JakEE JSP Expression Language

In this session, you will learn to:

- Explain how to use script expressions in JSP
- Describe the implicit objects used in EL
- Describe various operators used in EL
- Explain how to create static method and tag library descriptor using EL
- Explain how to access EL functions within JSP
- Explain the concept of boxing and unboxing
- Explain how to coerce a value to string or number type

image-133.png

Summary

- EL is simple and robust. It can handle both expressions and literals, which are constants and are assigned some memory location.
- EL is a great help to the page authors in accessing and manipulating the application data without mastering the complexities of the programming language such as Java and JavaScript.
- JSP implicit objects are a standard set of classes. The user creates an instance of an implicit object to use available methods and variables.
- Operators are used to perform different arithmetic, relational, and logical operations. Dot operator (`.`) or `[]` is used to access value of a variable. Various operators used in Expression Language are arithmetic operators, relational operators, logical operators, empty operators, and dot operators.
- In Expression language the static java methods can be called within the EL expression. To access the function using EL, the function must be implemented as a static function in a java class.
- The TLD file uses XML syntax to map the name of functions defined in a class with EL. Setting the value of the `<el-ignored>` element in the deployment descriptor can explicitly change the default mode.
- The accessing of the function created in a TLD file using a JSP file is possible by importing the TLD file using the `taglib` directive.
- Coercion means that the parameters are converted to the appropriate objects or primitives automatically. Coercion is an implicit type conversion.

image-136.png

Expression Language ***Superior in every Way***

Primary feature of the JSP Technology
Clean Syntax `${El expression}`

notables

Means of access to data in implicit objects is just better than other available options

- params -> `${param.name}`
- header -> `${header["host"]}`
- headerValues -> `${cookie.name.value}`
- pageScope -> `${pageScope.book}`
- requestScope -> `${requestScope.student.name}`
  You get the picture

Operators

- Since data is mapped from name to value, the access operations are what you would logically expect `bla.bla` or `bla["bla"]`
- Whether the value is null or empty can be determined using the empty operator, which is a prefix operation.
- The empty operator returns true if the string is empty. The string is said to be empty if it contains no character. If the string is not empty, then empty operator returns false.

Other Stuff

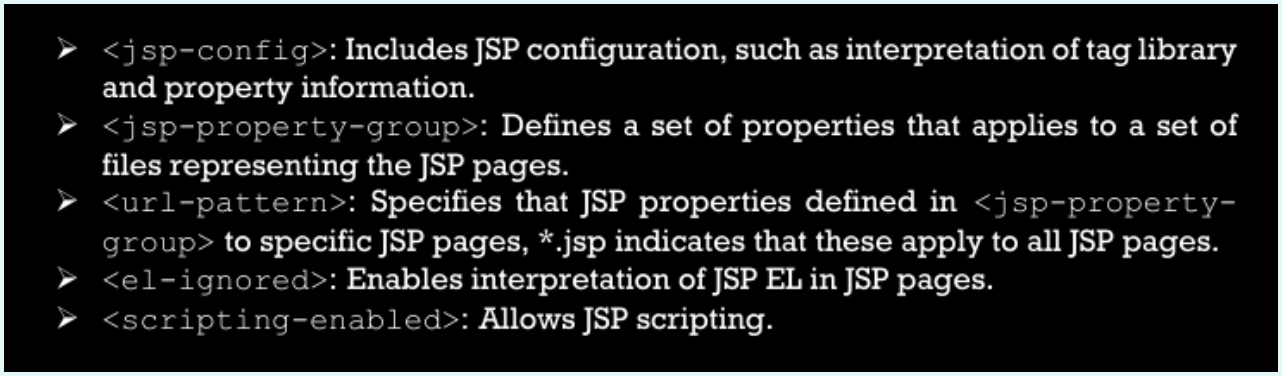
- The default mode for JSP version 1.2 technology or before is to ignore EL expressions. Starting with JSP version 2.0, the default mode is to evaluate EL expressions.
- The deployment descriptor can be changed with `<el-ignored>false</el-ignored>`
  - 

image-134.png

Functions using EL

needs

- Tag Library Descriptor in WEB-INF with
  - name to call
  - uri
- static function defined in a java class
- `<%@ taglib uri="http://example.com/functions" prefix="myfns" %>` in jsp

Coercion

notables

- Automatic conversion of a data from one data type to another data type within an expression is called Coercion
- Difference between type conversion and type coercion is conersion is explicit and coercion is implicit
- Boxing converts values of primitive type to corresponding values of reference type.
- Unboxing converts values of reference type to corresponding values of primitive type.

12.5.4 Coercion to Number

The rule to coerce a value to number type is as follows:

- If `A` is `null` or `""`, return `0`

A is character and is converted to short, you apply following rules:

- If `A` is `Boolean`, return `error`
- If `A` is number type, return `A`

A is number, coerce occurs quietly to type N using following algorithms:

- If `N` is `BigInteger`
- If `A` is `BigDecimal`, return `A.toBigInteger()`
- Otherwise, return `BigInteger.valueOf(A.longValue())`
- If `N` is `BigDecimal`
- If `A` is a `BigInteger`, return `new BigDecimal(A)`
- Otherwise, return `new BigDecimal(A.doubleValue())`

- If `N` is `Byte`, return `new Byte(A.byteValue())`
- If `N` is `Short`, return `new Short(A.shortValue())`
- If `N` is `Integer`, return `new Integer(A.integerValue())`
- If `N` is `Long`, return `new Long(A.longValue())`
- If `N` is `Float`, return `new Float(A.floatValue())`
- If `N` is `Double`, return `new Double(A.doubleValue())`
- Otherwise return `error`

image-135.png

Checkup

- C
- *B*
- *B*
- B
- *B*

- B - Read lil bro