Java Server Pages

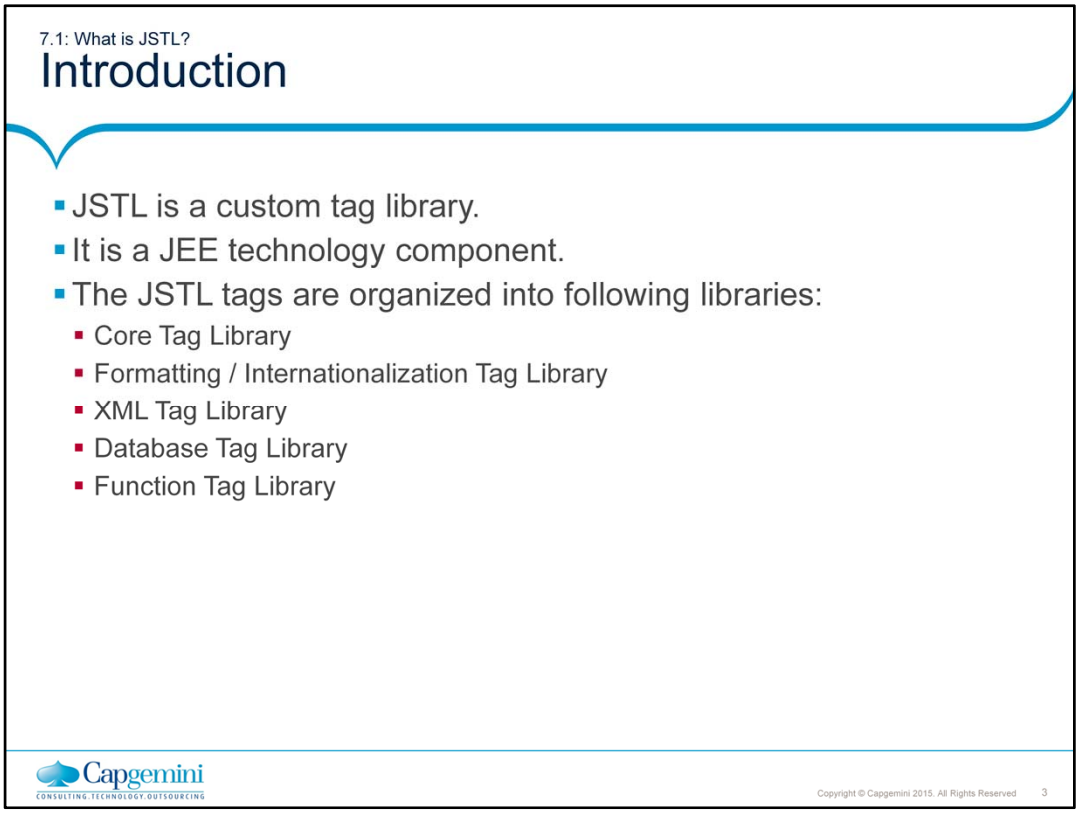
Lesson 7: JSP Standard Tag Library (JSTL)

Lesson Objectives

- In this lesson, you will learn:
  - What is JSTL?
  - Why JSTL?
  - Using Expression Language
    - EL Capabilities
    - Accessing Scoped Variables
    - Accessing Bean Properties
    - Accessing Collections
    - Accessing Implicit Objects
    - Using Lambda in EL
    - EL Operators
  - Using JSTL
    - Working with Core Tags

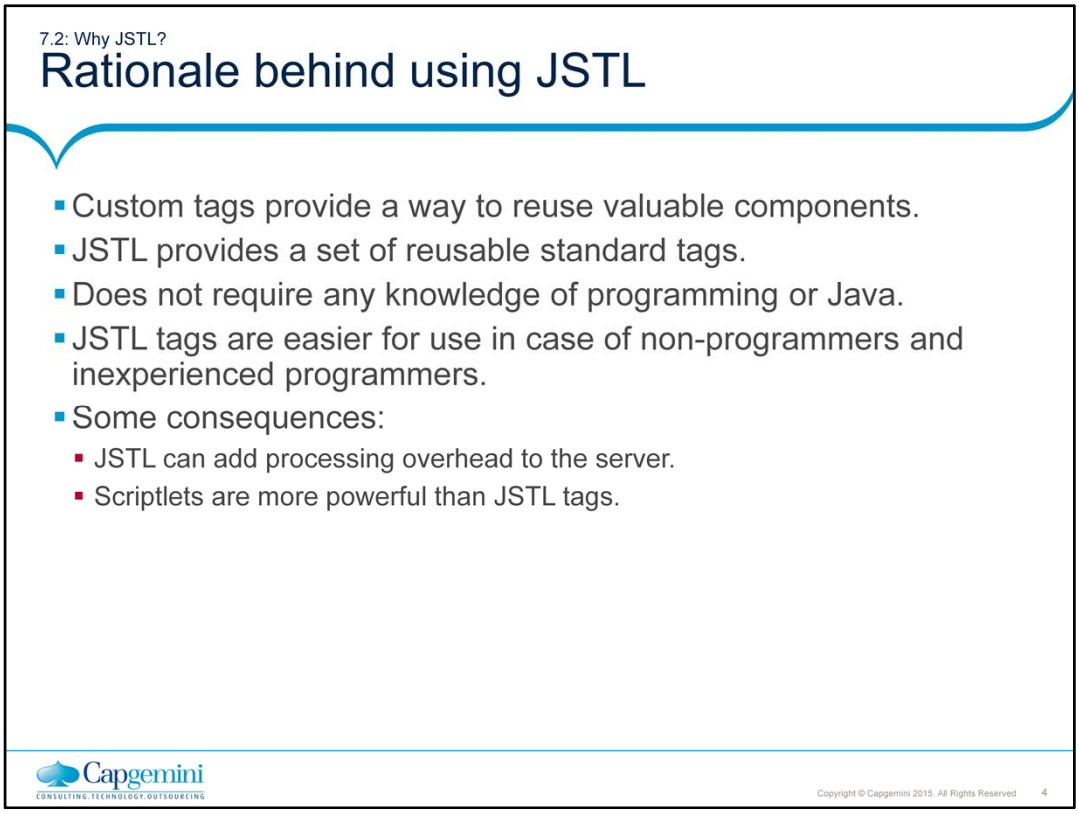
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 2



What is JSTL?

- **JSTL (JSP Standard Tag Libraries)** is a collection of JSP custom tags.
- The goal of JSTL, as described in the specification, is to help simplify JavaServer Pages page authors' lives.
- To achieve this goal, JSTL has provided custom tags for many common JSP page authoring tasks that require scripting statements to manipulate server side dynamic data.
- JSTL has tags such as iterators and conditionals for handling flow control, tags for manipulating XML documents, internationalization tags, tags for accessing databases using SQL, and commonly used functions.
- JSTL offers tags through following libraries:
  - **core:** Basic scripting functions
  - **xml:** XML processing
  - **fmt:** Internationalization of formatting
  - **sql:** Data base accessing
  - **fn:** JSTL functions

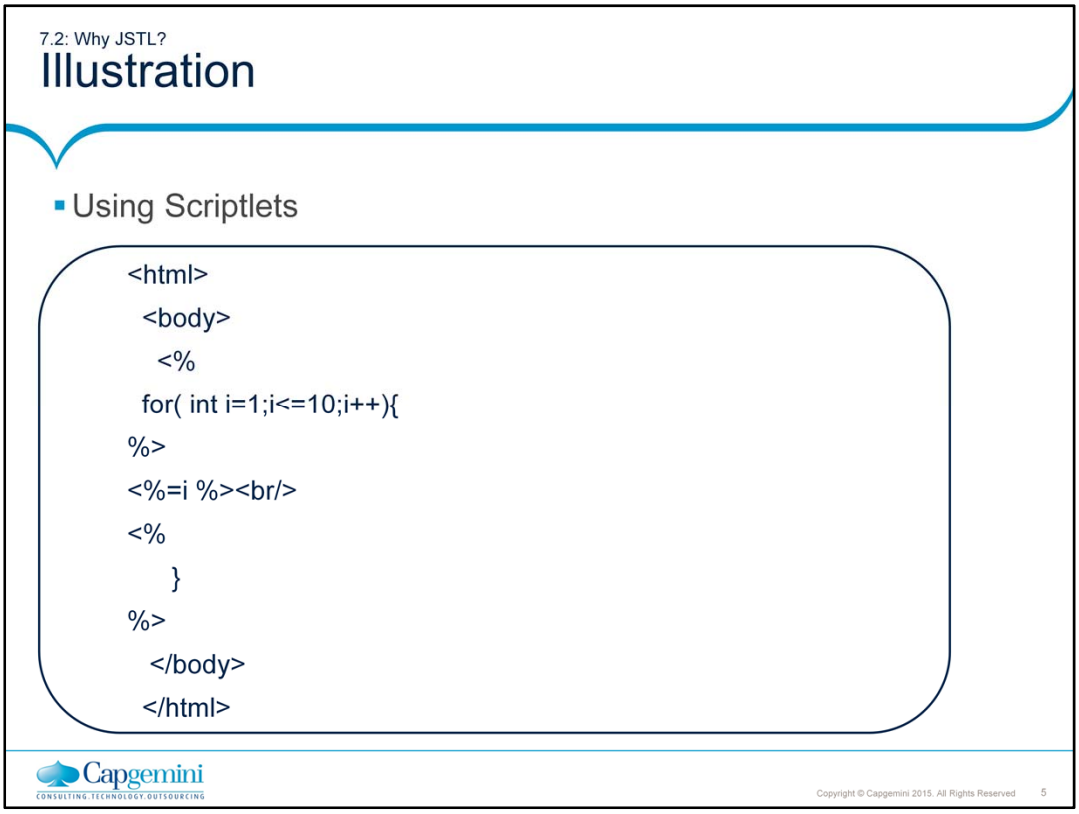


Why JSTL?

- Scriptlets are hard to read and hard to maintain. JSP was improved throughout its history, so that even people who do not know Java well (or have a limited knowledge of it) can make presentation pages.
- JSTL can help avoiding the use of scripting elements in JSP pages. As a matter of practice, a JSP page should be a view page, it should not have Java code embedded in **scriptlets**. This aids in separation of concerns.
- Custom tags provide a way to reuse valuable components.
- JSTL provides a set of *reusable* standard tags.
- JSTL tags can be used easily and effectively by non-programmers and inexperienced programmers. This is because they do not require any knowledge of programming or Java.
- Since JSTL tags are XML, they cleanly and uniformly blend into a page's HTML markup tags.

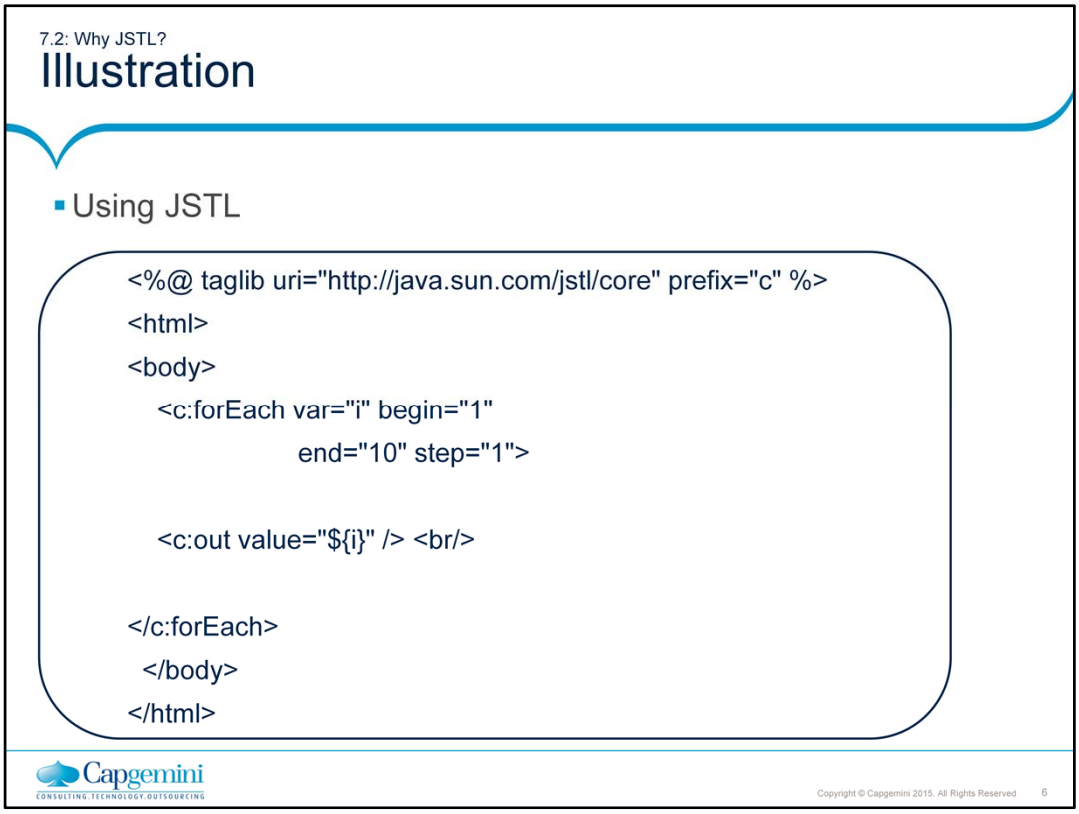
JSTL Consequences:

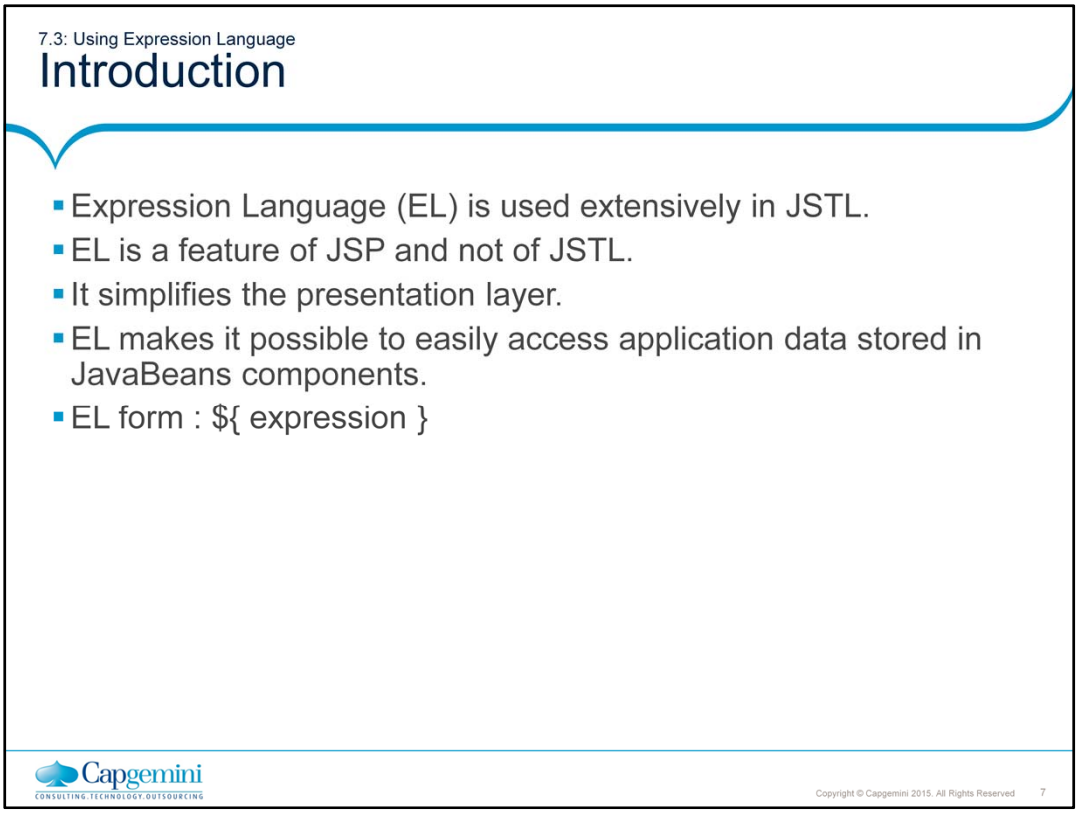
1. JSTL can add processing overhead to the server.
    - Java code embedded in scriptlet tags is pretty much just copied into the resulting servlet. JSTL tags, on the other hand, cause much more code to be added to the servlet.
2. Scriptlets are more powerful than JSTL tags.
    - Although JSTL provides a powerful set of reusable libraries, it cannot do everything that Java code can do. It is designed to facilitate scripting the presentation code.



Why JSTL?

- The above slide illustrates the JSP page with scriptlets and JSP page with JSTL. The JSP page with scriptlets contains the Java code embedded within JSP scripting tags. This requires an expert in Java programming.
- On the other hand, the JSP page with JSTL contains only tags and no Java code. So page authors (Non-Java programmers) can also develop the JSP pages using JSTL. The above example uses “ForEach” tag from core JSTL tag library which replaces the Java for loop.


7.2: Why JSTL?
Illustration
Using JSTL
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<html>
<body>
<c:forEach var="i" begin="1"
end="10" step="1">
<c:out value="${i}" />
</c:forEach>
</body>
</html>
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 6

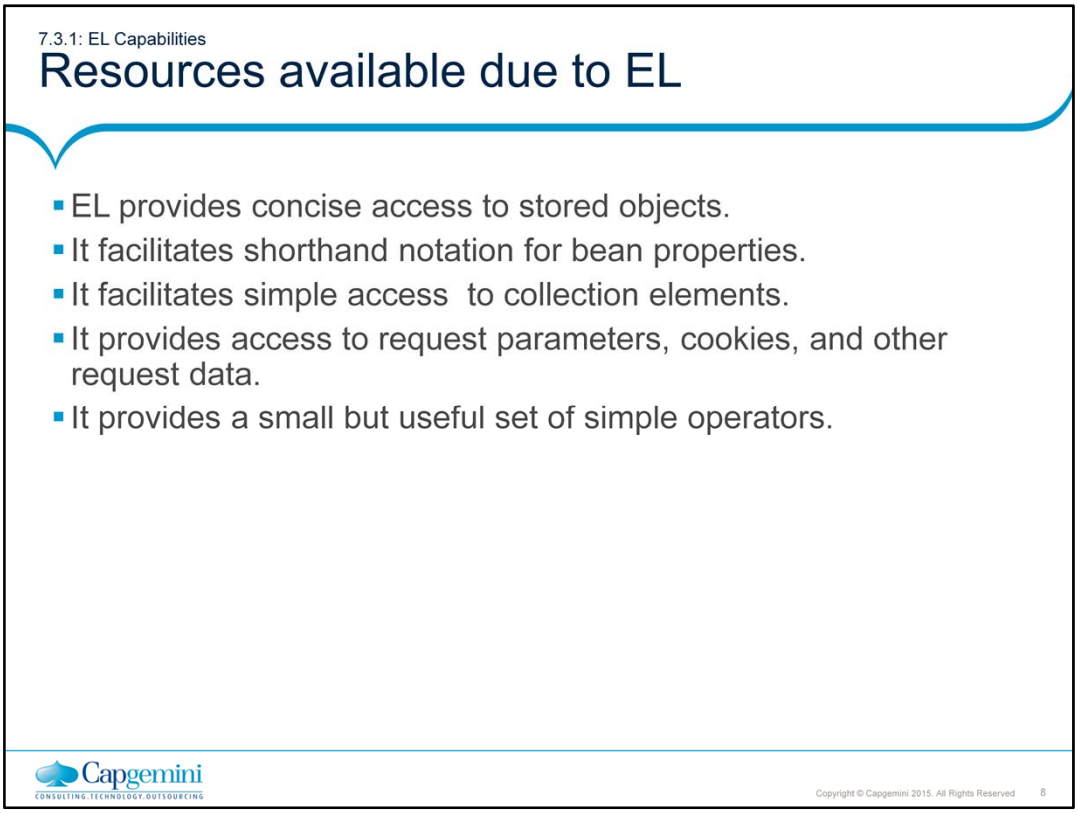


Using Expression Language:

- Expression Language is introduced with JSP 2.0, and many more capabilities are added with JSP 2.1.
- Expression Language is used with JSTL to simplify the presentation layer.
- Expression Language replaces the action tags <jsp:useBean> <jsp:getProperty> used to access Java beans properties with short and readable expressions.
- Express Language uses the form **${expr}** to access and specify an expression.

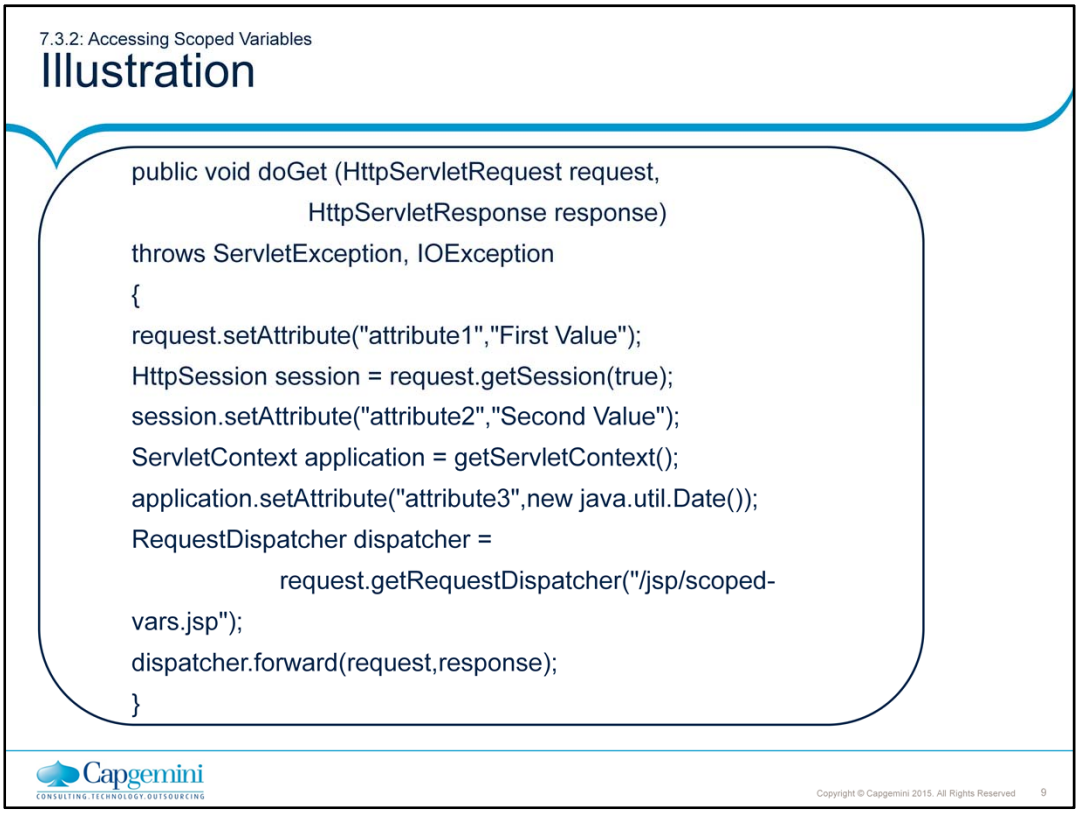
Example:

- ${rs.rowCount}, ${x+y}



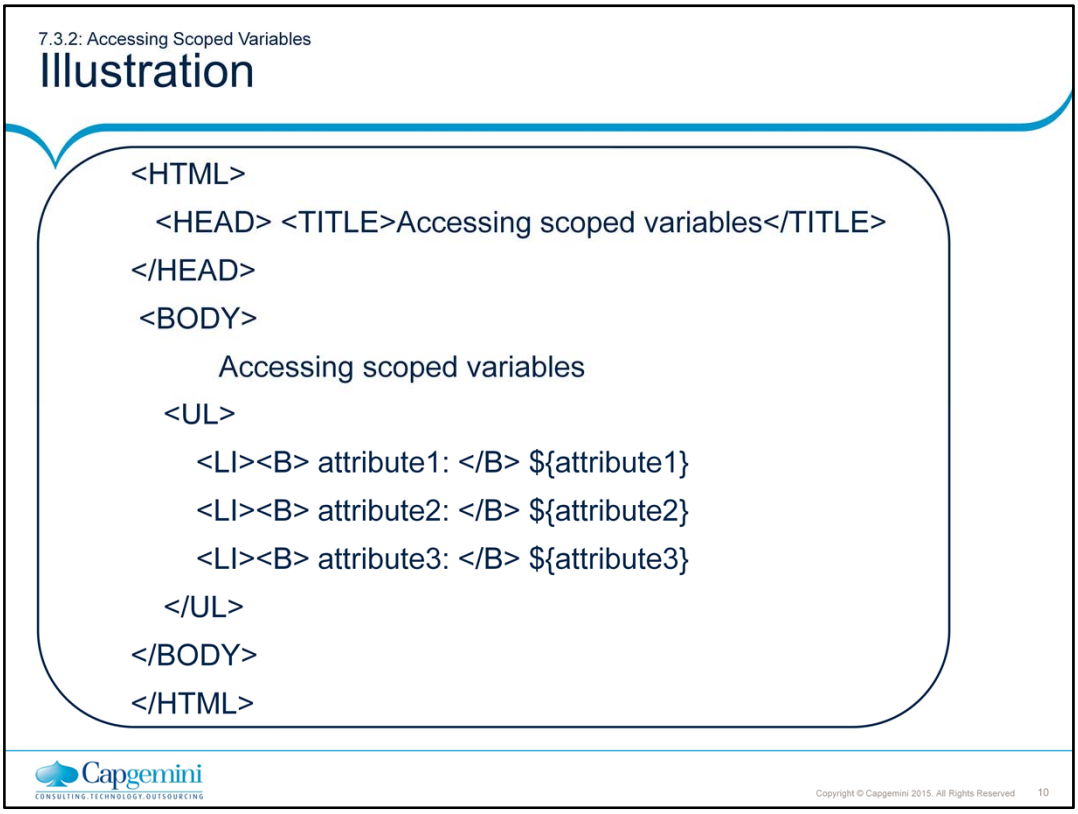
Expression Language Capabilities:

- EL provides concise access to stored objects: to output a “scoped variable” named saleItem, use ${saleItem}
- It facilitates shorthand notation for bean properties: To output the companyName property of a scoped variable named company, use ${company.companyName}.
- It facilitates simple access to collection elements: To access an element of an array, List or Map, use ${variable[indexorKey]}
- It provides access to request parameters, cookies, and other request data: To access the standard types of request data, we can use one of the implicit objects mentioned in first lesson.
- It provides a small but useful set of simple operators.



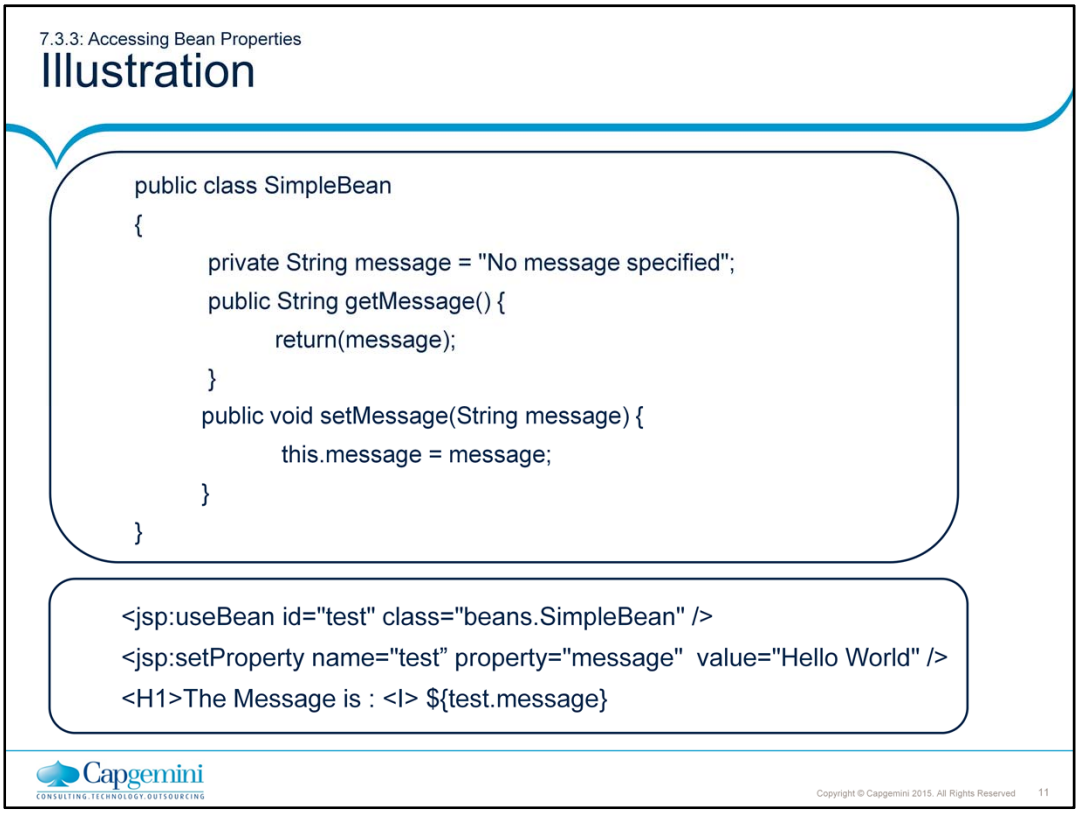
Accessing Scoped Variables:

- The code in the above slide defines different variables in different scopes and sets the values to these variables. There are various scopes like request, session, and application in which variables are set.
- Furthermore, this servlet forwards request to **scoped-vars.jsp** page. This jsp page uses expression language to retrieve these variables. The code snippet for that is given in the next slide.

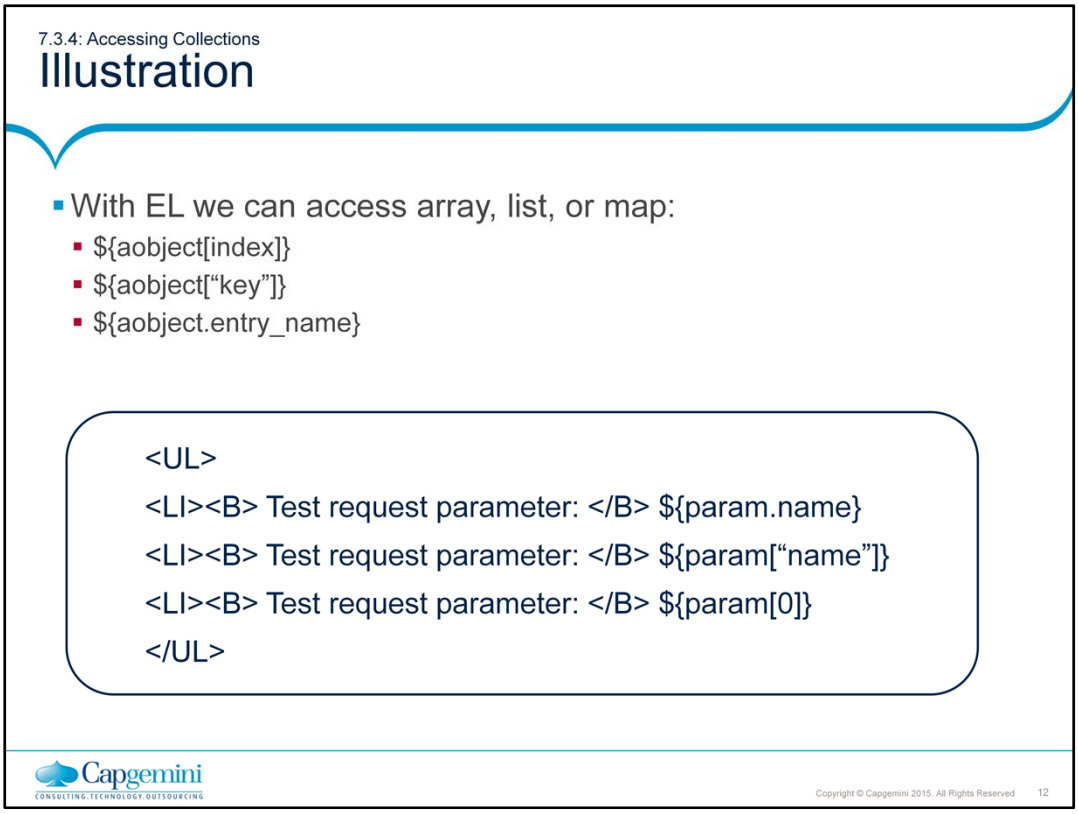


Accessing Scoped Variables:

- The above slide shows the code from **scoped-vars.jsp** page content. By using expression language the scoped variables defined in the previous servlet are accessed here.
- **Example:**
  ${attribute1} retrieves the value of scoped variable “attribute1”.



Accessing Bean Properties:
The illustration in the above slide is accessing the "**SimpleBean**" bean property "**message**" from JSP page using EL statement ${test.message}.



Accessing Collections:

- Using EL, we can access the collection elements, as well. The illustration is given in the above slide.
- The EL allows accessing collections using array notation.
  For example: If **AttributeName** is scoped variable and refers to an array, list or map, then we can access the collection with the following:

```
${AttributeName[entryname]}
```

- If the scoped variable is an array and entryname is the index, then an value is obtained with **array[index]**.
  For example: If CustomerNames refers to an array of strings, then the following entry will output first entry:

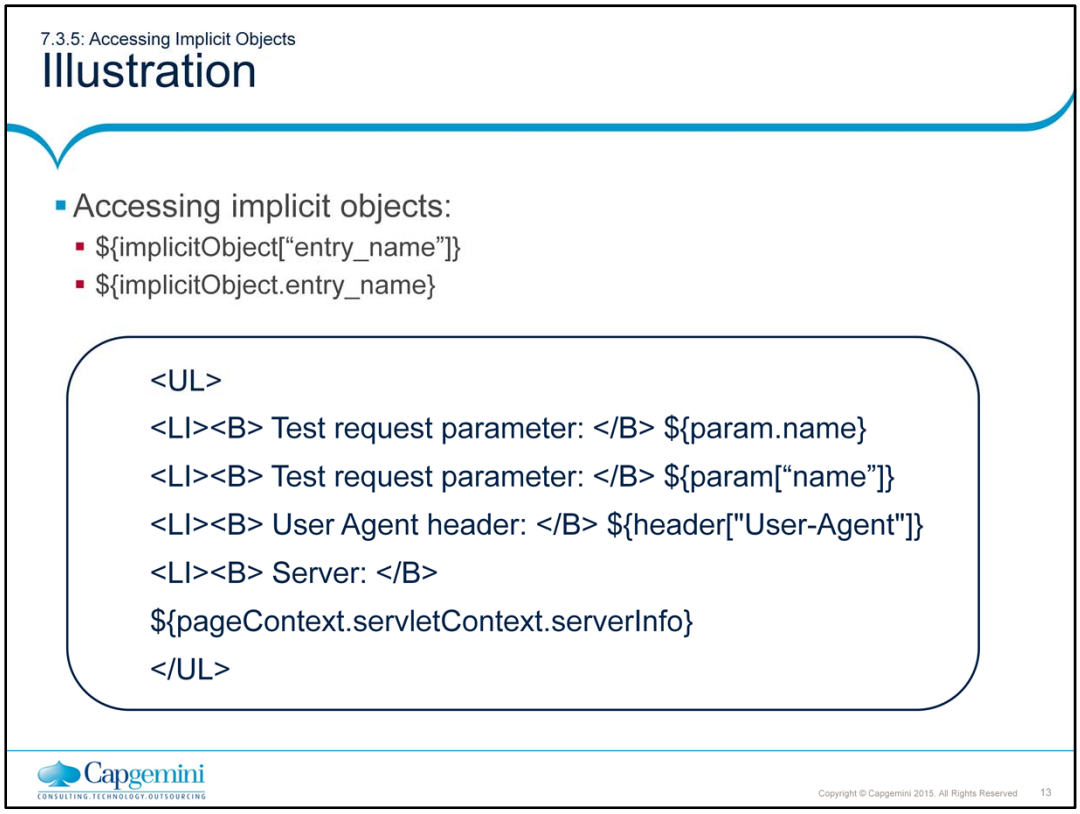
```
${CustomerNames[0]}
```

Similarly, the entries from a List can be output, as well.

- For Map, entry name is key and value is obtained with Map.get(key). In EL, the following example shows how to retrieve value from a map.

```
${stateCapitals["Maryland"]}
```

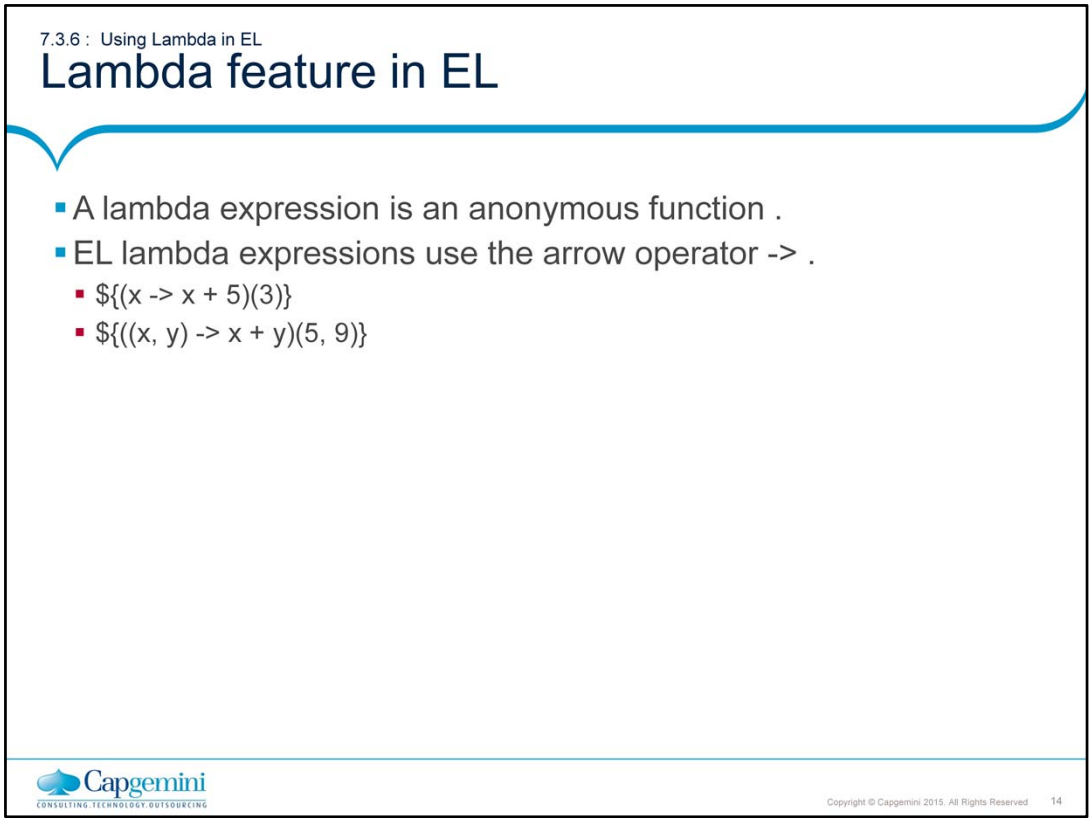
- The above example will return Annapolis.



Accessing Implicit Object:

The implicit objects provided by an JSP page can be used as follows:

- **PageContext:** For example, ${pageContext.session.id} returns the current session ID.
- **Param and paramValues:** These objects allows to use the primary request parameter value (param) or array of parameter values (paramValues).
  For example, ${param.custID} returns the value of custID request parameter, or null if it does not exist.
- **header and headerValues:** These access the HTTP request header values.
  For example: ${header.accept} accesses the accept header
- **initParam:** It allows to access context initialization parameters.
  For example: ${initParam.defaultColor} accesses init parameter called defaultColor.
- **pageScope, requestScope, sessionscope, applicationScope :** These objects restrict the locations where system looks up for scoped variables. For Example, ${name} will search for name in all scopes, whereas ${requestScope.name} will search for name only in HttpServletRequest.



A lambda expression is an anonymous function that, typically, is passed as an argument to a higher-order function (such as a Java method). In the most general sense, lambda expressions are a list of parameter names (or some placeholder if the function has no parameters), followed by some type of operator, and finally the function body.
The primary difference between the Java 8 and EL lambda expressions is that in Java the body of a lambda expression can contain anything that's legal in a Java method, whereas in EL the body of a lambda expression is another EL expression. EL lambda expressions use the arrow operator -> to separate the expression parameters on the left side from the expression in the right side. Also, again as with Java lambda expressions, the parentheses around the expression parameter are optional if there is exactly one parameter.${(x -> x + 5)(3)}${((x, y) -> x + y)(5, 9)}

7.3.7: EL Operators

Concept of EL Operators

- Arithmetic:
  - +, - , *, / and div, % and mod, - (unary)
- String Concatenation:
  - +=
- Logical:
  - and, &&, or, ||, not, !
- Relational:
  - ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le
- Empty:
  - null or empty

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 15

EL Operators:
The JSP expression language provides the following operators:
Arithmetic: +, - (binary), *, / and div, % and mod, - (unary).
String concatenation: +=.
Logical: and, &&, or, ||, not, !.
Relational: ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. Comparisons can be made against other values or against Boolean, string, integer, or floating-point literals.
Empty: The empty operator is a prefix operation that can be used to determine whether a value is null or empty.
Conditional: A ? B : C. Evaluate B or C, depending on the result of the evaluation of A.
Lambda expression: ->, the arrow token.
Assignment: =.
Semicolon: ;.

The precedence of operators highest to lowest, left to right is as follows:
The precedence of operators, highest to lowest, left to right, is as follows:
[] .
() (used to change the precedence of operators)
- (unary) not ! empty
* / div % mod
+ - (binary)
+=
<> <= >= lt gt le ge
== != eq ne
&& and
|| or
? :
->
=
;

7.3.7: EL Operators

Concept of EL Operators

- Conditional:
  - A ? B : C
- Lambda expression:
  - ->
- Assignment:
  - =
- Semicolon:
  - ;

EL Operators:
The JSP expression language provides the following operators:
Arithmetic: +, - (binary), *, / and div, % and mod, - (unary).
String concatenation: +=.
Logical: and, &&, or, ||, not, !.
Relational: ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. Comparisons can be made against other values or against Boolean, string, integer, or floating-point literals.
Empty: The empty operator is a prefix operation that can be used to determine whether a value is null or empty.
Conditional: A ? B : C. Evaluate B or C, depending on the result of the evaluation of A.
Lambda expression: ->, the arrow token.
Assignment: =.
Semicolon: ;.

The precedence of operators highest to lowest, left to right is as follows:
The precedence of operators, highest to lowest, left to right, is as follows:
[] .
() (used to change the precedence of operators)
- (unary) not ! empty
* / div % mod
+ - (binary)
+=
<> <= >= lt gt le ge
== != eq ne
&& and
|| or
? :
->
=
;

7.3.7: EL Operators

Illustration

EL Expression	Result
${1> (4/2)}	FALSE
${4.0>= 3}	TRUE
${100.0 == 100}	TRUE
${(10*10) ne 100}	FALSE
${'a' > 'b'}	FALSE
${'hip' lt 'hit'}	TRUE
${4> 3}	TRUE
${1.2E4 + 1.4}	12001.4
${3 div 4}	0.75
${10 mod 4}	2
${((x, y) -> x + y)(3, 5.5)}	8.5
${[1,2,3,4].stream().sum()}	10
${[1,3,5,2].stream().sorted().toList()}	[1, 2, 3, 5]
${!empty param.Add}	False if the request parameter named Add is null or an empty string

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 19

Note : For the following example in JSP page java.util.stream must be imported which is included in Java8:
[1,2,3,4].stream().sum()
[1,3,5,2].stream().sorted().toList()

Java.util.stream package contains the classes to support functional-style operations on streams of elements, such as map-reduce transformations on collections.

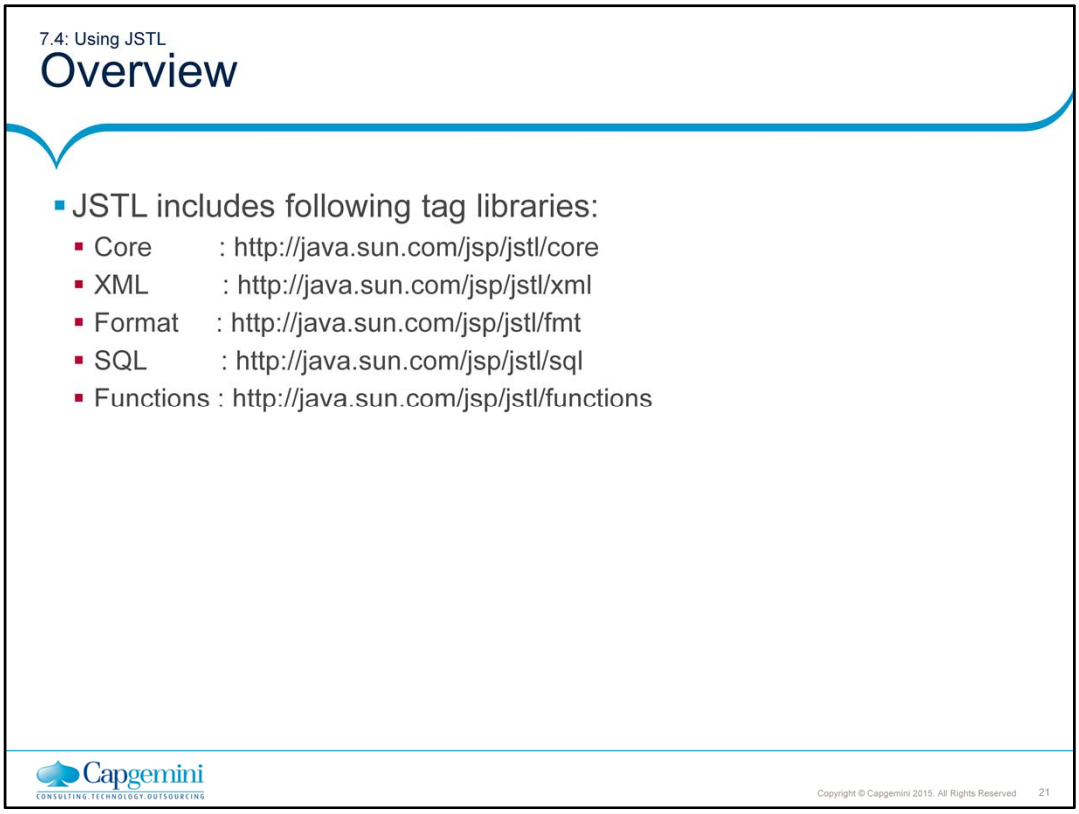
Demo

- Accessing bean properties
  - Refer JSP Pages in object folder
- Accessing collections
  - Refer JSP Pages in collection folder
- Accessing Different Scopes
  - Refer JSP Pages in scopes folder
- Using EL Operators
  - Refer JSP Pages in operator folder

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

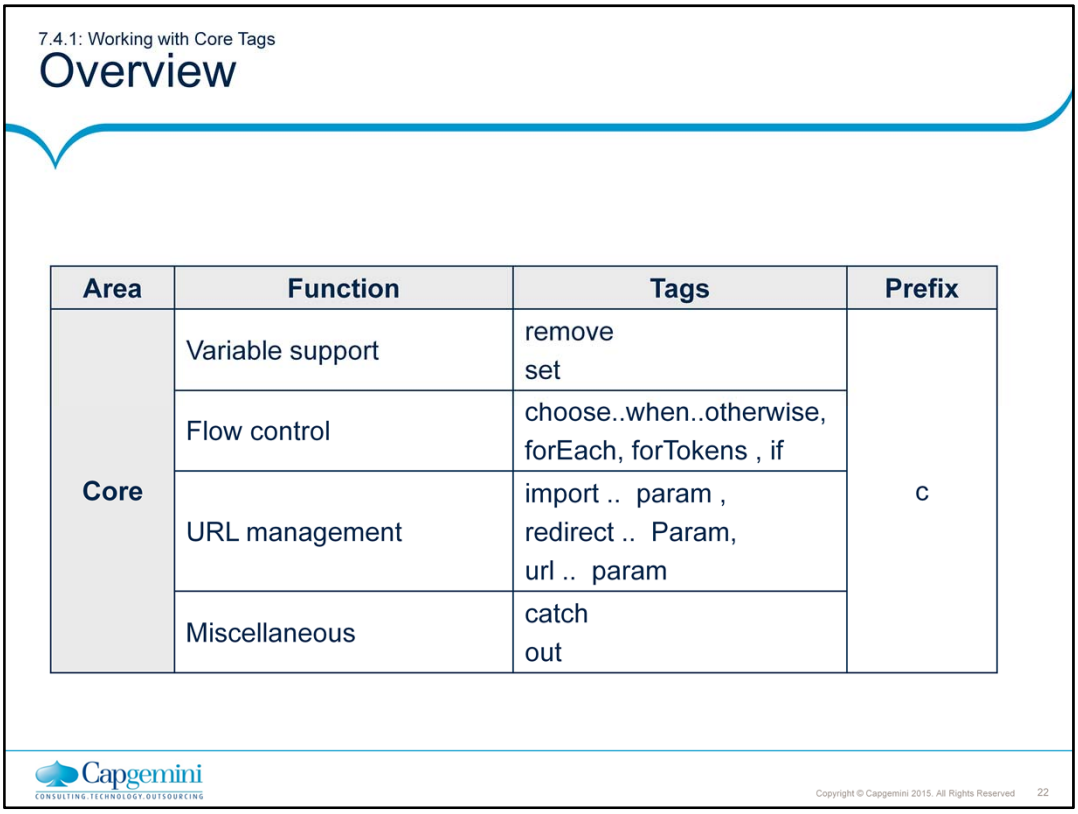
 20

Deploy web application **Lesson7-DemoEL** and show demo by executing each of the above JSP pages in corresponding folder.



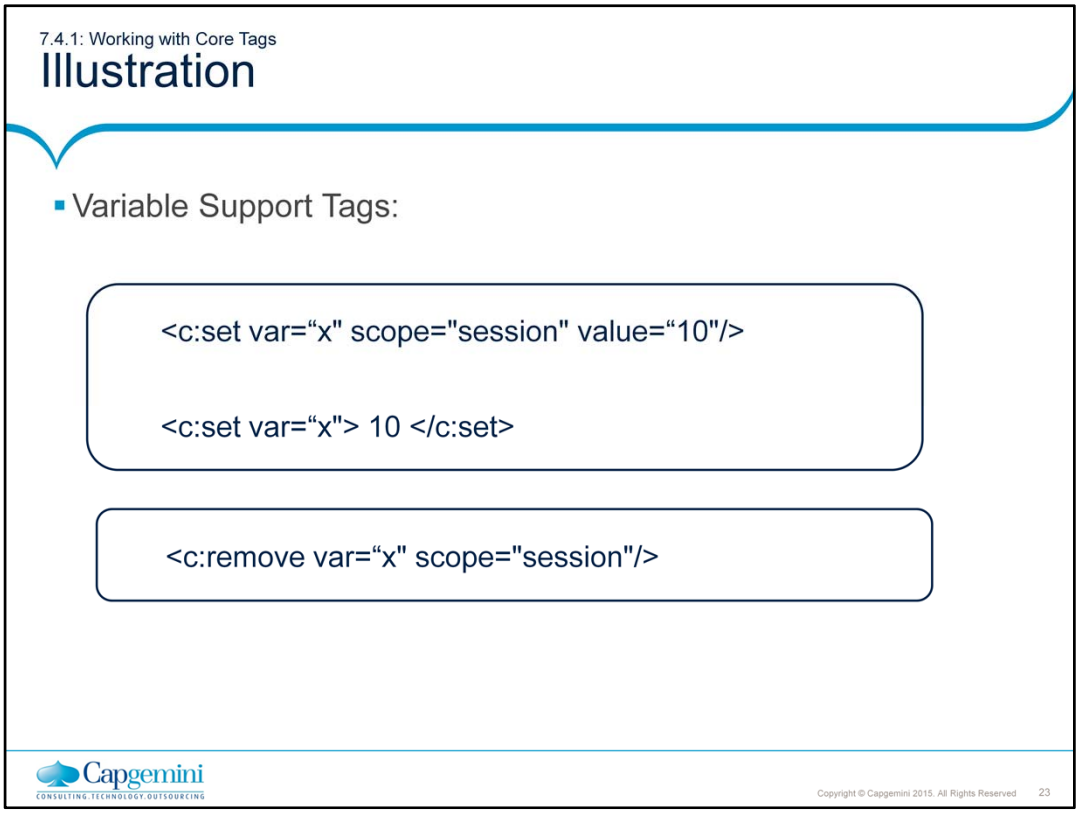
Using JSTL:

- The **JavaServer Pages Standard Tag Library (JSTL)** encapsulates core functionality common to many JSP applications. Instead of mixing tags from numerous vendors in JSP applications, JSTL allows to employ a single, standard set of tags. This standardization allows to deploy applications on any JSP container supporting JSTL and makes it more likely that the implementation of the tags is optimized.
- JSTL has tags such as iterators and conditionals for handling flow control, tags for manipulating XML documents, internationalization tags, tags for accessing databases using SQL, and commonly used functions.



Working with Core Tags:

- Core tags provide tags related to variables and flow control as well as a generic way to access URL-based resources whose content can then be included or processed within the JSP page.
  - **<c:set>:** It specifies basic variable setting actions
  - **<c:out>:** It specifies basic output actions
  - **<c:if test= ? :** Conditional action
  - **<c:choose>, <c:when> , <c:otherwise>:** Conditional action
  - **<c:forEach> :** Iteration actions
  - **<c:forTokens> :** Iteration actions
  - **<c:import> :** It retrieves content from a local jsp page or another server.
  - **<c:url> :** It specifies the URL of the content to retrieve.
  - **<c:redirect> :** It sends a HTTP redirect to the client.
  - **<c:param> :** It adds request parameters to an URL.
  - **<c:catch> :** It catches a java.lang.Throwable thrown by any of its nested actions.
- Next few slides illustrate these tags.



Working with Core Tags:

Variable Support Tags:

- The set tag sets the value of an EL variable or the property of an EL variable in any of the JSP scopes (page, request, session, or application). If the variable does not already exist, it is created.
- The JSP EL variable or property can be set from the attribute value as shown below:

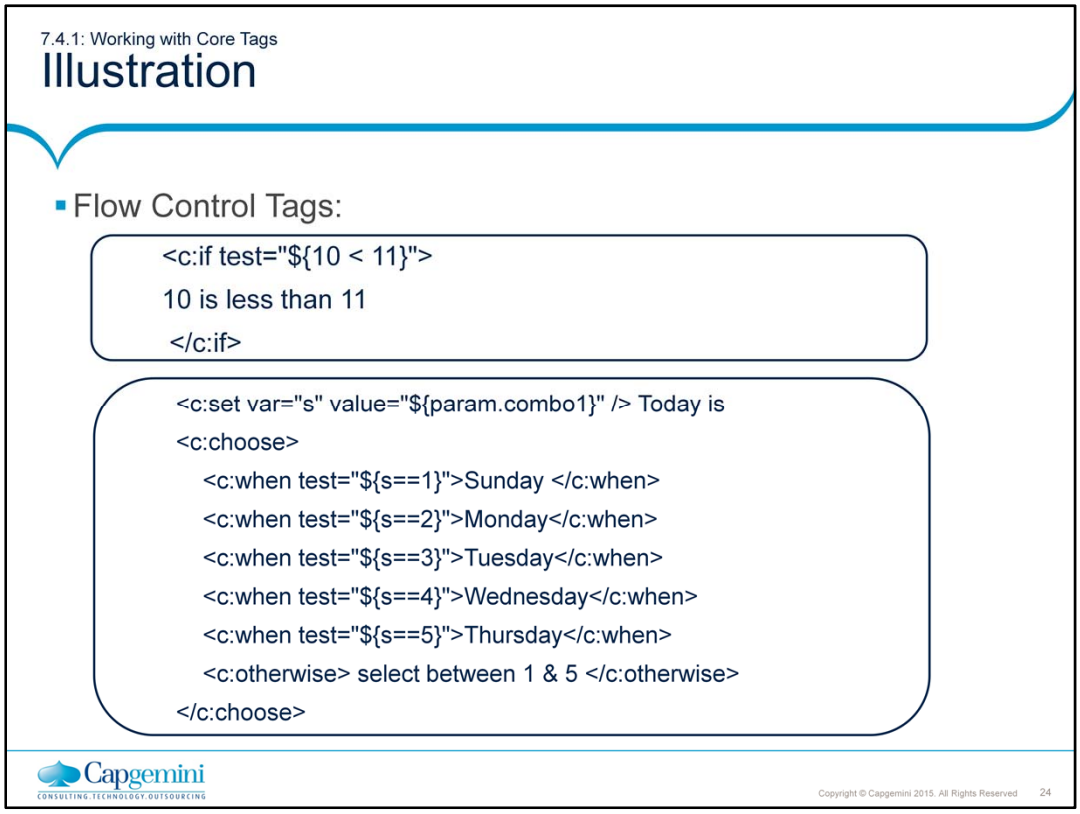
```
<c:set var="x" scope="session" value="10"/>
```

- Alternatively it can be set from the body of the tag, as shown below:

```
<c:set var="x">
  10
</c:set>
```

- To remove an EL variable, use the remove tag. Above variable is removed as follows:

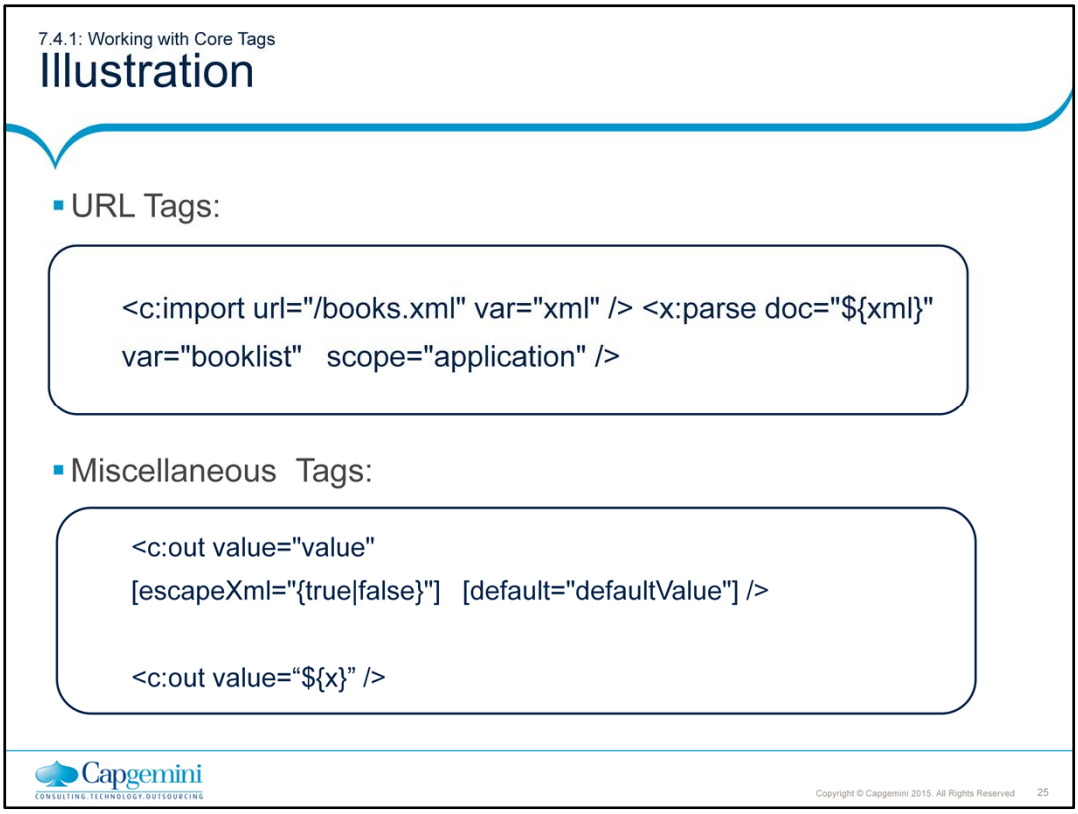
```
<c:remove var="x" scope="session"/>
```



Working with Core Tags:

Flow Control Tags :

- The illustration in the above slide shows the use of conditional tags like <c:if> and <c:choose>.
- For iteration tag illustration, please refer the slide no-6.



Working with Core Tags:

URL and Miscellaneous Tags:

- In the illustration in the above slide, “import” is used to read the XML document containing book information and assign the content to the scoped variable xml.
- The out tag evaluates an expression and outputs the result of the evaluation to the current **JspWriter** object.

Demo

- Working with Core Tags:
  - coreTags1.jsp
  - coreTags2.jsp
  - coreTags3.jsp
  - coreTags4.jsp
  - header.jsp
  - footer.jsp

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 26

Deploy web application **Lesson7-JSTL_Core_Tags** and show demo by executing index.jsp and referring to each of the above JSP pages.

Lab: JSTL

- Lab 3.1
- Lab 3.2
- Lab 3.3

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Summary

- In this lesson, you have learnt about:
  - The need for JSTL
  - The JSTL libraries
  - Features in EL expression
  - Different tags from core

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

- Question 1: Which of the following library provides tags for internationalization?
  - Option 1: sql
  - Option 2: xml
  - Option 3: fmt
  - Option 4: core

- Question 2: EL expression can be used within JSP scriptlet tags.
  - True/False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Question

- Question 3: Which of the following library provides tags for adding request parameters to an URL
  - Option 1: sql
  - Option 2: fn
  - Option 3: fmt
  - Option 4: core

- Question 4: EL language uses '#' symbol to write expression.
  - True / False

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

 30