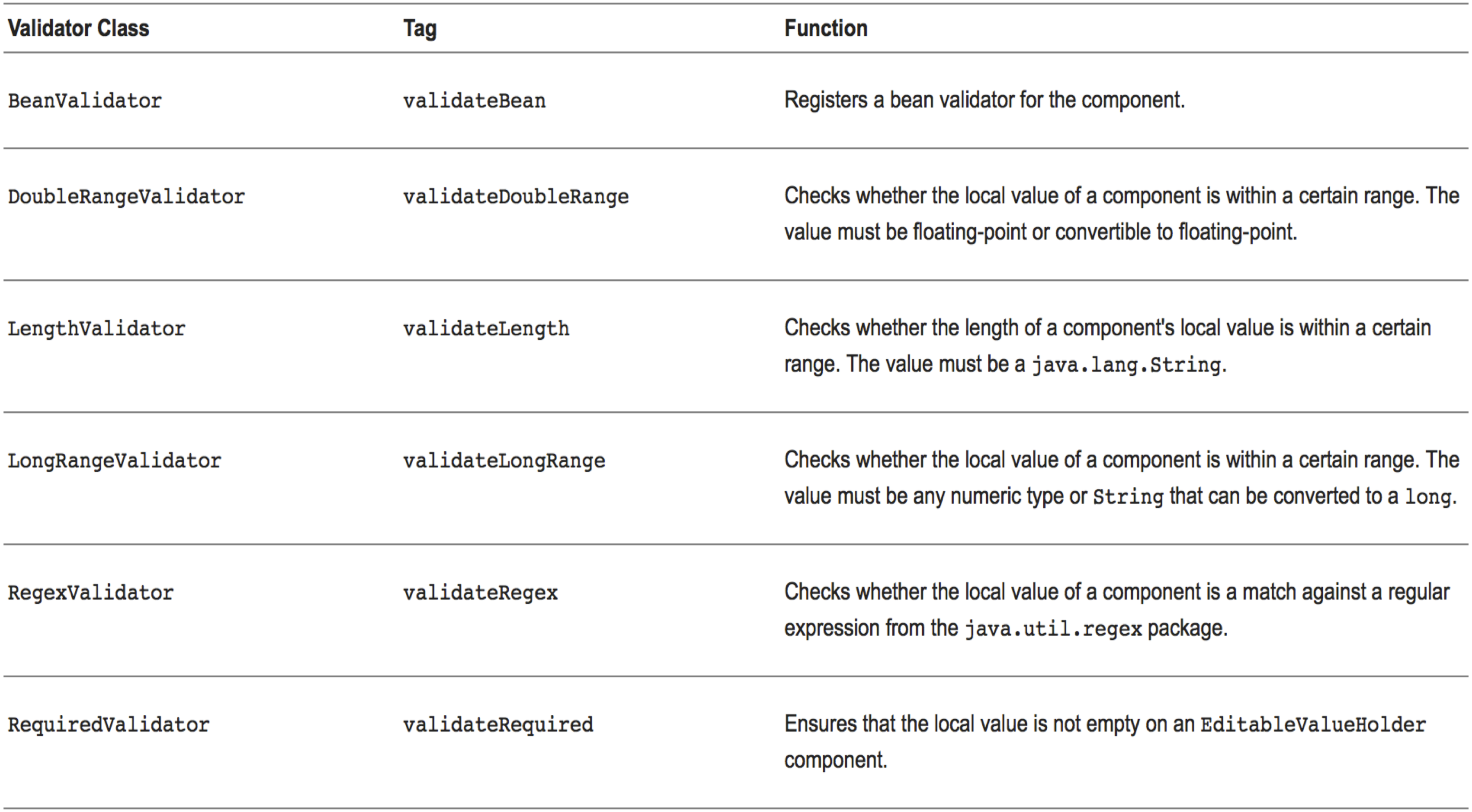
# Java EE Topics 1

- Expression Language (Hybrid Activity 3)
- Validators.  (Burns & Schalk, Chapter 8)
- Converters (Burns & Schalk, Chapter 8)
- Fun with Javascript

## Faces Validation System

- Validation applies to data coming from the user to our program
- Our Address Book example has the following examples of data validation:
  - Birthdays of the form DD/MM/YYYY and the date must be in the past
  - Email addresses match a regular expression representing all email addresses
  - Phone numbers must be 10 digits

# JSF Standard Validators



| Validator Class | Tag | Function |
|---|---|---|
| `BeanValidator` | `validateBean` | Registers a bean validator for the component. |
| `DoubleRangeValidator` | `validateDoubleRange` | Checks whether the local value of a component is within a certain range. The value must be floating-point or convertible to floating-point. |
| `LengthValidator` | `validateLength` | Checks whether the length of a component's local value is within a certain range. The value must be a `java.lang.String`. |
| `LongRangeValidator` | `validateLongRange` | Checks whether the local value of a component is within a certain range. The value must be any numeric type or `String` that can be converted to a `long`. |
| `RegexValidator` | `validateRegex` | Checks whether the local value of a component is a match against a regular expression from the `java.util.regex` package. |
| `RequiredValidator` | `validateRequired` | Ensures that the local value is not empty on an `EditableValueHolder` component. |

## Validator Example

```
<h:inputText id="quantity" size="4" value="#{item.quantity}">
<f:validateLongRange minimum="1"/>

</h:inputText>

<h:message for="quantity"/>
```

## Custom Validator

Here, **MyOwnJavaConv.java** is a class that implements the **Validator** interface (see https://docs.oracle.com/javaee/6/tutorial/doc/bnauw.html)

```
<td><h:inputText id="homePhone"
             value="#{contactController.selected.homePhone}"
              title="#{bundle.CreateContactTitle_homePhone}" >
              <f:validator validatorId=”MyOwnJavaConv” />
        </h:inputText></td>
                  <td><h:message for="homePhone" /></td>
```

## JSF Converter System

- Converters are two-way:
  - the text coming from the user is converted into Java Objects
  - the Java Objects coming from the program are converted into text
- Consider the Birthday field in the Address Book application (Create.xhtml):

```
<td><h:outputLabel value="#{bundle.CreateContactLabel_birthday}"
                              for="birthday" /></td>
       <td><h:inputText id="birthday"
                             value="#{contactController.selected.birthday}"
                             title="#{bundle.CreateContactTitle_birthday}" >
               <f:convertDateTime pattern="MM/dd/yyyy" />
            </h:inputText></td>
 <td><h:message for="birthday" /></td>
```

SEE: https://docs.oracle.com/javaee/7/tutorial/jsf-page-core001.htm#BNASV

## Built-in Converters

| Class in the javax.faces.convert Package | Converter ID |
|---|---|
| `BigDecimalConverter` | `javax.faces.BigDecimal` |
| `BigIntegerConverter` | `javax.faces.BigInteger` |
| `BooleanConverter` | `javax.faces.Boolean` |
| `ByteConverter` | `javax.faces.Byte` |
| `CharacterConverter` | `javax.faces.Character` |
| `DateTimeConverter` | `javax.faces.DateTime` |
| `DoubleConverter` | `javax.faces.Double` |
| `EnumConverter` | `javax.faces.Enum` |
| `FloatConverter` | `javax.faces.Float` |
| `IntegerConverter` | `javax.faces.Integer` |
| `LongConverter` | `javax.faces.Long` |
| `NumberConverter` | `javax.faces.Number` |
| `ShortConverter` | `javax.faces.Short` |

## Built-in Converters (cont’d)

- Built-in Converters are applied implicitly when an Expression Language expression is used (and therefor the datatype is known)

- DateTimeConverter has its own <f:convertDateTime> tag
  - As used in the Address Book example

- NumberConverter has its own <f:convertNumber> tag

## Custom Converters

Custom Converters implement the javax.faces.convert.Converter interface:

- public Object getAsObject(FacesContext *context*,
  UIComponent *component*, String *value*)
- public String getAsString(FacesContext *context*,
  UIComponent *component*, Object *value*)

## Explicit Converter Example

- Consider the color of our Sprites