Expression Language

Objectives

What and Why use Expression Language?

How to write with the EL syntax?

Immediate evaluation

How to use scoped variables in EL expressions?

+ pageContext
+ requestScope
+ sessionScope
+ applicationScope

Implicit Variables in EL

Conditional evaluation

What is Unified EL?

- Unified EL is expression language for accessing objects in JSP
  - Provides access to JavaBeans objects and their properties
    - Get values, set values, access methods
  - Can access arrays, lists and maps
  - Example:

```
${someBean.someProperty}
```

History of the EL

- Starts from JSTL EL and SPEL
  - JSTL 1.0
- Standard part of JSP 2.0
  - JSP EL
- Problems with JSF (Faces 1.0)
  - JSP EL is weak for JSF
  - JSF EL is created
- Unification in UEL (JSP 2.1 and JSF 1.2)

FPT UNIVERSITY
DREAM OF INNOVATION

JSP and JavaBeans

Using JavaBeans in a JSP Page

<jsp:useBean>

- <jsp:useBean> lets you load in a JavaBean to be used in the JSP
  - Lets you exploit the reusability of Java classes
  - Syntax:

```
<jsp:useBean id="name" class="package.class"
scope="page | request | session | application" />
```

  - After having the JavaBean instance you can modify its properties with:
    - <jsp:setProperty>
    - <jsp:getProperty>

<jsp:useBean> and Scope

- The scope of the beans specifies where the bean should be stored
  - `page` scope – the bean is available to the current JSP page only
  - `request` scope – the bean is available to all pages that take part in the processing of the current request
  - `session` scope – the bean is stored in the client's session
  - `application` scope – the beans is stored globally, available to whole the application

<jsp:useBean> – Example

- Declaring a bean (scoped variable), available during the whole client's session

```
<jsp:useBean id="user" class="beans.User"
  scope="session" />
```

  - This is compiled to this code:

```
beans.User user =
  (beans.User) session.getAttribute("user");
if (user == null) {
  user = new beans.User();
  session.setAttribute("user", user);
}
```

<jsp:setProperty>

- <jsp:setProperty>
  - Assigns a value to given property of given bean

```
<jsp:setProperty name="user"
  property="firstName" value="Nakov" />
```

  - When used inside <jsp:useBean> is executed only when the new object was instantiated

```
<jsp:useBean id="user" class="beans.User"
    scope="page" />
  <jsp:setProperty name="user"
    property="firstName" value="Nakov" />
</jsp:useBean>
```

<jsp:getProperty>

- `<jsp:getProperty>`
  - Retrieves the value of a bean property
  - Converts it to a string
  - Inserts it into the output

```
<jsp:useBean id="user" ... />
...
User login:
<jsp:getProperty name="user" property="login" />
User home page:
<jsp:getProperty name="user" property="homePage" />
```

FPT UNIVERSITY
DREAM OF INNOVATION

Expression Language

Why Do We Need EL?

- EL simplifies the development
  - Consider we have the following bean:

```
<jsp:useBean id="someBean"
  class="somePackage.someClass"
  scope="request, session, or application"/>
```

  - We can simplify this expression this way:

```
<jsp:getProperty name="someBean"
  property="someProperty"/>
```

```
${someBean.someProperty}
```

EL and JavaBeans – Example

```
<%
  String name = "Svetlin Nakov";
  pageContext.setAttribute("name", name);
%>
Name: ${name} <br />

<%
  User user = new User(
    "snakov", "Svetlin", "Nakov");
  pageContext.setAttribute("user", user);
%>
User first name: ${user.firstName} <br />
User last name: ${user['lastName']} <br />
User login: ${user["login"]} <br />
```

Advantages of the EL

- Concise access to a scoped variables (in the `page`, `session`, `application`, etc.)
  - To output a scoped variable named `saleItem`, you can use:

```
${saleItem}
```

- Simple access to collection elements
  - To access an element of an array, `List`, or `Map`, you can use:

```
${collectionObject[indexOrKey]}
```

Advantages of the EL (2)

- Shorthand notation for bean properties
  - To output the `companyName` property of a scoped variable named `company`, you use:

```
${company.companyName}
```

  - To access the `firstName` property of the `president` property of a scoped variable named `company`, you use:

```
${company.president.firstName}
```

Advantages of the EL (3)

- Fast access to request parameters, cookies, headers and other request data

```
${param["username"]}
${cookie["dateOfLastVisit"]}
```

- Useful set of simple operators
  - +, -, *, /, <, >, ==, &&, ||, empty, ?:

```
${(2 + 5) * 3}
${visitors.totalCount - 1}
${empty userBean} → true / false
${empty userBean ? "N/A" : "user found"}
```

Advantages of the EL (4)

- Automatic type conversion
  - Data is automatically converted from and to `String` values
- Empty values instead of error messages
  - Missing values or `NullPointerException` result in empty strings, not thrown exceptions
  - This could be a problem, especially during the debugging

How Scoped Variables Are Accessed?

- Consider the following expression:

```
${someBean}
```

- The EL evaluator performs search for the key "`someBean`" in these contexts:
  - `pageContext` (`PageContext`)
  - `request` (`HttpServletRequest`)
  - `session` (`HttpSession`)
  - `application` (`ServletContext`)
- Search is done in the given order

How Bean Properties Are Accessed?

- Consider the following expression:

```
${customer.firstName}
```

  - It finds scoped variable of given name and outputs the specified bean property

- Equivalent to the following code:

```
<%@ page import="beans.Customer" %>
<% Customer customer = (Customer)
   pageContext.findAttribute("customer"); %>
<%= customer.getFirstName() %>
```

How Collections Are Accessed?

- Accessing collection entries:

```
${someCollection[entryName]}
${someCollection.entryName}
```

- For Array is equivalent to:
  - *theArray[index]*
- For List is equivalent to:
  - *theList.get(index)*
- For Map is equivalent to:
  - *theMap.get(keyName)*

Referencing Implicit Objects

- `pageContext` – the objects in the `PageContext`
  - `${pageContext.session.id}`
- `param` – request parameters
  - `${param.custID}`
- `header` – HTTP request headers
  - `${header["Accept-Encoding"]}`
- `cookie` – cookie objects (not cookie values)
  - `${cookie["userCookie"].value}`
- `initParam` – context initialization param
- `pageScope`, `requestScope`, `sessionScope`, `applicationScope` – directly accessing scopes
  - `${pageScope.user.firstName}`

EL Operators

- Arithmetic operators
  - `+ - * / div % mod`
- Relational operators
  - `== eq != ne < lt > gt <= le >= ge`
- Logical operators
  - `&& and || or ! not`
- Empty check operator
  - `empty <object>`
  - Returns `true` for null, empty string, empty array, empty list, empty map

Conditional Evaluation

- Evaluates `condition` and outputs either `expression1` or `expression2`

```
${condition ? expression1 : expression2}
```

- Problems:
  - Relatively weak
    - `c:if` and `c:choose` from JSTL are much more powerful
  - Tempts you to put business (processing) logic in the JSP page
    - Should be used for presentation logic only

Summary

1. **What's Unified Expression Language?**
2. **Why use Expression Language?**
3. **How to use Expression Language?**