UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECJ 3303 – INTERNET PROGRAMMING

TOPIC 3 – EL(Expression Language) & JSTL (JSP Standard Tag Library)

innovative ● entrepreneurial ● global

UTM JOHOR BAHRU

JSP EL & JSTL
(Expression Language & JSP Standard Tag Library)

- EL (Expression Language) and JSTL (JSP Standard Tag Library) were introduced with the JSP 2.0 specification.
- These tags have several advantages over the older JSP tags that were used prior to the JSP 2.0 specification
- These provides a compact syntax that lets us to get data from JavaBeans, maps, arrays, and lists that have been stored as attributes of a web application
- These tags provide a way to reduce the amount of scripting in your applications. .
- EL + JSTL can improve your JSPs code significantly
- (For most applications, you can use JSTL and EL to remove all JSP scripting ie scriptlet, expression, declaration etc…)
- As a matter of principle – using MVC, a JSP page should be a view page without scriptlets.

JSP EL (Expression Language)

Advantages:

- EL is more compact and elegant. This makes it easier to code and read
- EL makes it easy to access nested properties.
  - simpler syntax ie ${varname} to print value of a simple variable
  - ${user.address.postcode}, ${user.address.state} to print nested properties

- EL lets you access collections such as arrays, maps, and lists easily
- EL handles null values better than standard JSP tags
- EL provides functionality that isn't available from the standard JSP tags.
  - ie, it lets you work with HTTP headers, cookies, and context initialization parameters. It also lets you perform calculations and comparisons

Using EL for Accessing Application Data

- EL expressions must be enclosed between **${** and **}**.
- ${data} – scoped variable data.
- The dot (.) operator.
- The bracket ['name'] operator.
- E.g you can you either of these styles.
  - ${user.name}
  - ${user["name"]}

* ${customer.name} is equivalent to ${customer["name"]} is equivalent to ${customer[‘name’]}

TOPIC 3

Syntax comparison – Standard JSP Tags vs EL

Syntax – Standard JSP tags

```
<jsp:useBean id="user" scope="session" class="model.User"/>
<label>Email:</label>
<span><jsp:getProperty name="user" property="email"/></span><br>
<label>First Name:</label>
<span><jsp:getProperty name="user" property="firstName"/></span><br>
<label>Last Name:</label>
<span><jsp:getProperty name="user" property="lastName"/></span><br>
```

Syntax – EL

```
<label>Email:</label>           <span>${user.email}</span><br>
<label>First Name:</label> <span>${user.firstName}</span><br>
<label>Last Name:</label>  <span> ${user.lastName} </span><br>
```

Examples

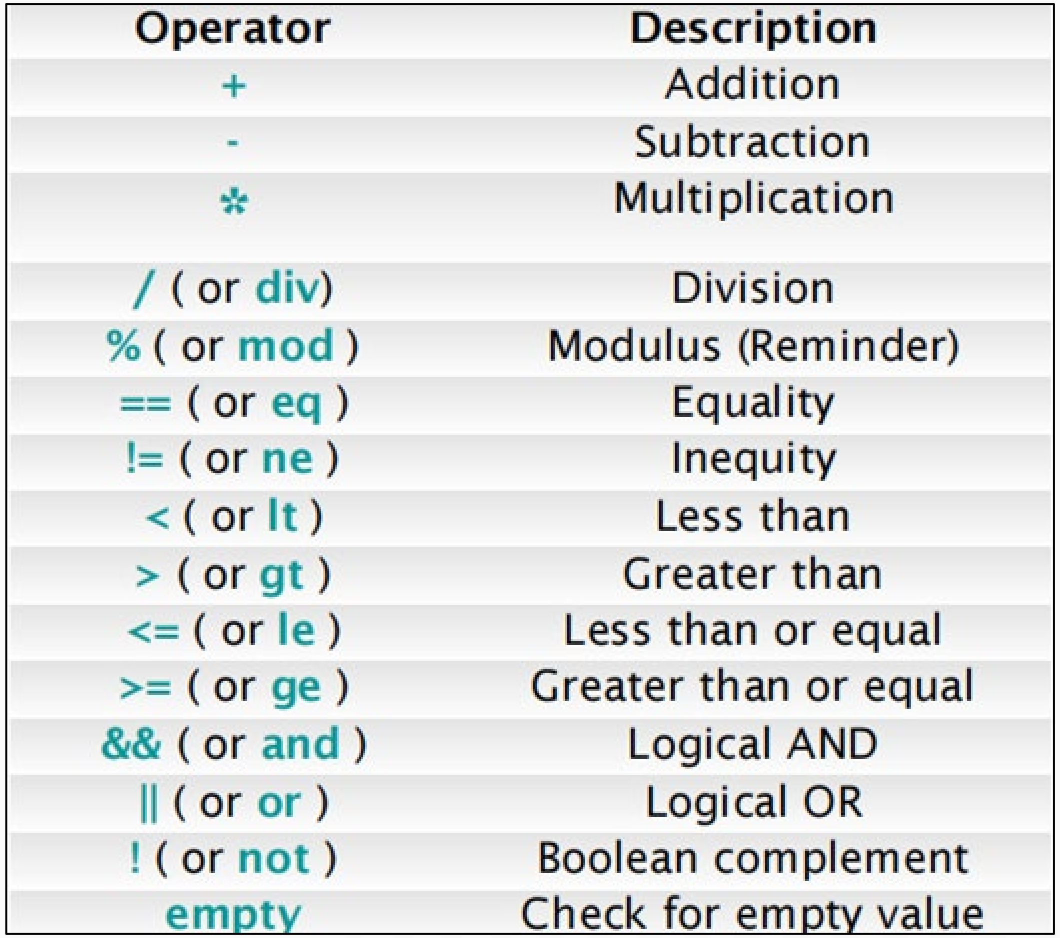
code in controller or servlet

```
String name = “muhammad”;
String nameArr[] = {“ali”, “siti” ,”zaki”};
Person p = new Person(“dina”,”female”,23);
request.setAttribute(“name”,name);
request.setAttribute(“nameArr”,nameArr);
request.setAttribute(“p”,p);
```

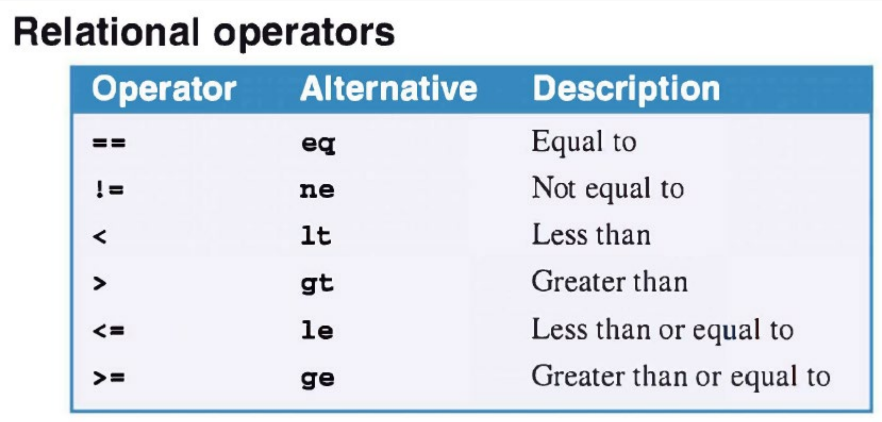
code in JSP page

```
Hello ${name}                                   //prints Hello muhammad
Hello ${nameArr[0]}                             //prints Hello ali
Hello ${nameArr[2]}                             //prints Hello zaki
Helllo ${p.gender}
or Hello ${p[“gender”]}                         //prints Hello female
```

Operators in EL

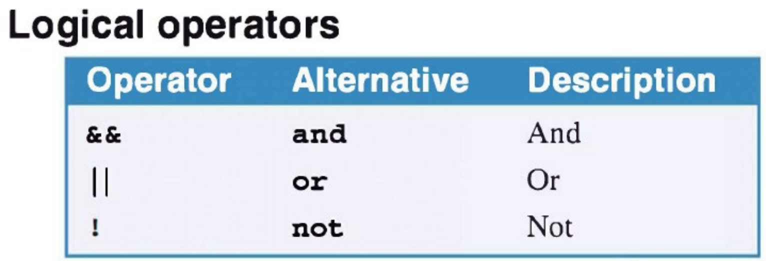


Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/ (or div)	Division
% (or mod)	Modulus (Reminder)
== (or eq)	Equality
!= (or ne)	Inequity
< (or lt)	Less than
> (or gt)	Greater than
<= (or le)	Less than or equal
>= (or ge)	Greater than or equal
&& (or and)	Logical AND
|| (or or)	Logical OR
! (or not)	Boolean complement
empty	Check for empty value



Relational operators

Operator	Alternative	Description
==	eq	Equal to
!=	ne	Not equal to
<	lt	Less than
>	gt	Greater than
<=	le	Less than or equal to
>=	ge	Greater than or equal to

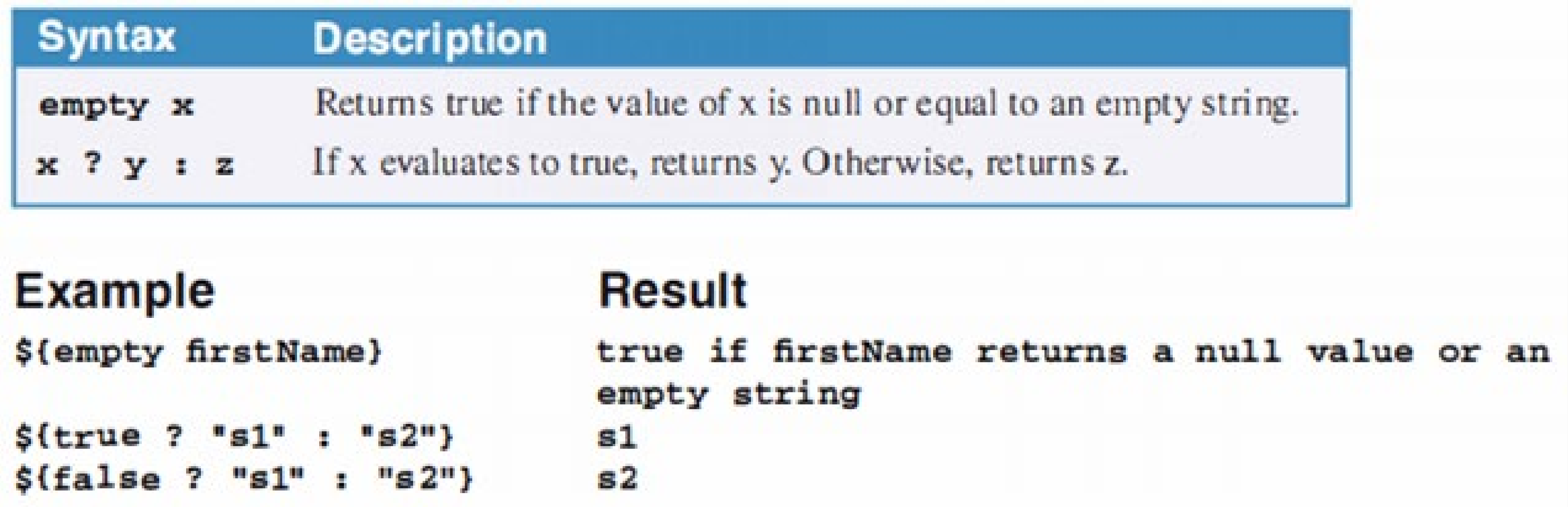


Logical operators

Operator	Alternative	Description
&&	and	And
||	or	Or
!	not	Not

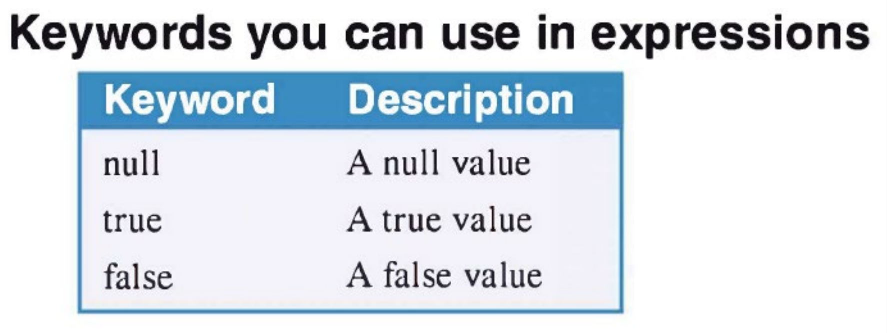
TOPIC 3

Other Operators



Syntax	Description
`empty x`	Returns true if the value of x is null or equal to an empty string.
`x ? y : z`	If x evaluates to true, returns y. Otherwise, returns z.

Example	Result
`${empty firstName}`	`true if firstName returns a null value or an empty string`
`${true ? "s1" : "s2"}`	`s1`
`${false ? "s1" : "s2"}`	`s2`



Keywords you can use in expressions

Keyword	Description
null	A null value
true	A true value
false	A false value

Examples

Code in jsp page

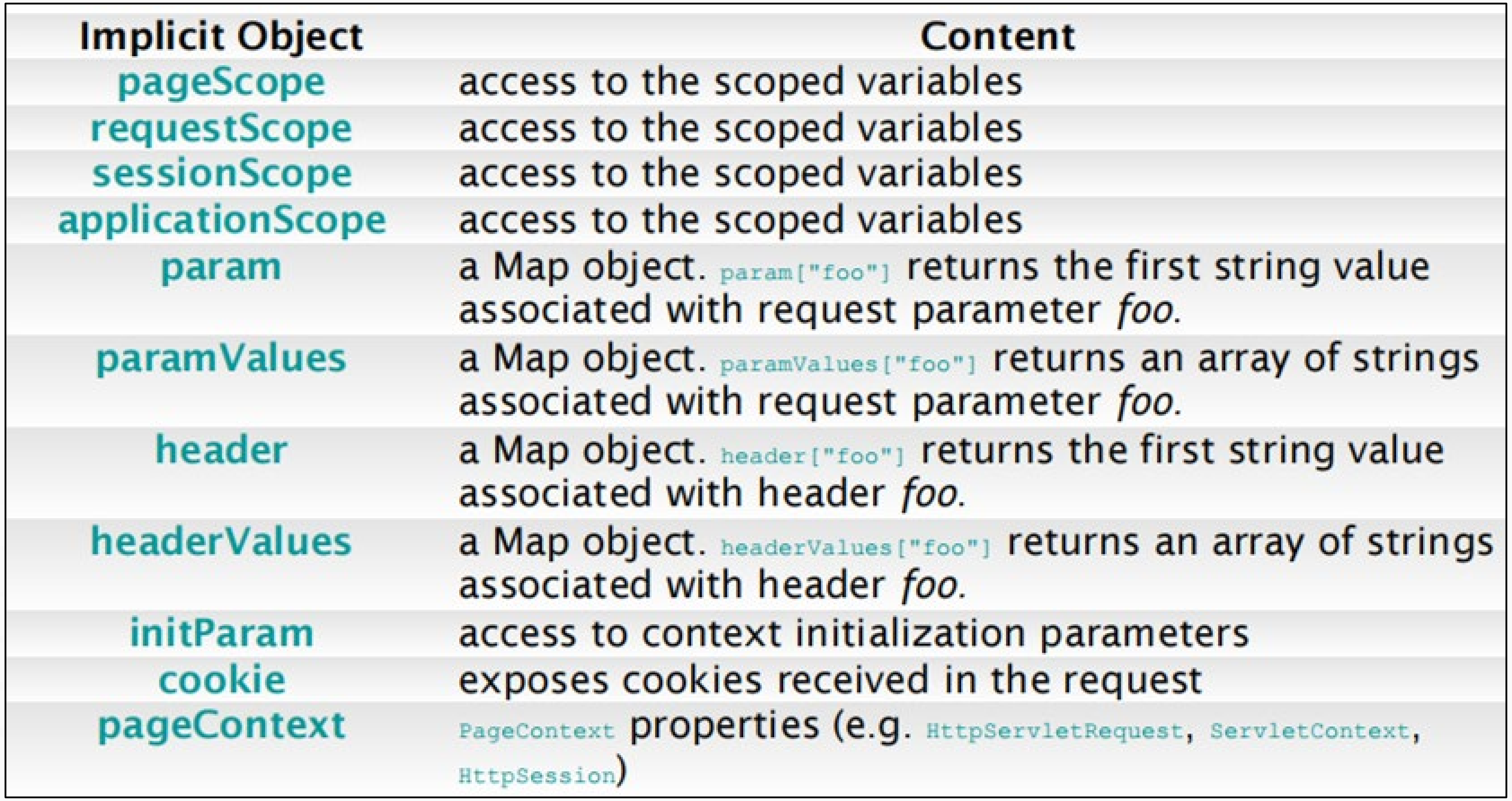
```
<h4>1+3 result is : ${1+3 }</h4>
<h4>7/2 result is : ${7/3 }</h4>
<h4>9 % 2 result is : ${9 % 2}</h4>
<h4>9 gt 5 result is : ${9 gt 5 }</h4>
<h4>9 le 5 result is : ${9 le 5 }</h4>
<h4>9 ne 5 result is : ${9 ne 5 }</h4>
<h4>9 < 5 result is : ${9 < 5 }</h4>
<h4>Result is : ${9 > 5 }</h4>
<h4>9 gt 5? : ${9 gt 5? "yes it is true": "no it is false"
}</h4>
```

Result :

```
1+3                 result is : 4
7/2                 result is :
2.3335
9 % 2     result is : 1
9 gt 5    result is : true
9 le 5    result is : false
9 ne 5    result is : true
9 < 5     result is : false
9 > 5     result is : true
9 gt 5? : yes it is true
```

TOPIC 3

Implicit Objects in EL



Implicit Object	Content
pageScope	access to the scoped variables
requestScope	access to the scoped variables
sessionScope	access to the scoped variables
applicationScope	access to the scoped variables
param	a Map object. `param["foo"]` returns the first string value associated with request parameter *foo*.
paramValues	a Map object. `paramValues["foo"]` returns an array of strings associated with request parameter *foo*.
header	a Map object. `header["foo"]` returns the first string value associated with header *foo*.
headerValues	a Map object. `headerValues["foo"]` returns an array of strings associated with header *foo*.
initParam	access to context initialization parameters
cookie	exposes cookies received in the request
pageContext	`PageContext` properties (e.g. `HttpServletRequest`, `ServletContext`, `HttpSession`)

JSTL

- JSTL provides tags for common tasks that need to be performed in JSPs
- JSTL + EL can be used to replace scriplet code from a JSP page
- To use JSTL tags, we must make the jstl-impl.jar and jstl-api.jar files available to the application (add into lib, or add depencency if use maven)
- also need to add the taglib directive that identifies the JSTL library and its prefix

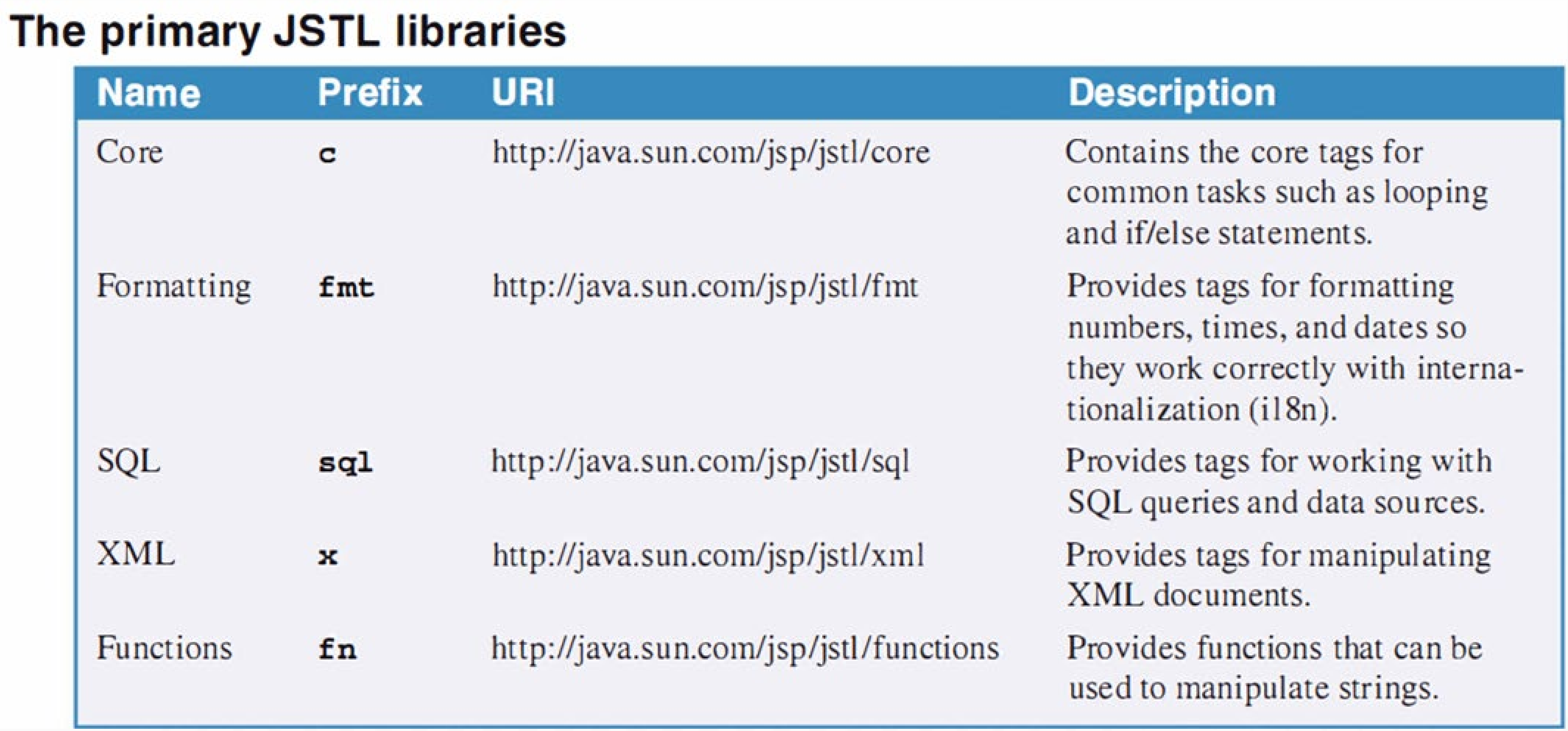
The 5 JSTL Libraries

JSP Standard Tag Library (JSTL) provides tags for common tasks that need to be performed in JSP.
The 5 JSTL Libraries are :-
(i) **core library:** This library contains tags that you can use to encode URLs, loop through collections, and code if/else statements.
If you use the MVC pattern, the tags in **the core library are often the only JSTL tags you'll need** as you develop your JSPs.

If necessary, though, you can use the other four libraries to work with (ii) internationalization, (iii) databases, (iv) XML, and (v) strings

In this course, will only cover the (i) JSTL Core Library/ i.e c:out, c:forEach, c:if

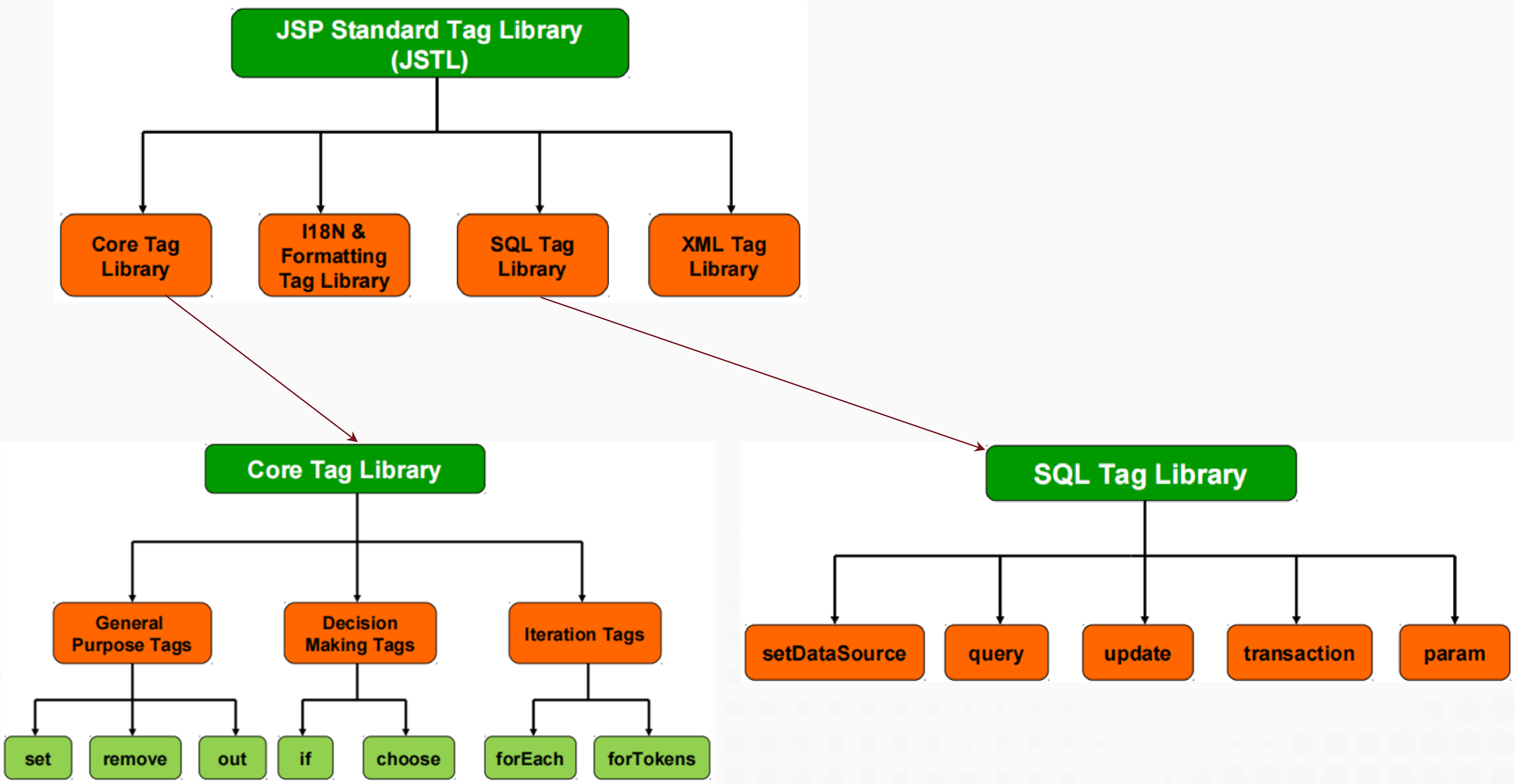
TOPIC 3



The primary JSTL libraries

Name	Prefix	URI	Description
Core	`c`	http://java.sun.com/jsp/jstl/core	Contains the core tags for common tasks such as looping and if/else statements.
Formatting	`fmt`	http://java.sun.com/jsp/jstl/fmt	Provides tags for formatting numbers, times, and dates so they work correctly with internationalization (i18n).
SQL	`sql`	http://java.sun.com/jsp/jstl/sql	Provides tags for working with SQL queries and data sources.
XML	`x`	http://java.sun.com/jsp/jstl/xml	Provides tags for manipulating XML documents.
Functions	`fn`	http://java.sun.com/jsp/jstl/functions	Provides functions that can be used to manipulate strings.

TOPIC 3



JSP Standard Tag Library (JSTL)

Core Tag Library

I18N & Formatting Tag Library

SQL Tag Library

XML Tag Library

Core Tag Library

General Purpose Tags

Decision Making Tags

Iteration Tags

set

remove

out

if

choose

forEach

forTokens

SQL Tag Library

setDataSource

query

update

transaction

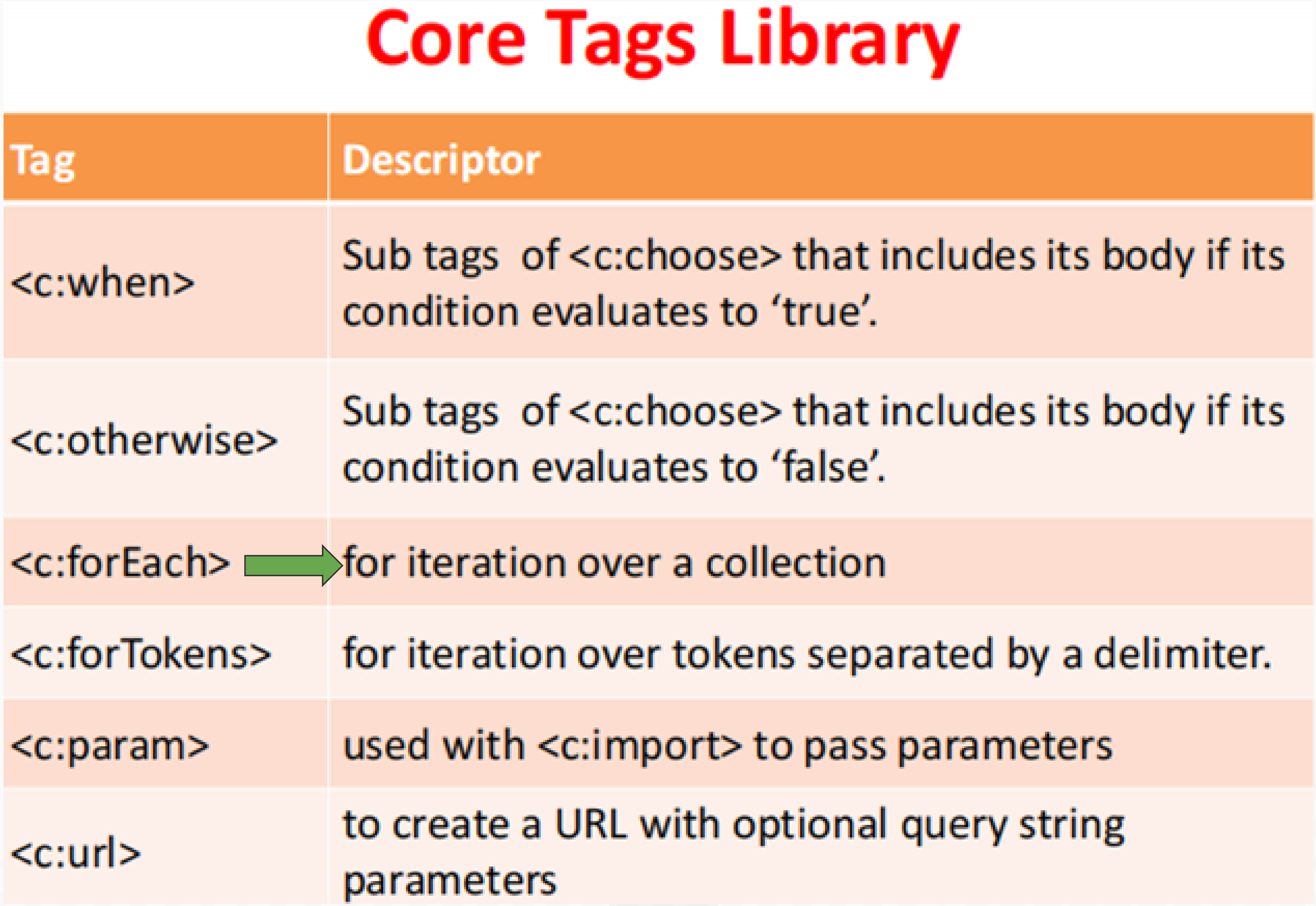
param

JSTL core

- JSTL Core provide support for output, iteration, conditional logic, catch exception, url, forward or redirect, response etc.
- Before you can use JSTL tags within an application, you must make the **(i) jstl-impl.jar** and **(ii) jstl-api.jar** files available to the application (**add jstl-1.2.jar to your project lib folder**, or add dependency for maven project)
- we also need to include the taglib in the JSP page as below:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

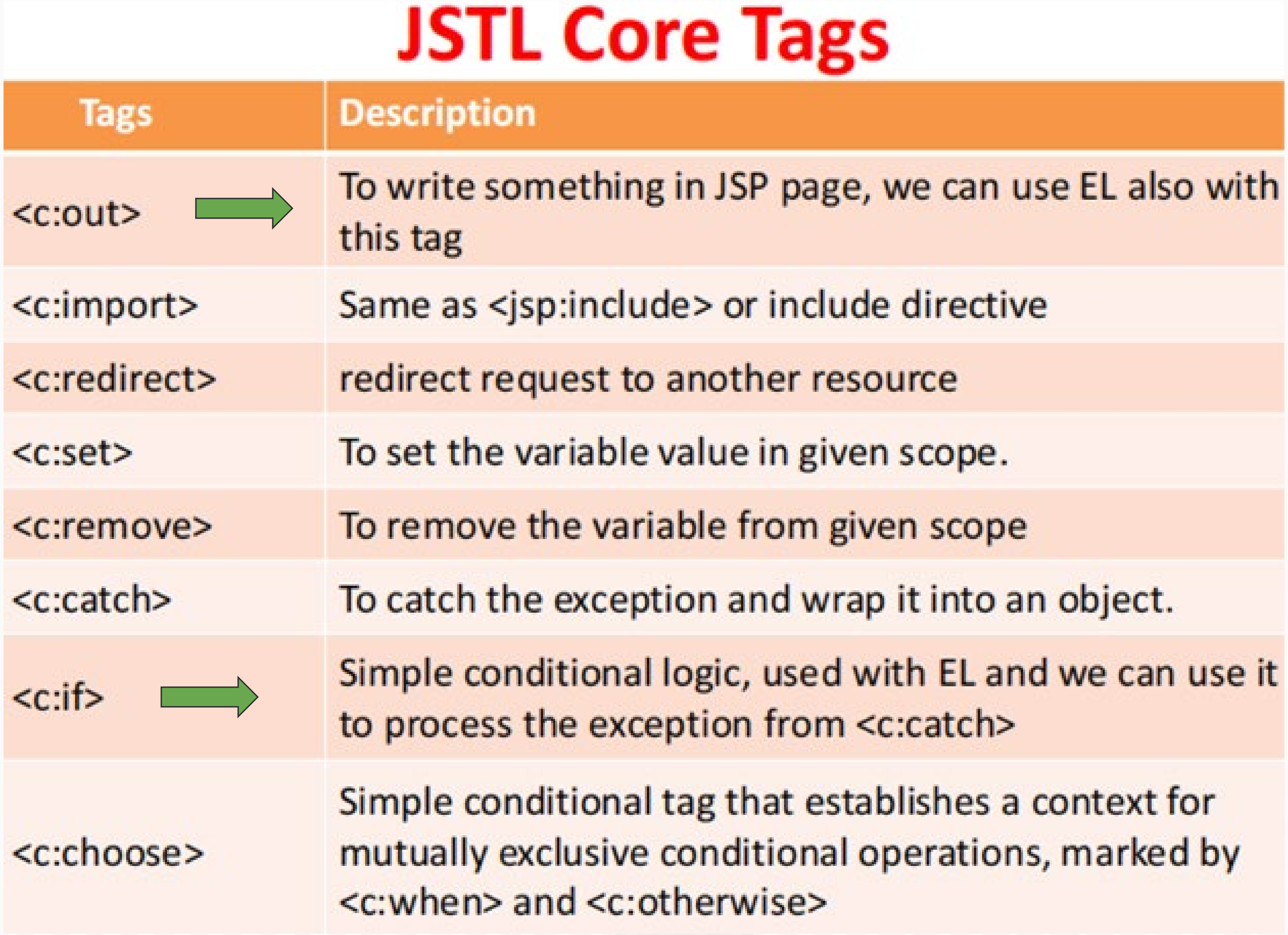
TOPIC 3



Core Tags Library

Tag	Descriptor
<c:when>	Sub tags of <c:choose> that includes its body if its condition evaluates to ‘true’.
<c:otherwise>	Sub tags of <c:choose> that includes its body if its condition evaluates to ‘false’.
<c:forEach>	for iteration over a collection
<c:forTokens>	for iteration over tokens separated by a delimiter.
<c:param>	used with <c:import> to pass parameters
<c:url>	to create a URL with optional query string parameters

TOPIC 3



JSTL Core Tags

Tags	Description
<c:out>	To write something in JSP page, we can use EL also with this tag
<c:import>	Same as <jsp:include> or include directive
<c:redirect>	redirect request to another resource
<c:set>	To set the variable value in given scope.
<c:remove>	To remove the variable from given scope
<c:catch>	To catch the exception and wrap it into an object.
<c:if>	Simple conditional logic, used with EL and we can use it to process the exception from <c:catch>
<c:choose>	Simple conditional tag that establishes a context for mutually exclusive conditional operations, marked by <c:when> and <c:otherwise>

TOPIC 3

Using the general purpose - c:out tag

code in controller or servlet

```
String name = “muhammad”;
String nameArr[] = {“ali”, “siti” ,”zaki”};
Person p = new Person(“dina”,”female”,23);
request.setAttribute(“name”,name);
request.setAttribute(“nameArr”,nameArr);
request.setAttribute(“p”,p);
```

code in JSP page

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
:
<c:out value="Hello ${name}" /><br>                          //prints Hello muhammad
<c:out value="Hello ${nameArr[0]}" /><br>              //prints Hello ali
<c:out value="Hello ${nameArr[2]}" /><br>              //prints Hello zaki
<c:out value="You are a ${p.gender}"  /><br>           //prints You are a female
or Hello ${p["gender"]}                 /><br>
```

Using the iterator - c:forEach tag

- You can use the forEach tag to loop through items that are stored in most collections (i.e array, list, etc..)

- You can use the var attribute to specify the variable name that is used to access each item within the collection.
- You can use the items attribute to specify the collection that stores the data.
- If necessary, you can nest one forEach tag within another.

TOPIC 3

Using the iterator – c:forEach tag

code in controller or servlet

```
String nameArr[] = {"ali", "siti" ,"zaki"};
request.setAttribute("nameArr",nameArr);
List<Person> pList = new ArrayList<Person>();
pList.add(new Person("dina","female",23));
pList.add(new Person("kali","male",32));
pList.add(new Person("Saly","female",17));
request.setAttribute("pList",pList);
```

code in JSP page

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
:
to print pList
<c:forEach var="person" items="${pList}" >
        Name  : ${person.name} <br>
        Gender : ${person.gender}<br>
        Age   : ${person.gender}<br><br>
</c:forEach>
```

TOPIC 3

cont ...

code in JSP page

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
:
to print array name
<c:forEach var=”name” items=”${nameArr}” >
     Name  : ${name} <br>
</c:forEach>


//prints                    Name ali
                            Name siti
                            Name zaki
```

Using the decision making- c:if tag

code in controller or servlet

```
Person p = new Person(“dina”,”female”,23);
request.setAttribute(“p”,p); // Passing Person object to JSP
```

code in JSP page

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
:
<c:if test="${p.gender == 'female'}" >
    <c:out value="you can wear a pink shirt tomorrow" />
</c:if>
<c:if test="${p.gender == 'male'}" >
    <c:out value="lets play futsal this evening" />
</c:if>
```

Simplifying JSP with EL and JSTL

- Accessing Java Objects Easily:
  - Use EL to access Java objects and attributes without scriptlets.
  - Example: Accessing session attributes with EL:

```
<p>User: ${sessionScope.user.name}</p>
```

- Benefits of EL and JSTL:
  - Cleaner Code: Eliminates the need for scriptlets (<% %>), enhancing readability.
  - More Maintainable: Encourages a separation of logic and presentation, making it easier to manage code.

- Combining EL and JSTL:

```
<c:choose>
  <c:when test="${not empty user}">
    <p>Welcome, ${user.name}!</p>
  </c:when>
  <c:otherwise>
    <p>Please log in.</p>
  </c:otherwise>
</c:choose>
```

Using JSTL in Combination with EL to Minimize Scriptlets in JSP Pages

Before Using EL and JSTL (Using Scriptlets)

```
<%
   List<Product> products = (List<Product>)
request.getAttribute("productList");
   for (Product product : products) {
%>
   <div>
      <h2><%= product.getName() %></h2>
      <p>Price: <%= product.getPrice() %></p>
   </div>
<%
   }
%>
```

After Using EL and JSTL

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>

<c:forEach var="product" items="${productList}">
   <div>
      <h2><c:out value="${product.name}" /></h2>
      <p>Price: <c:out value="${product.price}" /></p>
   </div>
</c:forEach>
```

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

TOPIC 03 – EL & JSTL

The End

innovative • entrepreneurial • global

UTM JOHOR BAHRU