Web Component Development with Servlet & JSP Technologies (EE 6)

Module-6: More View Facilities

Team Emertxe

ORACLE
Certified Expert
Java EE 6 Web Component Developer

EMERTXE

Objectives

Upon completion of this module, you should be able to:

- Understand and use the four scopes
- Understand and use the EL
- Recognize and use the EL implicit objects

Relevance

Discussion – The following questions are relevant to understanding what technologies are available for developing web applications and the limitations of those technologies:

- What lifetimes and accessibilities are necessary for variables in a web application?
- How can the EL be best used, and what facilities should it offer that have not been introduced yet?

Scopes

You saw in several earlier modules that servlets can store and access application and parameter data in several places. To review, the locations discussed so far are:

- The request object – Contains a map of attributes and is shared across forward calls. Its scope is bounded by a request from a single client.

- The session object – Contains a map of attributes shared across all calls from a single browser within a single session. Session duration may be bounded by time limits or by a login/logout type process.

Scopes

- The servlet’s initialization parameters – Is a map of read-only configuration data available to this servlet from the deployment descriptor.
- The application's initialization's parameters: Is a map of read only configuration data from the deployement descriptor and available to any servlet in the applicationt.
- The browser cookies: Is a map of cookie value set in the brrowser by the application.

EL Implicit Objects

- Page: Between local variables within a JSP only. Equivalent to local variables in a doXxx servlet method.
- Request: Between components cooperating in the execution of a single request from a single browser. Typically used to carry data from a controller servlet to the view JSP.
- Session: Between servlets and JSPs used during a single user session, across the different requests being made.
- Application: Between servlets and JSPs used during a single user session, across the different requests being made.

More Details About the Expression Language (EL)

Syntax Overview:

The syntax of EL in a JSP page is as follows: ${expr}

EL and Scopes

The following example retrieves the firstName property from the bean cust located in the session scope:

```
${sessionScope.cust.firstName}
```

EL Implicit Object

- PageContext: The PageContext object
- PageScope: A map containing page-scoped attributes and their values
- RequestScope: A map containing request-scoped attributes and their values
- SessionScope: A map containing session-scoped values attributes and their values
- ApplicationScope: A map containing application-scoped values attributes and their values

EL Implicit Object

- Param: A map containing request parameters and their corresponding string Values
- ParamValues: A map containing request parameters and their corresponding string arrays
- Header: A map containing header names and single string values
- HeaderValues: A map containing header names and their corresponding string arrays
- Cookie: A map containing cookie names and their values
- InitParam: A map of the servlet’s init parameters

Example of EL Implicit Objects

Using these implicit objects, for example, a page can access a request parameter called username using the following EL expression:

```
${param.username}
```

The Dot Operator In EL

If the request scope contains an object with the attribute name pet which is a Dog JavaBean, and that Dog bean has a getOwner() method, and that method returns a Human object, which in turn has a getName() method that returns a String, then the following EL syntax would be legitmate and would access the name of the dog's owner:

```
${requestScope.pet.owner.name}
```

Array Access Syntax With EL

If a bean returns an array, an element in the array can be accessed by providing its index:

```
${paramValues.fruit[2]}
```

As with Java, the dot operator may be used to access elements of an object after it has been selected from an array. So, if the Dog bean mentioned above happened to return an array of two Human objects from a method getOwnerFamilyMembers() then this would display the name of the first owner:

```
${requestScope.pet.ownerFamilyMembers[0].name}
```

Array Access Syntax With EL

In addition to accessing elements of arrays in this way, the EL array syntax notation behaves as an associative array. So, an attribute name may be used as the array subscript if preferred:

```
${requestScope[“pet”].owner[“name”]}
```

EL and Errors

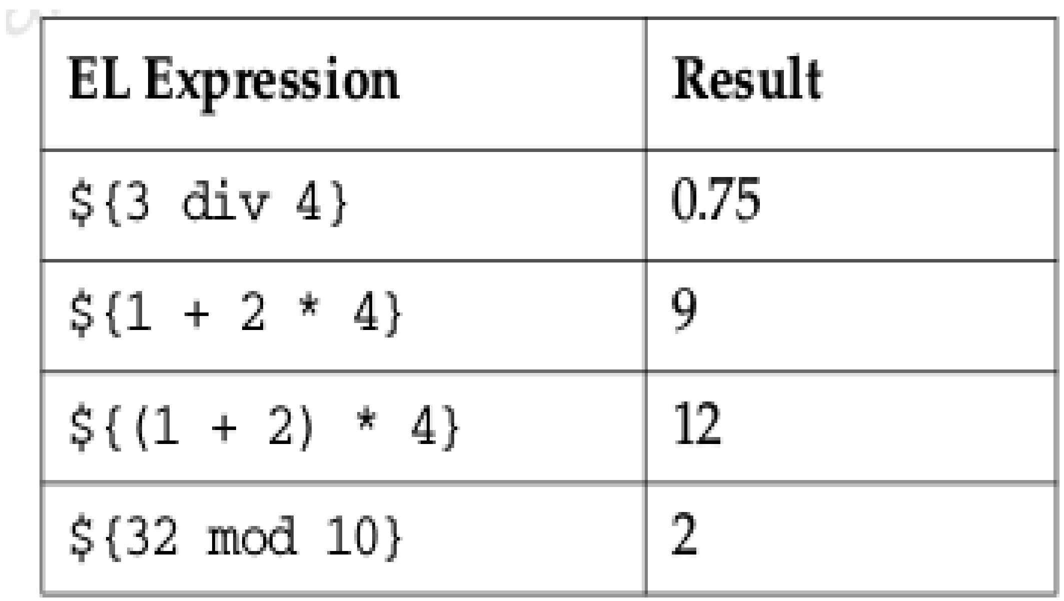
EL silently hides errors that arise from null references or not-found variables. That is, such expressions translate to empty strings. This is deliberate, as it ensures that errors do not mess up the pages that the user sees. Of course, the information on the page might be incomplete.

Note – If an EL expression refers to a non-existent attribute, such as ${exists.notFound} then EL will cause an exception, and an unhelpful, and potentially insecure, stack trace will be shown to the user.

EL Arithmetic Operators

Arithmetic Operation	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/ and div
Remainder	% and mod

Some Example EL Arithmetic Operations



EL Expression	Result
`${3 div 4}`	0.75
`${1 + 2 * 4}`	9
`${(1 + 2) * 4}`	12
`${32 mod 10}`	2

EL Comparison Operators

Equals:	== and eq
Not equals:	!= and ne
Less than:	< and lt
Greater than:	> and gt
Less than or equal:	<= and le
Greater than or equal:	>= and ge

EL Logical Operators

And	&& and and
Or	|| and or
Not	! and not
Test if an array or list is empty	empty

Configuring the JSP Environment

This section outlines the deployment descriptor configuration for the JSP environment. You can modify the behavior of JSP pages in the application using the jsp-config tag in the web.xml deployment descriptor.

Configuring the JSP Environment

```
<web-app>
<jsp-config>
<jsp-property-group>
<url-pattern>/scripting_off/*</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-property-group>

<jsp-property-group>
<url-pattern>/EL_off/*</url-pattern>
<el-ignored>true</el-ignored>
</jsp-property-group>
</jsp-config>
</web-app>
```

Configuring the JSP Environment

Within the jsp-config tag, you can declare zero or more jsp-property-group elements. Within a jsp-property-group, you declare a url-pattern that specifies the files that belong to this group.

The scripting-invalid tag is used to turn off scripting within the JSP pages in the group. If set to true, the JSP pages in the group are no prohibited from containing scripting code. If a JSP page contains scripting, an error will result.

Configuring the JSP Environment

The el-ignored tag is used to disable EL in the JSP pages in the group. By default, el-ignored has the value false, indicating that EL is executed. If set to true, any EL on the JSP pages in the group is ignored. This may be useful when pages are used from an older project and those pages use EL-like text that is intended for literal presentation.

Custom Tag

- When the tag library mechanism was introduced, tags were called “custom tags,” as each developer was expected to create their own. However, a popular set of tags was rapidly adopted as the “standard tags” (or JSTL).

- For this reason, you’ll often here tags referred to as both custom, and standard, sometimes in the same sentence. Strictly speaking, there is now a set of standard tags, and others are custom tags. Both use the same custom tags mechanism for their implementation and execution.

Example of JSP Custom Tag

For creating any custom tag, we need to follow following steps:

- Create the Tag handler class and perform action at the start or at the end of the tag.
- Create the Tag Library Descriptor (TLD) file and define tags
- Create the JSP file that uses the Custom tag defined in the TLD file

JSTL Tags

JSTL if Tag

```
<%-- Report any errors (if any) --%>
<c:if test=”${not empty errorMsgs}”>
<p>
<font color=’red’>Please correct the following errors:
<ul>
<c:forEach var=”message” items=”${errorMsgs}”>
<li>${message}</li>
</c:forEach>
</ul>
</font>
</p>
</c:if>
```

JSTL Tags

JSTL forEach Tag

```
<c:forEach begin=”1” end=”10”>
<!-- the repeated HTML code here -->
</c:forEach>
```

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe

https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

Thank You