Java EE Expression Language (EL)

Expression Language

- The Expression Language (also referred to as the EL), *provides an important mechanism for enabling the presentation layer (web pages) to communicate with the application logic (managed beans)*. The EL is used by both JavaServer Faces technology and JavaServer Pages (JSP) technology.

Expression Language

- You have allready used EL in previous excercises and examples:

```
<h:form>
    <h1>Welcome to greetings page</h1>
    <h:inputText id="inputName" value="#{greetingsBean.name}"></h:inputText><br/>
    <h:commandButton value="Say Hello">
    </h:commandButton><br/>
    <h:outputLabel id="outPutMessage" value="#{greetingsBean.sayGreeting}"></h:outputLabel>
</h:form>
```

Expression Language

- The EL allows page authors to use simple expressions to dynamically access data from JavaBeans components. For example, the test attribute of the following conditional tag is supplied with an EL expression that compares 0 with the number of items in the session-scoped bean named cart.

```
<c:if test="${sessionScope.cart.numberOfItems > 0}">
  ...
</c:if>
```

Expression Language

- JavaServer Faces technology uses the EL for the following functions:
  - Deferred and immediate evaluation of expressions
  - The ability to set as well as get data
  - The ability to invoke methods

Expression Language

- To summarize, the EL provides a way to use simple expressions to perform the following tasks:
  - Dynamically read application data stored in JavaBeans components, various data structures, and implicit objects
  - Dynamically write data, such as user input into forms, to JavaBeans components
  - Invoke arbitrary static and public methods
  - Dynamically perform arithmetic operations

Expression Language

- Immediate evaluation means that the expression is evaluated and the result returned as soon as the page is first rendered.
- Deferred evaluation means that the technology using the expression language can use its own machinery to evaluate the expression sometime later during the page's lifecycle, whenever it is appropriate to do so.

Expression Language

- Those expressions that are evaluated immediately use the ${} syntax.
- Expressions whose evaluation is deferred use the #{} syntax.

Expression Language

- Both rvalue and lvalue expressions can refer to the following objects and their properties or attributes:
  - JavaBeans components
  - Collections
  - Java SE enumerated types
  - Implicit objects

Implicit Objects

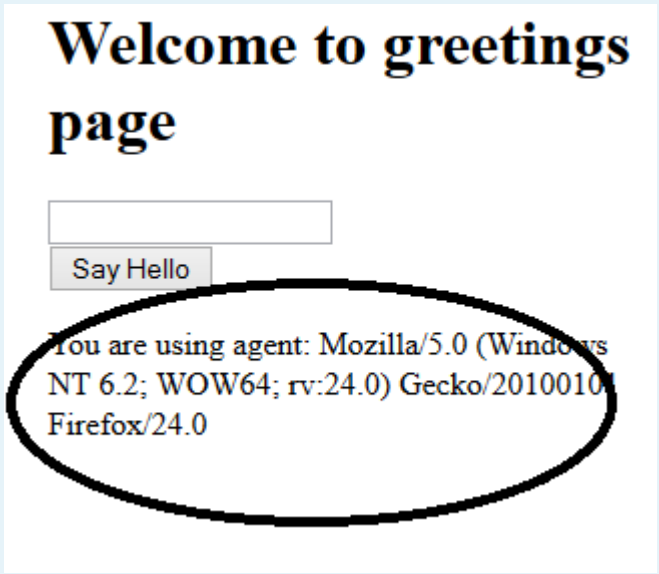
- There are few implicit objects (created by framework) you can use in EL:
- facesContext: The actual FacesContext instance.
- application: The ServletContext or PortletContext.
- session: The HttpSession or PortletSession.
- view: The current UIViewRoot for this view.
- request: The ServletRequest or PortletRequest.
- applicationScope: A Map of all application scope attributes.
- sessionScope: A Map of all session scope attributes.
- viewScope: A Map of all view scope attributes.
- requestScope: A Map of all request scope attributes.

Implicit Objects

- param: A Map of HTTP request parameters, containing only the first value for each name.
- paramValues: A Map of HTTP request parameters, yielding a String[] array of all values for a given name.
- header: A Map of HTTP header parameters, containing only the first value for each name.
- headerValues: A Map of HTTP header parameters, yielding a String[] array of all values for a given name.
- initParam: A Map of the initialization parameters of this web application.
- cookie: A Map of the cookie names and values of the current request.
- flash: A Map for forwarding objects to the next view.
- resource: A Map of application resources.
- component: The UIComponent instance being currently processed.

Example using Implicit objects

```
</h:head>
<h:body>
    <h:form>
        <h1>Welcome to greetings page</h1>
        <h:inputText id="inputName" value="#{greetingsBean.name}"></h:inputText><br/>
        <h:commandButton value="Say Hello">
        </h:commandButton><br/>
        <h:outputLabel id="outPutMessage" value="#{greetingsBean.sayGreeting}"></h:outputLabel>
        <p> You are using agent: #{header['User-Agent']}</p>
    </h:form>
</h:body>
```



Expression Language

- To refer to these objects, you write an expression using a variable that is the name of the object. The following expression references a managed bean called customer:

  ${customer}

Expression Language

- To refer to properties of a bean or an enum instance, items of a collection, or attributes of an implicit object, you use the . or [] notation.

Expression Language

- To refer to properties of a bean or an enum instance, items of a collection, or attributes of an implicit object, you use the . or [] notation.

Nested Beans Example

- The Managed Beans

```java
@ManagedBean(name="address")
@SessionScoped
public class AddressBean implements Serializable{

    private static final long serialVersionUID = 1L;

    private String city;

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }
    public AddressBean() {
    }
}
```

```java
@ManagedBean(name = "user")
@SessionScoped
public class UserBean implements Serializable{

 private static final long serialVersionUID = 1L;
    private String name = "";

    //When an UserBean instance is constructed, the value expression #{address} is
    //evaluated, and the result is passed to the setAddress method.
    @ManagedProperty(value="#{address}")
    private AddressBean address;

    public String getName() {
        return name;
    }
    public void setName(String newValue) {
        name = newValue;
    }
    public AddressBean getAddress() {
        return address;
    }
    // Must provide setter method for property injection
    public void setAddress(AddressBean address) {
        this.address = address;
    }
}
```

El Expression

- The '.' notation

```
<h:body>
     <h:form prependId="false">
     <h3>Please Enter:</h3>
     <table>
        <tr>
           <td>City:</td>
           <td><h:inputText value="#{user.address.city}" id="name"/></td>
        </tr>
     </table>
    <p><h:commandButton value="Welcome"/></p>
    <h:outputText value="#{user.address.city}"></h:outputText>
  </h:form>
</h:body>
```

EL Expression

- The '[]' notation

```
<h:body>
     <h:form prependId="false">
     <h3>Please Enter:</h3>
     <table>
        <tr>
           <td>City:</td>
           <td><h:inputText value="#{user['address']['city']}" id="name"/></td>
        </tr>
     </table>
    <p><h:commandButton value="Welcome"/></p>
      <h3>Welcome to #{user["address"]["city"]}!</h3>
  </h:form>
</h:body>
```

Method Expression

- Method Expression are used to invoke public non-static methods of managed beans. A method expression denotes an object and a method that can be applied to it.
- The basic method expression format as following:

```
<h:commandButton action="#{someBean.someMethod}"/>
```

Example

```java
@ManagedBean(name = "user")
@SessionScoped
public class UserBean implements Serializable{

 private static final long serialVersionUID = 1L;
    private String name = "";

    public String hello()
    {
        return "Hello There";
    }

    public String hello(String str1, String str2)
    {
        return "Hello " + str1 + " " + str2;
    }
```

```
  <h:body>
       <h:form prependId="false">
       <h3>Please Enter:</h3>
       <table>
          <tr>
             <td>City:</td>
             <td><h:inputText value="#{user['address']['city']}" id="name"/></td>
          </tr>
       </table>
      <p><h:commandButton value="Welcome"/></p>
      <h3>Welcome to #{user.hello()}!</h3>
      <h3>Welcome also #{user.hello('markus','erkki')}!</h3>
    </h:form>
  </h:body>
</html>
```

Collections, Arrays and Enums

- Using Collection Arrays or Enums is propably something you will use in your managed beans (for example you managed bean can retrieve a collection of data from DB)
- Collections are handled in particular way in JSF page using expression language.

Collections, Arrays and Enums

- Legal collection types are:
  - Array
  - List (ArrayList, LinkedList)
  - ResultSet
  - Result
  - DataModel

Collections and ListExample

```java
public class Product {
    private String productName;
    private double productPrice;

    public Product(String name, double price){
        this.productName = name;
        this.productPrice = price;
    }

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    public double getProductPrice() {
        return productPrice;
    }

    public void setProductPrice(double productPrice) {
        this.productPrice = productPrice;
    }

}
```

```java
@ManagedBean(name="items")
@SessionScoped
public class ListItems {


    private List<Product> products;
    public ListItems() {
        products = new ArrayList<>();
        products.add(new Product("Nexus",250.0));
        products.add(new Product("Lumia",260.0));
        products.add(new Product("iPhone S5",600.0));
        products.add(new Product("LG",234.50));
    }

    public List<Product> getProducts() {

        return products;
    }

    public void setProducts(List<Product> products) {
        this.products = products;
    }

}
```

Collections and ListExample

```
<h:form>
<h1>Here is the list of our products</h1>
<ul>
    <ui:repeat value="#{items.products}" var="it">
        <li> <h:outputText value="#{it.productName} #{it.productPrice}"></h:outputText> </li
    </ui:repeat>
</ul>
</h:form>
```

Here is the list of our products

- Nexus 250.0
- Lumia 260.0
- iPhone S5 600.0
- LG 234.5

Collections and Data Table

```
<h:dataTable value="#{items.products}" var="some"
             styleClass="order-table"
             headerClass="order-table-header"
             rowClasses="order-table-odd-row order-table-even-row">
    <h:column>
        <!--Column Header-->
        <f:facet name="header">Product</f:facet>
        <!--Column row-->
        #{some.productName}
    </h:column>
    <h:column>
        <!--Column Header-->
        <f:facet name="header">Price</f:facet>
        <!--Column row-->
        #{some.productPrice}
    </h:column>
</h:dataTable>
```

Product	Price
Nexus	250.0
Lumia	260.0
iPhone S5	600.0
LG	234.5