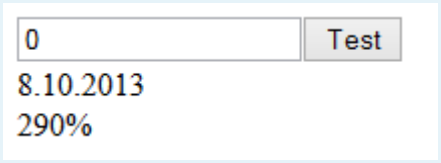
- The currencyCode is defined in ISO 4217. To use currencyCode attribute, the type attribute have to change to “currency“.

```
<h:outputText value="#{myBean.amount}">
    <f:convertNumber currencyCode="GBP" type="currency" />
</h:outputText>
```



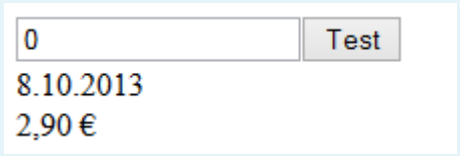
Number Converter Example

```
<h:outputText value="#{myBean.amount}">
    <f:convertNumber type="percent" />
</h:outputText>
```



Number Converter Example

```
<h:outputText value="#{myBean.amount}">
    <f:convertNumber currencySymbol="€" type="currency" />
</h:outputText>
```



Custom Converter

- If you need to specify some custom converter (like for email address or url) you need to define the converter yourself.
- Defining a custom converter in JSF is a three step process:
  - Create a converter class by implementing javax.faces.convert.Converter interface.
  - Implement getAsObject() and getAsString() methods of above interface.
  - Use Annotation @FacesConvertor to assign a unique id to the custom convertor.

Helper Class

```
public class UriHelper {
    private String url;

    @Override
    public String toString() {
        return url;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }
    public UriHelper(String url){
        this.url = url;
    }
}
```

Converter Class

```
@FacesConverter("com.opiframe.converters.CustomConverter")
public class CustomConverter implements Converter {

    @Override
    public Object getAsObject(FacesContext context, UIComponent component, String value) {
        URI uri;
         StringBuilder url = new StringBuilder();

        if(!value.startsWith("http://", 0)){
            url.append("http://");
        }

        url.append(value);
        try
        {
           new URI(url.toString());
        }
        catch(URISyntaxException e)
        {
            FacesMessage msg = new FacesMessage("Error converting URL",
            "Invalid URL format");
            msg.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ConverterException(msg);
        }
        UriHelper helper = new UriHelper(url.toString());
        return helper;
    }

    @Override
    public String getAsString(FacesContext context, UIComponent component, Object value) {
        return value.toString();
    }

}
```

Managed Bean

```
@ManagedBean
@Named(value = "myBean")
@SessionScoped
public class MyBean {

    private int age;
    private UriHelper uri;
    private Date date = new Date();
    private double amount = 2.90012d;

    public UriHelper getUri() {
        return uri;
    }

    public void setUri(UriHelper uri) {
        this.uri = uri;
    }
```

JSF

```
  </h:outputText><br/>
  <h:inputText label="URL" value="#{myBean.uri}">
      <f:converter converterId="com.opiframe.converters.CustomConverter"></f:converter>
  </h:inputText>
h:form>
```