



The JSF 2 Expression Language

JSF 2.2 Version

Originals of slides and source code for examples: <http://www.coreservlets.com/JSF-Tutorial/jsf2/>

Also see the PrimeFaces tutorial – <http://www.coreservlets.com/JSF-Tutorial/primefaces/>
and customized JSF2 and PrimeFaces training courses – <http://courses.coreservlets.com/jsf-training.html>

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



3



**For live training on JSF 2, PrimeFaces, or other
Java EE topics, email hall@coreservlets.com**
Marty is also available for consulting and development support

**Taught by the author of *Core Servlets and JSP*, this tutorial,
and JSF 2.2 version of *Core JSF*. Available at public venues, or
customized versions can be held on-site at your organization.**

- Courses developed and taught by Marty Hall
 - JSF 2, PrimeFaces, Ajax, jQuery, Spring MVC, JSP, Android, general Java, Java 8 lambdas/streams, GWT, custom topic mix
 - Courses available in any location worldwide. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Hadoop, Spring, Hibernate/JPA, RESTful Web Services

Contact hall@coreservlets.com for details



Topics in This Section

- **Motivating use of the expression language**
 - Comparing to the JSF 1.x and JSP 2.0 ELs
- **Simplified testing of EL capabilities**
- **Accessing bean properties**
 - Direct
 - Nested
- **Submitting bean properties**
 - Expressions in output values
 - Expressions in submission values
 - Expressions for action controllers
- **Accessing collection elements**
- **Using implicit objects and operators**
- **Conditionally rendering output**
- **Passing arguments to methods**

5

© 2015 Marty Hall



Overview



6

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

The Expression Language

- **JSP scripting not supported in facelets**
 - So, you need a way to indirectly invoke Java
- **Quick examples**
 - `#{employee.firstName}`
 - Call `getFirstName` on bean named `employee`. Output it.
 - `<h:inputText value="#{employee.firstName}"/>`
 - When form displayed, call `getFirstName`, and if non-empty, fill it in as initial value of textfield.
 - When form submitted, validate value and if it is OK, pass value to the `setFirstName` method
 - `#{employee.addresses[0].zip}`
 - Call `getAddresses` on bean named `employee` (which should return an array or list), then take first entry, then call `getZip` on that, then output it

7

Advantages of the Expression Language (Very Important)

- **Shorthand notation for bean properties**
 - To reference the result of the `getCompanyName` method of a managed bean named `company`, you use `#{company.companyName}`.
 - To reference the `firstName` property of the `president` property of a managed bean named `company`, you use `#{company.president.firstName}`.
- **Simple access to collection elements**
 - To reference an element of an array, List, or Map, you use `#{someBean.someProperty[indexOrKey]}`.
 - E.g., `#{person.friends[2]}`

8

Advantages of the EL (Moderately Important)

- **A small but useful set of simple operators**
 - To manipulate objects within EL expressions, you can use any of several arithmetic, relational, logical, or empty-testing operators.
- **Conditional output**
 - To choose among output options:
 - `{test ? option1 : option2}`
 - `<h:someElement ... rendered="{test}"/>`
 - `<ui:fragment rendered="...">...</ui:fragment>`
 - We will give very brief examples in this tutorial section. The later section on looping with `ui:repeat` will give more details.

9

Advantages of the EL (Less Important)

- **Predefined variables (implicit objects)**
 - To access request params, cookies, HTTP headers, and other standard types of request data, you can use one of several predefined implicit objects.
- **Passing arguments**
 - Version 2.1 of the EL lets you pass arbitrary arguments to methods. Works only in Java EE 6 or other servers that support EL 2.1. *Not part of JSF 2 itself.*
 - E.g, works in Tomcat 7 but not Tomcat 6, even though JSF 2 works in both.
- **Empty values instead of error messages**
 - In most cases, missing values or `NullPointerExceptions` result in empty strings, not thrown exceptions.

10

JSF vs. JSP ELs

Feature	JSF 2.0 EL	JSF 1.x EL (with JSP)	JSP 2.0 EL
Format	<code>#{blah}</code> (immediate output values could be accessed with <code>\${blah}</code>)	<code>#{blah}</code>	<code>\${blah}</code>
Where used	Anywhere in facelets page Eg: <code>#{customer.firstName}</code>	Only in attributes of JSF tags. Eg: <code><h:outputText value="#{customer.firstName}"/></code>	Anywhere in page Eg: <code>\${customer.firstName}</code>
Represents	Output data, later location for submitted data. Eg: <code><h:inputText value="#{customer.firstName}"/></code>	Output data, later location for submitted data. Eg: <code><h:inputText value="#{customer.firstName}"/></code>	Output data. Eg <code>\${customer.firstName}</code>
Where it looks for beans	Request, session, application (etc.) scopes and managed bean defs.	Request, session, application (etc.) scopes and managed bean defs.	Request, session, application scopes.
Declaration type	None needed for simplest usage. <code>xmlns</code> declaration for <code>h:</code> , <code>ui:</code> , <code>f:</code> tags.	<code>@taglib</code>	None needed
Environments	Java EE 6 servers or servlet 2.5 servers with JSF 2.0 JARs.	Java EE 5 servers or servlet 2.4 servers with JSF 1.x JARs.	Servlet 2.4+ servers

11

© 2015 Marty Hall



Simplified Testing of EL Capabilities



12

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

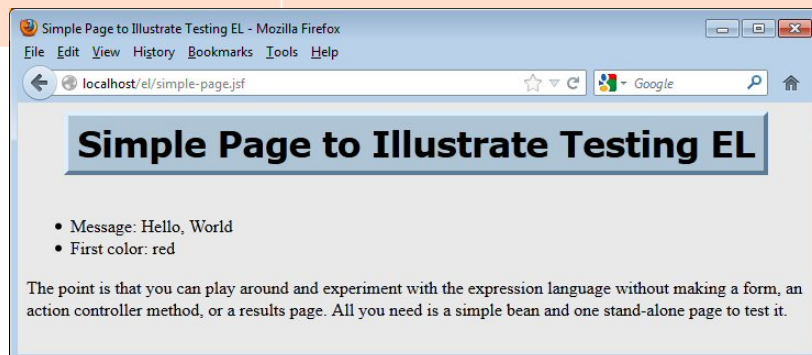
Simplifying Testing of the EL

- **JSP**
 - Checks existing scopes (request, session, etc.). If not found, gives up.
- **JSF**
 - Checks existing scopes (request, session, etc.). If not found, looks for managed bean definition of that name (either from @ManagedBean or from faces-config.xml).
- **Implication for testing and experimenting**
 - You can create a simple bean and a simple standalone page to test it. No form, no action controller, no results page. Great for experimenting with EL features.
 - See next page for an example

13

Simplifying Testing of the EL: Example

Bean	Standalone Test Page
<pre>@ManagedBean public class SimpleBean { private String[] colors = { "red", "orange", "yellow" }; public String getMessage() { return("Hello, World"); } public String[] getColors() { return(colors); } }</pre>	<pre><!DOCTYPE ...> <html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://xmlns.jcp.org/jsf/html"> ... Message: #{simpleBean.message} First color: #{simpleBean.colors[0]} ...</pre>



14



Outputting Simple Bean Properties



15

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Outputting Simple Bean Properties

- **Format**

- `#{varName.propertyName}`
- `<h:outputText value="#{varName.propertyName}" .../>`
 - For new JSF 2 code, top version is usually used unless you need some other attribute of `h:outputText` (e.g. “id”, “rendered”, or “escape”)

- **Interpretation**

- First, find `varName`
 - Search for “`varName`” in all defined scopes, from most specific to most general (request, session, application, in that order for standard Web app scopes). Then look in managed bean defs and instantiate if found.
- Call `getPropertyName` and output the result
 - This must be a normal zero-arg accessor method. If boolean, name of method could be `isPropertyName`

16

Bean Properties Example: Java Code

```
@ManagedBean
@ApplicationScoped
public class TestBean1 {
    private Date creationTime = new Date();
    private String greeting = "Hello";

    public Date getCreationTime() {
        return(creationTime);
    }

    public String getGreeting() {
        return(greeting);
    }

    public double getRandomNumber() {
        return(Math.random());
    }
}
```

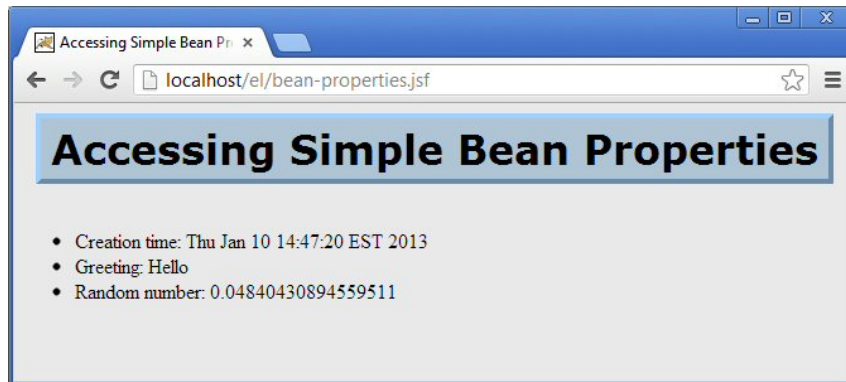
17

Bean Properties Example: Facelets Code

```
<!DOCTYPE ...>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
<h:head><title>Accessing Simple Bean Properties</title>
<link href="./css/styles.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
...
<ul>
    <li>Creation time: #{testBean1.creationTime}</li>
    <li>Greeting: #{testBean1.greeting}</li>
    <li>Random number: #{testBean1.randomNumber}</li>
</ul>
</h:body></html>
```

18

Bean Properties Example: Result



19

© 2015 Marty Hall



Accessing Nested Bean Properties



20

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Nested Bean Properties

- **Format**

- `{var.prop1.prop2.prop3}`
- `<h:outputText value="{var.prop1.prop2.prop3}" .../>`
 - Again, use this form only if you need some extra attribute of `h:outputText` such as “id”, “rendered”, or “escape”

- **Interpretation**

- First, find var
 - Same as before. Look in existing scopes (narrowest to widest). Use if found. If not found, look in managed bean defs and instantiate.
- Call `getProp1` on bean
- Call `getProp2` on result of `getProp1`
- Call `getProp3` on result of `getProp2`
 - And then output the result

21

Nested Properties Example: Name

```
public class Name {
    private String firstName, lastName;

    public Name(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return(firstName);
    }

    public void setFirstName(String newFirstName) {
        firstName = newFirstName;
    }
    ...
}
```

22

Nested Properties Example: Company

```
public class Company {  
    private String companyName, business;  
  
    public Company(String companyName, String business) {  
        this.companyName = companyName;  
        this.business = business;  
    }  
  
    public String getCompanyName() { return(companyName); }  
  
    public void setCompanyName(String newCompanyName) {  
        companyName = newCompanyName;  
    }  
    ...  
}
```

23

Nested Properties Example: Employee

```
public class Employee {  
    private Name name;  
    private Company company;  
  
    public Employee(Name name, Company company) {  
        this.name = name;  
        this.company = company;  
    }  
  
    public Name getName() { return(name); }  
  
    public Company getCompany() { return(company); }  
  
    ...  
}
```

24

Nested Properties Example: Employee1

```
@ManagedBean
public class Employee1 extends Employee {
    public Employee1() {
        super(new Name("Marty", "Hall"),
            new Company("coreservlets.com",
                "Customized Java EE and Ajax Training"));
    }
}
```

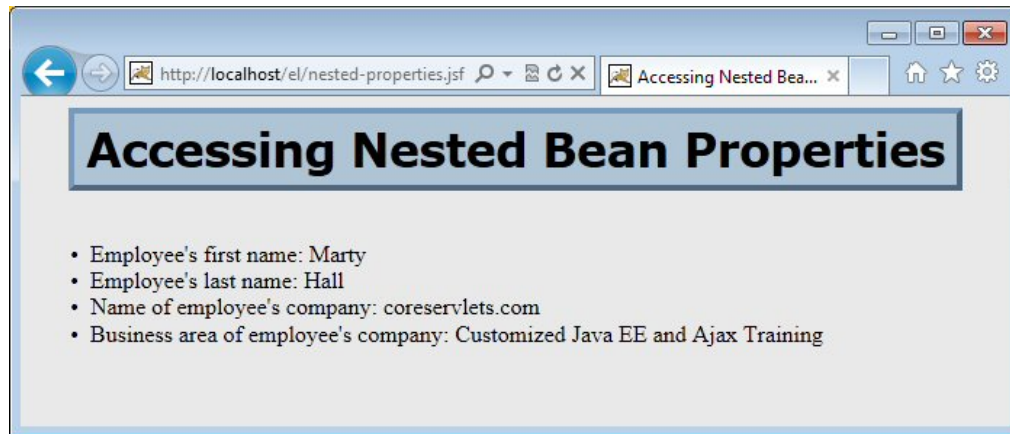
25

Nested Properties Example: Facelets Code

```
...
<ul>
    <li>Employee's first name:
        #{employee1.name.firstName}</li>
    <li>Employee's last name:
        #{employee1.name.lastName}</li>
    <li>Name of employee's company:
        #{employee1.company.companyName}</li>
    <li>Business area of employee's company:
        #{employee1.company.business}</li>
</ul>
...
```

26

Nested Properties Example: Result



27

© 2015 Marty Hall



Submitting Bean Properties



28

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Three Uses of #{...}

- **Designating output value**

- `#{employee.address}` or `<h:outputText value="#{employee.address}"/>`
 - Anytime accessed, means to output `getAddress`
- `<h:inputText value="#{employee.address}"/>`
 - When form initially displayed, means to prepopulate field. Call `getAddress` and put value in field if non-empty.

- **Designating submitted value**

- `<h:inputText value="#{employee.address}"/>`
 - When form submitted, designates where value stored. Pass textfield value to `setAddress`.

- **Designating method call after submission**

- `<h:commandButton value="Button Label" action="#{employee.processEmployee}"/>`
 - When form submitted, designates action handler. This is exact method name, not a shorthand for it.

29

Understanding Getter vs. Setter Method Correspondence

- **Example**

- `<h:inputText value="#{myBean.a.b.c.d}"/>`

- **When displaying form**

- Find or instantiate `myBean`. Call `getA`. Call `getB` on result. Call `getC` on that result. Call `getD` on that result. If non-empty use as initial value of textfield.

- **When submitting form**

- Find `myBean` (instantiate new version if in request scope). Call `getA`. Call `getB` on result. Call `getC` on that result. Then pass submitted value to the `setD` method of that result.
 - Point: only *final* one becomes setter on submission.
 - This assumes value passes validation. Discussed later.

30

Submitting Properties Example: Employee

```
public class Employee {
    private Name name;
    private Company company;

    ...

    public String processEmployee() {
        if (Math.random() < 0.5) {
            return("accepted");
        } else {
            return("rejected");
        }
    }
}
```

31

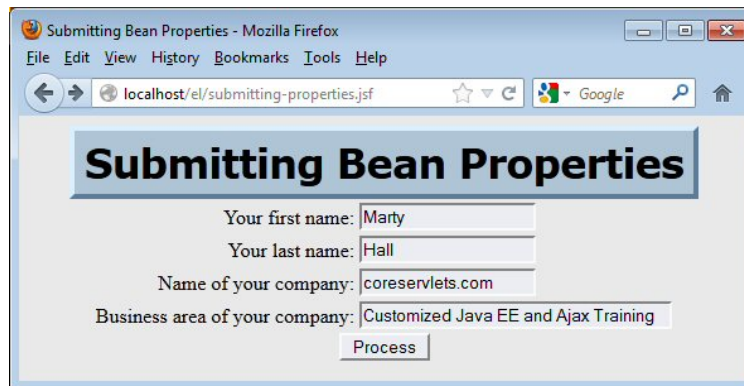
Submitting Properties Example: Facelets Code for Form

```
...
<h:form>
<h:panelGrid columns="2" styleClass="formTable">
Your first name:
<h:inputText value="#{employee1.name.firstName}"/>
Your last name:
<h:inputText value="#{employee1.name.lastName}"/>
Name of your company:
<h:inputText value="#{employee1.company.companyName}"/>
Business area of your company:
<h:inputText value="#{employee1.company.business}"
              size="38"/>
</h:panelGrid>
<h:commandButton value="Process"
                  action="#{employee1.processEmployee}"/>
</h:form>
```

32

h:panelGrid is a shortcut for making an HTML table.
More details in the lecture on validating form data.

Submitting Properties Example: Input Page Initial Result



Submitting Bean Properties - Mozilla Firefox

File Edit View History Bookmarks Tools Help

localhost/el/submitting-properties.jsf

Google

Submitting Bean Properties

Your first name:

Your last name:

Name of your company:

Business area of your company:

33

Submitting Properties Example: accepted.xhtml

```
...  
<table border="5" align="center">  
  <tr><th class="title">Employee Accepted</th></tr>  
</table>  
<p/>  
<ul>  
  <li>Employee's first name:  
    #{employee1.name.firstName}</li>  
  <li>Employee's last name:  
    #{employee1.name.lastName}</li>  
  <li>Name of employee's company:  
    #{employee1.company.companyName}</li>  
  <li>Business area of employee's company:  
    #{employee1.company.business}</li>  
</ul>  
...
```

34

Submitting Properties Example: rejected.xhtml

```
...
<table border="5" align="center">
  <tr><th class="title">Employee Rejected</th></tr>
</table>
<p/>
<ul>
  <li>Employee's first name:
    #{employee1.name.firstName}</li>
  <li>Employee's last name:
    #{employee1.name.lastName}</li>
  <li>Name of employee's company:
    #{employee1.company.companyName}</li>
  <li>Business area of employee's company:
    #{employee1.company.business}</li>
</ul>
...
```

35

Submitting Properties Example: Results

The screenshot shows two overlapping browser windows. The top window, titled 'Submitting Bean Properties - Mozilla Firefox', displays a web form at 'localhost/el/submitting-properties.jsf'. The form contains four input fields: 'Your first name:' with 'Bill', 'Your last name:' with 'Gates', 'Name of your company:' with 'microsoft.com', and 'Business area of your company:' with 'Wielding Monopoly'. A 'Process' button is at the bottom. The bottom window shows the result page, also at 'localhost/el/submitting-properties.jsf', with a heading 'Employee Accepted' and a bulleted list of the submitted data: Employee's first name: Bill, Employee's last name: Gates, Name of employee's company: microsoft.com, and Business area of employee's company: Wielding Monopoly.

Submitting Bean Properties - Mozilla Firefox

File Edit View History Bookmarks Tools Help

localhost/el/submitting-properties.jsf

Google

Submitting Bean Properties

Your first name: Bill

Your last name: Gates

Name of your company: microsoft.com

Business area of your company: Wielding Monopoly

Process

localhost/el/submitting-properties.jsf

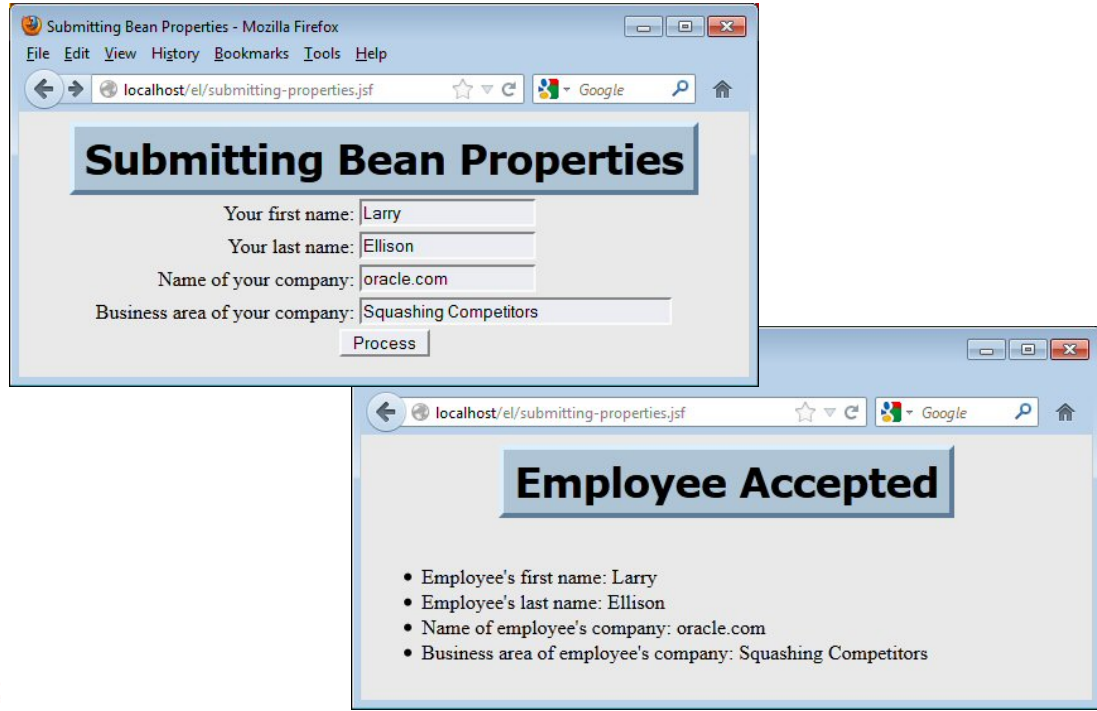
Google

Employee Accepted

- Employee's first name: Bill
- Employee's last name: Gates
- Name of employee's company: microsoft.com
- Business area of employee's company: Wielding Monopoly

36

Submitting Properties Example: Results (Continued)



37

© 2015 Marty Hall



Accessing Collections



38

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Equivalence of Dot and Array Notations

- **Equivalent forms**
 - `#{name.property}`
 - Only legal if “property” would be legal Java variable name
 - `#{name["property"]}`
- **Reasons for using bracket notation**
 - To access arrays, lists, and other collections
 - See upcoming slides
 - To calculate the property name at request time.
 - `#{name1[name2]}` (no quotes around name2)
 - To use names that are illegal as Java variable names
 - `#{foo["bar-baz"]}`
 - `#{foo["bar.baz"]}`

39

Using the [] Form

- **Works for**
 - Array. Equivalent to
 - `theArray[index]` (getting and setting)
 - List. Equivalent to
 - `theList.get(index)` or `theList.set(index, submittedVal)`
 - Map. Equivalent to
 - `theMap.get(key)` or `theMap.put(key, submittedVal)`
- **Equivalent forms (for Maps)**
 - `#{stateCapitals["maryland"]}`
 - `#{stateCapitals.maryland}`
 - But you can't use this for lists (numbers are not legal Java variables names, so `#{listVar.2}` is illegal). And not all hash table keys are legal variable names. So, use brackets.

40

Collections Example: Purchases

```
@ManagedBean
public class Purchases {
    private String[] cheapItems =
        { "Gum", "Yo-yo", "Pencil" };
    private List<String> mediumItems =
        new ArrayList<>();
    private Map<String,String> valuableItems =
        new HashMap<>();
    private boolean isEverythingOK = true;

    public Purchases() {
        mediumItems.add("iPod");
        mediumItems.add("GameBoy");
        mediumItems.add("Cell Phone");
        valuableItems.put("low", "Lamborghini");
        valuableItems.put("medium", "Yacht");
        valuableItems.put("high", "JSF Training Course");
    }
}
```

41

Collections Example: Purchases (Continued)

```
public String[] getCheapItems() {
    return(cheapItems);
}

public List<String> getMediumItems() {
    return(mediumItems);
}

public Map<String,String> getValuableItems() {
    return(valuableItems);
}
```

42

Collections Example: Purchases (Continued)

```
public String purchaseItems() {
    isEverythingOK = Utils.doBusinessLogic(this);
    isEverythingOK = Utils.doDataAccessLogic(this);
    if (isEverythingOK) {
        return("purchase-success");
    } else {
        return("purchase-failure");
    }
}
```

43

Collections Example: Utils

```
public class Utils {
    public static boolean doBusinessLogic
                          (PurchaseBean bean) {
        // Business logic omitted
        return(Math.random() > 0.1);
    }

    public static boolean doDataAccessLogic
                          (PurchaseBean bean) {
        // Database access omitted
        return(Math.random() > 0.1);
    }
}
```

44

Collections Example: using-collections.xhtml

...

```
<h:form>
```

```
<ul>
```

```
<li><b>Cheap Items</b>
```

```
<ol>
```

```
<li>
```

```
  <h:inputText value="#{purchases.cheapItems[0]}" />
```

```
</li>
```

```
<li>
```

```
  <h:inputText value="#{purchases.cheapItems[1]}" />
```

```
</li>
```

```
<li>
```

```
  <h:inputText value="#{purchases.cheapItems[2]}" />
```

```
</li>
```

```
</ol></li>
```

This example uses explicit indices. See the tutorial section on looping to see how to redo this example with `ui:repeat` and a variable for the index.

Collections Example: using-collections.xhtml (Continued)

```
<li><b>Medium Items</b>
```

```
<ol>
```

```
<li>
```

```
  <h:inputText value="#{purchases.mediumItems[0]}" />
```

```
</li>
```

```
<li>
```

```
  <h:inputText value="#{purchases.mediumItems[1]}" />
```

```
</li>
```

```
<li>
```

```
  <h:inputText value="#{purchases.mediumItems[2]}" />
```

```
</li>
```

```
</ol></li>
```

Collections Example: using-collections.xhtml (Continued)

```
<li><b>Valuable Items</b>
<ul>
<li>Low:
    <h:inputText value="#{purchases.valuableItems["low"]}" />
</li>
<li>Medium:
    <h:inputText
        value="#{purchases.valuableItems["medium"]}" />
</li>
<li>High:
    <h:inputText
        value="#{purchases.valuableItems["high"]}" />
</li>
</ul></li>
</ul>
<h:commandButton value="Purchase"
    action="#{purchases.purchaseItems}" />
```

Since I use double quotes around the Map key, I use single quotes here.

47

Collections Example: Input Page Initial Result

Using Collections - Mozilla Firefox

File Edit View History Bookmarks Tools Help

localhost/el/using-collections.jsf

Using Collections

Choose Purchases

- Cheap Items
 1. Gum
 2. Yo-yo
 3. Pencil
- Medium Items
 1. iPod
 2. GameBoy
 3. Cell Phone
- Valuable Items
 - Low: Lamborghini
 - Medium: Yacht
 - High: JSF Training Course

Purchase

48

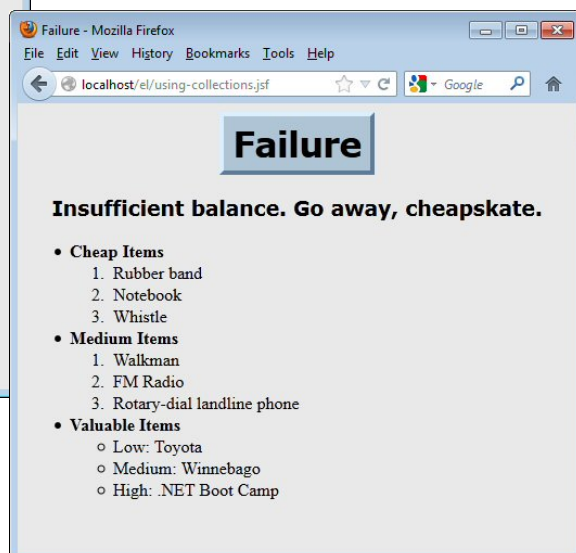
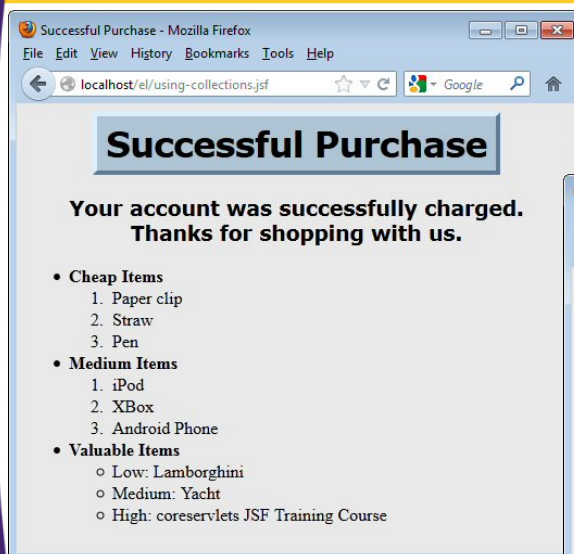
Submitting Properties Example: purchase-success.xhtml

[purchase-failure.xhtml](#) is very similar.

```
...
<li><b>Cheap Items</b>
<ol>
<li>#{purchases.cheapItems[0]}</li>
<li>#{purchases.cheapItems[1]}</li>
<li>#{purchases.cheapItems[2]}</li>
</ol></li>
<li><b>Medium Items</b>
<ol>
<li>#{purchases.mediumItems[0]}</li>
<li>#{purchases.mediumItems[1]}</li>
<li>#{purchases.mediumItems[2]}</li>
</ol></li>
<li><b>Valuable Items</b>
<ul>
<li>Low: #{purchases.valuableItems["low"]}</li>
<li>Medium: #{purchases.valuableItems["medium"]}</li>
<li>High: #{purchases.valuableItems["high"]}</li>
</ul></li>
</ul>
```

49

Submitting Properties Example: Results



50



Implicit Objects and Operators



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

51

JSF EL Has Almost the Same Predefined Variables as JSP 2

- **Predefined variables**

- facesContext. The FacesContext object.
 - E.g. `#{facesContext.externalContext.remoteUser}`
- param. Request params.
 - E.g. `#{param.custID}`
- header. Request headers.
 - E.g. `#{header.Accept}` or `#{header["Accept"]}`
 - `#{header["Accept-Encoding"]}`
- cookie. Cookie object (not cookie value).
 - E.g. `#{cookie.userCookie.value}` or `#{cookie["userCookie"].value}`
- request, session
 - `#{request.contextPath}`, `#{request.queryString}`, `#{session.id}`
 - `#{request.contextPath}` useful for making relative URLs. See templating section.
- initParam. Context initialization param.

- **Problem**

- Using implicit objects works poorly with MVC model. You usually want to use these values in the Java code, not in the facelets pages.

52

Example: Implicit Objects (Facelets Code)

```
...
<ul>
  <li><b>Context path:</b>
    #{request.contextPath}</li>
  <li><b>Value of JSESSIONID cookie:</b>
    #{cookie.JSESSIONID.value}</li>
  <li><b>The "test" request parameter:</b>
    #{param.test}</li>
  <li><b>User-Agent request header:</b>
    #{header["User-Agent"]}</li>
</ul>
...
```

53

Example: Implicit Objects (Result)



54

Expression Language Operators

- **Arithmetic**
 - + - * / div % mod
- **Relational**
 - == or eq, != or ne, < or lt, > or gt, <= or le, >= or ge
 - Note: in many contexts in XML, using the operators that contain "<" is illegal. So, you usually use lt instead of <, le instead of <=, etc.
- **Logical**
 - && and || or ! Not
- **Empty**
 - empty
 - True for null, empty string, empty array, empty list, empty map. False otherwise.
- **Note**
 - Use operators sparingly to preserve MVC model

55

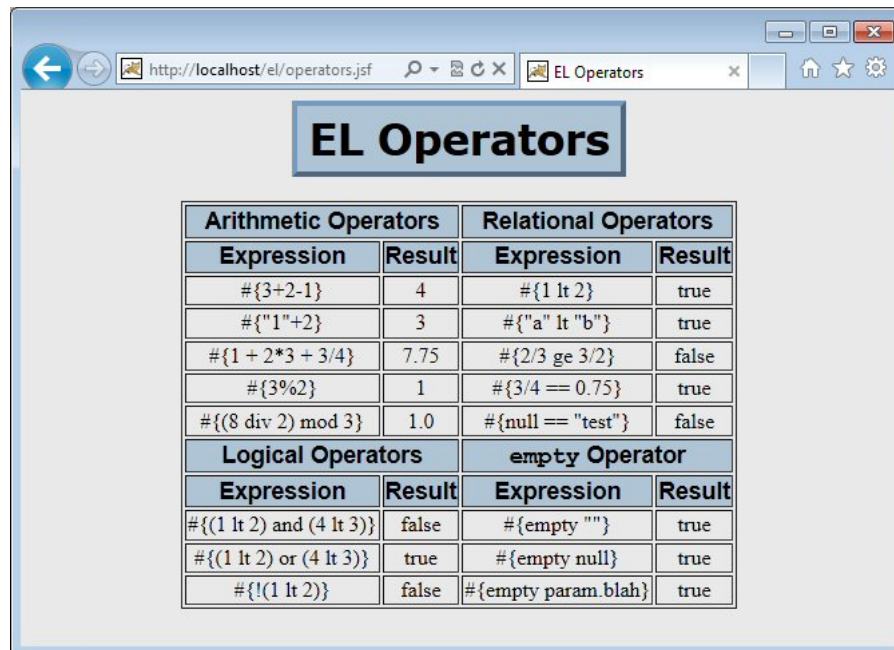
Example: Operators

```
...
<table border="1" align="center">
...
  <tr align="center">
    <td>\#{3+2-1}</td><td>#{3+2-1}</td>
    <td>\#{1 lt 2}</td><td>#{1 lt 2}</td></tr>
  <tr align="center">
    <td>\#{1"+2}</td><td>#{1"+2}</td>
    <td>\#{a lt b}</td><td>#{a lt b}</td></tr>
  <tr align="center">
    <td>\#{1 + 2*3 + 3/4}</td><td>#{1 + 2*3 + 3/4}</td>
    <td>\#{2/3 ge 3/2}</td><td>#{2/3 ge 3/2}</td></tr>
  <tr align="center">
    <td>\#{3%2}</td><td>#{3%2}</td>
    <td>\#{3/4 == 0.75}</td><td>#{3/4 == 0.75}</td></tr>
...
</table>
...
```

\#{blah} is taken literally. The backslash prevents EL evaluation.

56

Example: Operators (Result)



The screenshot shows a web browser window with the address bar displaying 'http://localhost/el/operators.jsf'. The page title is 'EL Operators'. The content area contains four tables of EL operators and their results.

Arithmetic Operators		Relational Operators	
Expression	Result	Expression	Result
<code>#{3+2-1}</code>	4	<code>#{1 lt 2}</code>	true
<code>#{"1"+2}</code>	3	<code>#{"a" lt "b"}</code>	true
<code>#{1 + 2*3 + 3/4}</code>	7.75	<code>#{2/3 ge 3/2}</code>	false
<code>#{3%2}</code>	1	<code>#{3/4 == 0.75}</code>	true
<code>#{(8 div 2) mod 3}</code>	1.0	<code>#{null == "test"}</code>	false

Logical Operators		empty Operator	
Expression	Result	Expression	Result
<code>#{(1 lt 2) and (4 lt 3)}</code>	false	<code>#{empty ""}</code>	true
<code>#{(1 lt 2) or (4 lt 3)}</code>	true	<code>#{empty null}</code>	true
<code>#{!(1 lt 2)}</code>	false	<code>#{empty param.blah}</code>	true

57

© 2015 Marty Hall



Conditional Output



58

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Conditional Text in JSF

- **Alternatives**

- `{someCondition ? simpleVal1 : simpleVal2}`
- `<h:outputText value="{someValue}" rendered="{someCondition}"/>`
 - Or, in general, use `h:blah` and the “rendered” attribute
- `<ui:fragment rendered="{someCondition}">`
`<someHTML>...</someHTML>`
`</ui:fragment>`

- **Note**

- More detailed examples shown in tutorial section on looping in facelets pages

59

Conditional Text with `{ condition ? val1 : val2 }`

- **Idea**

- The EL directly supports limited conditional output via the ternary operator (`test ? thenResult : elseResult`). Supply a boolean for the test, put conditional content after the “?” and/or the “:”. Values can be literal strings or EL expressions, but they cannot contain HTML tags.
 - Note: you are not permitted to omit the “else” part!

- **Examples**

- `<td class="{customer.balance < 0 ? 'red': 'black'}">`
- `{ !status.last ? ',' : '' }`

- **When used**

- When you are outputting simple text (no HTML).

If you want to output HTML, you could use the ternary operator within `h:outputText` and supply `escape="false"`. But in that case, one of the other two upcoming alternatives is probably simpler.

60

Conditional Text with h:outputText and “rendered”

- **Idea**

- Pass a boolean to the “rendered” attribute, put conditional content in “value” attribute. The value can be a literal string or an EL expression, but the literal string cannot contain HTML tags.

- **Examples**

- `<h:outputText rendered="#{!status.last}" value=","/>`
- `<h:outputText rendered="#{status.index > 5}" value="#{user.someWarning}" escape="false"/>`

The assumption here is that the `getSomeWarning` method outputs a string containing HTML tags. If so, the `escape="false"` is needed to prevent JSF from turning the `<` into `<` and so forth.

- **When used**

- When you are outputting simple text (no HTML) or when the HTML comes from a bean.

61

More on “rendered” Attribute

- **Almost all h:blah elements use “rendered”**

- So, you can insert almost any JSF element conditionally.

- **Example**

- Insert either textfield followed by button or simple value (full example in tutorial section on h:dataTable)

```
<h:inputText value="#{programmer.level}" size="12"
  rendered="#{programmer.levelEditable}"/>
<h:commandButton value="Update"
  rendered="#{programmer.levelEditable}">
  <f:ajax render="@form" execute="@form"/>
</h:commandButton>
<h:outputText value="#{programmer.level}"
  rendered="#{!programmer.levelEditable}"/>
```

62

Example: Use of “rendered”

h:dataTable: Conditional Output & Editable Table Cells

You can use the “rendered” attribute to switch between output and input elements so as to make cells editable.

Company: My-Small-Company.com

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	(Edit? ☐) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? ☒) Junior	Java, C++, Python, and Go
Steve	Ballmer	(Edit? ☐) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	(Edit? ☒) Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	(Edit? ☐) Intermediate	REXX, CLIST, Java, PL/I, and COBOL

After clicking on checkbox

After typing in “Emeritus” and clicking “Update”

h:dataTable: Conditional Output & Editable Table Cells

You can use the “rendered” attribute to switch between output and input elements so as to make cells editable.

Company: My-Small-Company.com

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	(Edit? ☐) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? ☒) Junior	Java, C++, Python, and Go
Steve	Ballmer	(Edit? ☐) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	(Edit? ☒) Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	(Edit? ☐) Emeritus	REXX, CLIST, Java, PL/I, and COBOL

Conditional Text with ui:fragment

- **Idea**
 - Pass a boolean to the “rendered” attribute, put conditional content in body content. The value can be a literal string or an EL expression, and the literal string *can* contain HTML tags.
- **Example**
 - `<ui:fragment rendered="#{!status.last}">`
`<b>,</b>`
`</ui:fragment>` Outputs a bold comma after every entry except the last
- **When used**
 - When you are outputting literal HTML.
 - Can always be used in lieu of h:outputText, but if no HTML, h:outputText is more succinct.
- **Note: define the ui namespace at top**
 - `<html ... xmlns:ui="http://xmlns.jcp.org/jsf/facelets">`



Passing Arguments to Methods



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Big Idea

- **EL version 2.2 lets you call regular methods**
 - Rather than only zero-arg accessor methods
- **Syntax**
 - Basic syntax is straightforward
 - `#{someBean.someMethod(arg1, arg2)}`
 - The arguments can also be EL expressions
- **Cautions**
 - Use sparingly: put complexity in Java, not facelets
 - Works only in EL 2.2. **Not part of JSF 2.0 itself.**
 - Server must support servlets 3.0
 - All Java EE 6 servers automatically do
 - So, works in Glassfish 3, JBoss 6, and Tomcat 7.
Fails in Tomcat 6, JBoss 5, and other servlet 2.5 engines.

Method Args: Java Code

```
@ManagedBean
@ApplicationScoped
public class TestBean2 {
    private final String HELLO_ENGLISH = "Hello!";
    private final String HELLO_SPANISH = "¡Hola!";

    public String greeting(boolean useSpanish) {
        if (useSpanish) {
            return(HELLO_SPANISH);
        } else {
            return(HELLO_ENGLISH);
        }
    }

    public String greeting() {
        return(greeting(false));
    }

    public double randomNumber(double range) {
        return(range * Math.random());
    }
}
```

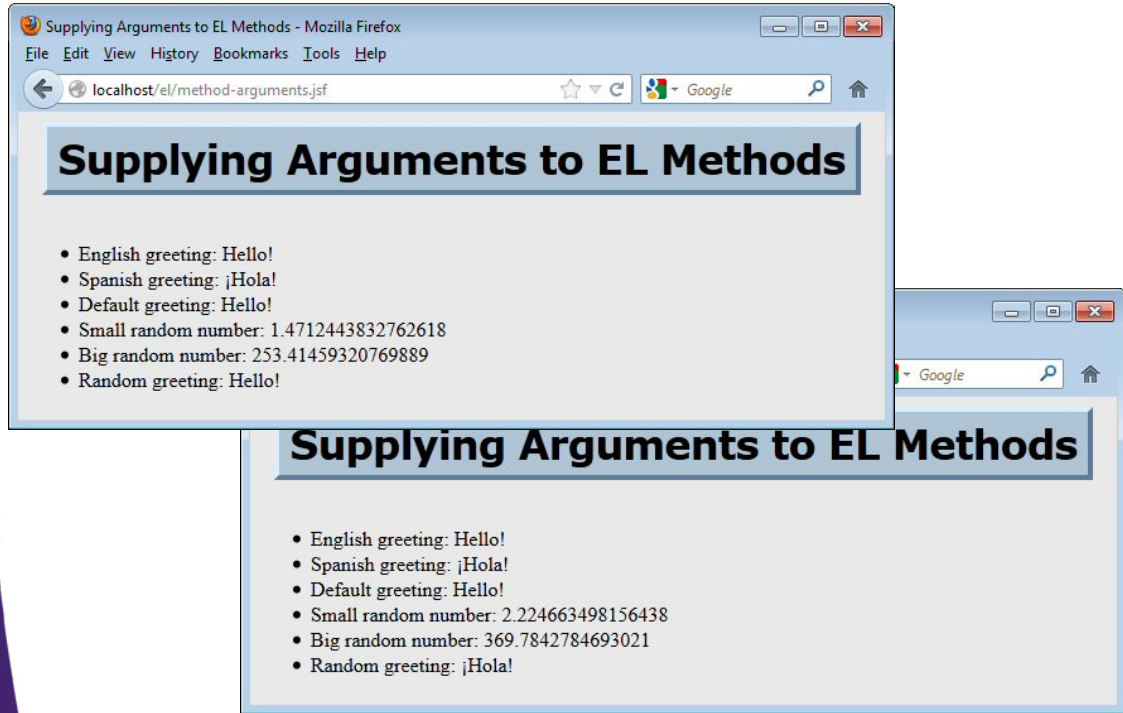
67

Method Args: Facelets Code

```
...
<ul>
    <li>English greeting: #{testBean2.greeting(false)}</li>
    <li>Spanish greeting: #{testBean2.greeting(true)}</li>
    <li>Default greeting: #{testBean2.greeting()}</li>
    <li>Small random number: #{testBean2.randomNumber(5)}</li>
    <li>Big random number: #{testBean2.randomNumber(500)}</li>
    <li>Random greeting:
        #{testBean2.greeting(testBean2.randomNumber(200) gt 100)}
    </li>
</ul>
...
```

68

Method Args: Results



The screenshot shows a Mozilla Firefox browser window with the address bar displaying 'localhost/el/method-arguments.jsf'. The page title is 'Supplying Arguments to EL Methods'. The content area lists the following results:

- English greeting: Hello!
- Spanish greeting: ¡Hola!
- Default greeting: Hello!
- Small random number: 1.4712443832762618
- Big random number: 253.41459320769889
- Random greeting: Hello!

A second, partially visible browser window in the background shows the same page title and a different set of results:

- English greeting: Hello!
- Spanish greeting: ¡Hola!
- Default greeting: Hello!
- Small random number: 2.224663498156438
- Big random number: 369.7842784693021
- Random greeting: ¡Hola!

69

© 2015 [Marty Hall](#)



Wrap-Up



70

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Summary

- **Outputting bean properties**
 - `#{customer.company.name}`
 - `<h:outputText value="#{customer.company.name}"/>`
 - `h:outputText` needed only when using `rendered`, `escape`, etc.
- **Textfields and other input elements**
 - `<h:inputText value="#{customer.firstName}"/>`
 - When form displayed, calls `getFirstName`
 - When form submitted, passes value to `setFirstName`
- **Collections**
 - `#{customer.addresses[0].zip}`
 - Call `getAddresses`, index into array or list, call `getZip`
 - See also separate tutorial section on looping
- **Operators, conditional evaluation, args**
 - Use for display logic, not for things that could be in Java code

71

© 2015 Marty Hall



Questions?

More info:

<http://www.coreservlets.com/JSF-Tutorial/jsf2/> – JSF 2.2 tutorial

<http://www.coreservlets.com/JSF-Tutorial/primefaces/> – PrimeFaces tutorial

<http://courses.coreservlets.com/jsf-training.html> – Customized JSF and PrimeFaces training courses

<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training



72

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



h:dataTable – Building HTML Tables from Collections

JSF 2.2 Version

Originals of slides and source code for examples: <http://www.coreservlets.com/JSF-Tutorial/jsf2/>

Also see the PrimeFaces tutorial – <http://www.coreservlets.com/JSF-Tutorial/primefaces/>
and customized JSF2 and PrimeFaces training courses – <http://courses.coreservlets.com/jsf-training.html>

Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live training on JSF 2, PrimeFaces, or other
Java EE topics, email hall@coreservlets.com**
Marty is also available for consulting and development support

**Taught by the author of *Core Servlets and JSP*, this tutorial,
and JSF 2.2 version of *Core JSF*. Available at public venues, or
customized versions can be held on-site at your organization.**

- Courses developed and taught by Marty Hall
 - JSF 2, PrimeFaces, Ajax, jQuery, Spring MVC, JSP, Android, general Java, Java 8 lambdas/streams, GWT, custom topic mix
 - Courses available in any location worldwide. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Hadoop, Spring, Hibernate/JPA, RESTful Web Services

Contact hall@coreservlets.com for details



Topics in This Section

- **Options for handling variable-length data**
 - Building HTML from a bean property
 - Using a builtin component like h:dataTable
 - Making your own composite component
 - Looping with ui:repeat
- **Using h:dataTable**
 - Basics: h:dataTable and h:Column
 - Headings
 - Style sheets
 - Ajax-enabled tables
 - Tables with conditional values

5

© 2015 Marty Hall



Overview



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Motivation for h:dataTable

- **Goal**

- You want results pages to be simple and HTML-oriented
 - Separation of concerns
 - Allows Web page developers to build GUI

- **Problem**

- What if the action controller method produces data whose length can change? How do you generate output without resorting to JSP scripting and explicit Java looping?

- **Solutions**

- There are a number of alternatives, *all* of which would work well in some circumstances. The issue is how much control the Web page author needs.
 - This section covers h:dataTable, but other alternatives are covered in other tutorial sections

7

JSF Constructs for Handling Variable-Length Data

Simplest
for Page
Author



Most
Control
for Page
Author

- **Bean**

- Have bean getter method spit out string or very simple HTML based on a collection

- **h:dataTable**

- Use a builtin component that builds a table from a collection.

- **Your own composite component**

- Make own component that builds some HTML construct (e.g., list) from a collection

- **ui:repeat**

- Do explicit looping in results page

8

Which Option to Use

- **In general**
 - Use the simplest option that gives the Web page author enough control
- **h:dataTable**
 - The tutorial section on ui:repeat compares and contrasts the options with a brief example of each.
 - Bottom line of the comparison:
h:dataTable is usually best when
 - You want to produce an HTML table from the collection
 - Each entry in the collection corresponds to a table row in a relatively consistent manner
- **Other options**
 - There are separate tutorial sections on ui:repeat and composite components

9

Example Notes

- **Data**
 - Normally, the data is produced in the business logic that is called by the action controller
 - E.g., you collect a bank customer ID and month in a form, and the button says
`<h:commandButton ... action="#{user.findChanges}"/>`
where findChanges finds the bank account changes (deposits and withdrawals) in the month and puts them into an array or List
 - Here, we will hardcode the data for simplicity
 - I.e., make a managed bean with a method that returns a fixed collection

10



Basics



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Simplest Syntax

- **Attributes**
 - var, value, border
- **Nested element**
 - h:column
- **Example**

```
<h:dataTable var="someVar" value="#{bean.someCollection}"
            border="...">
    <h:column>#{someVar.property1}</h:column>
    <h:column>#{someVar.property2}</h:column>
    ...
</h:dataTable>
```
- **Legal types for “value” attribute**
 - **Array**, **List**, Collection, ResultSet, Result, DataModel
 - Never use ResultSet: breaks “business logic” separation discussed in managed beans section.
 - JSF 2.3 promises to also support Map and Iterable

Supporting Class: Programmer

```
public class Programmer {
    private String firstName, lastName, level;
    private double salary;
    private String[] languages;

    public Programmer(String firstName,
                      String lastName,
                      String level,
                      double salary,
                      String... languages) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.level = level;
        this.salary = salary;
        this.languages = languages;
    }

    // getFirstName, getLastName, getLevel, getSalary,
    // getLanguages
}
```

13

Programmer (Continued)

```
/** Formats the salary like "$24,567.89". */
public String getFormattedSalary() {
    return(String.format("%.2f", salary));
}

/** Returns a String like "Java, C++, and Lisp".
 * That is, it puts commas and the word "and" into list.
 */
public String getLanguageList() {
    StringBuilder langList = new StringBuilder();
    for(int i=0; i<languages.length; i++) {
        if(i < (languages.length-1)) {
            langList.append(languages[i] + ", ");
        } else {
            langList.append("and " + languages[i]);
        }
    }
    return(langList.toString());
}
```

14

Supporting Class: Company

```
public class Company {
    private String companyName;
    private Programmer[] programmers;

    public Company(String companyName,
                    Programmer... programmers) {
        this.companyName = companyName;
        this.programmers = programmers;
    }

    public String getCompanyName() {
        return (companyName);
    }

    public Programmer[] getProgrammers() {
        return (programmers);
    }
}
```

15

Managed Bean: Company1

```
@ManagedBean
@ApplicationScoped
public class Company1 extends Company {
    public Company1() {
        super("My-Small-Company.com",
            new Programmer("Larry", "Ellison", "Junior", 34762.52,
                           "SQL", "Prolog", "OCL", "Datalog"),
            new Programmer("Larry", "Page", "Junior", 43941.86,
                           "Java", "C++", "Python", "Go"),
            new Programmer("Steve", "Ballmer", "Intermediate", 83678.29,
                           "Visual Basic", "VB.NET", "C#", "Visual C++",
                           "Assembler"),
            new Programmer("Sam", "Palmisano", "Intermediate", 96550.03,
                           "REXX", "CLIST", "Java", "PL/I", "COBOL"),
            new Programmer("Steve", "Jobs", "Intermediate", 103488.80,
                           "Objective-C", "AppleScript", "Java", "Perl",
                           "Tcl"));
    }
}
```

This data never changes, so you might as well make it application scoped.

16

Facelets Page: Top

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head><title>h:dataTable Basics</title>
  <link href="./css/styles.css"
        rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
```

This first simple example doesn't use any f: tags, but most real h:dataTable examples use f:facet. So, plan ahead and add this namespace from the beginning.

You probably are already using this namespace for h:form, h:outputText, etc. But even if not, you need it for h:dataTable and h:column.

17

Facelets Page: Main Code

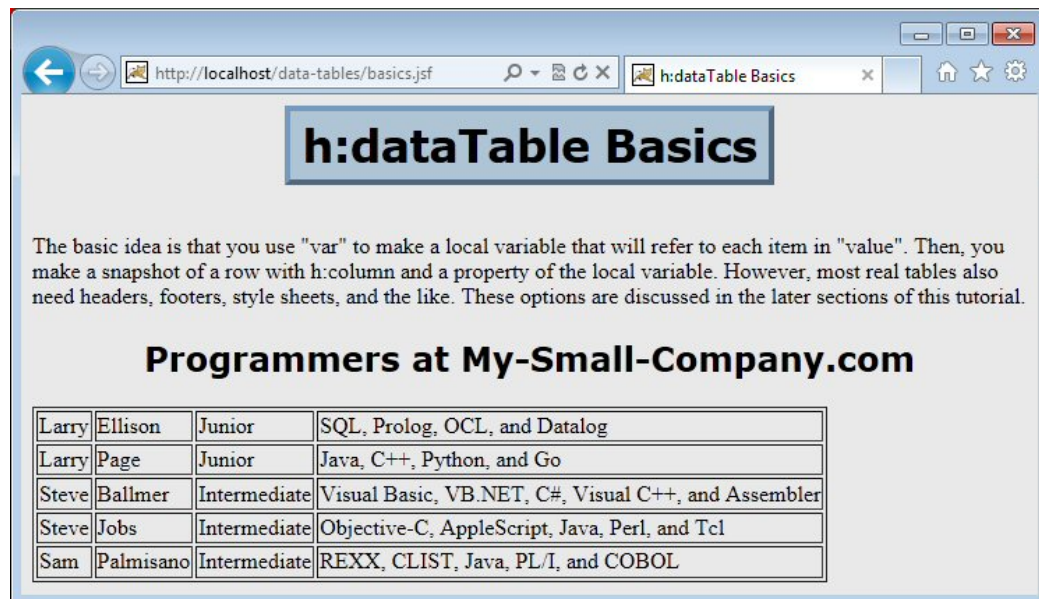
```
<h2>Programmers at #{company1.companyName}</h2>

<h:dataTable var="programmer"
              value="#{company1.programmers}"
              border="1">
  <h:column>#{programmer.firstName}</h:column>
  <h:column>#{programmer.lastName}</h:column>
  <h:column>#{programmer.level}</h:column>
  <h:column>#{programmer.languageList}</h:column>
</h:dataTable>
```

The border attribute of h:dataTable just becomes the border attribute of table.

18

Results



The basic idea is that you use "var" to make a local variable that will refer to each item in "value". Then, you make a snapshot of a row with h:column and a property of the local variable. However, most real tables also need headers, footers, style sheets, and the like. These options are discussed in the later sections of this tutorial.

Programmers at My-Small-Company.com

Larry	Ellison	Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	Junior	Java, C++, Python, and Go
Steve	Ballmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tcl
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL

19

© 2015 [Marty Hall](#)



Headers and Captions



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **Problem**

- Regular content gives a snapshot of a row. So, most content inside `h:column` is repeated for every row.

- **Solution**

- Mark headings with `f:facet`. Shown for first row only.

```
<h:dataTable var="someVar" value="#{someCollection}">
  <h:column>
    <f:facet name="header">First Heading</f:facet>
    #{someVar.property1}
  </h:column>
  ...
</h:dataTable>
```

This is a reserved name, not something arbitrary.
So, it will fail if you use "heading", "Header", or "headers".

- **Other `f:facet` options**

- `name="footer"`, `name="caption"` (outside `h:caption`)

21

Other Options for `h:dataTable`

- **Summary**

```
<h:dataTable var="..." value="..." otherAttributes>...</h:dataTable>
```

- **Most important other attributes**

- `border`, `bgcolor`, `cellpadding`, `cellspacing`, `width`, `onclick`, `ondblclick`, `onmouseover`, etc.
 - Same as for the normal `<table>` tag.
- `styleClass`, `captionClass`, `headerClass`, `footerClass`
 - CSS style names for main table, caption, header, and footer. Discussed in upcoming section.
- `rowClasses`, `columnClasses`
 - List of CSS style names for each row and column. Will use each name in turn and then repeat. Discussed in upcoming section.
- `first`, `rows`
 - First entry in collection to use, total number of rows to show.
- `id`, `rendered`
 - Same as for all `h:elements`. Use `id` for Ajax. Use `rendered` if you will omit the table in certain situations (e.g., if no rows).

22

Facelets Page: Main Code

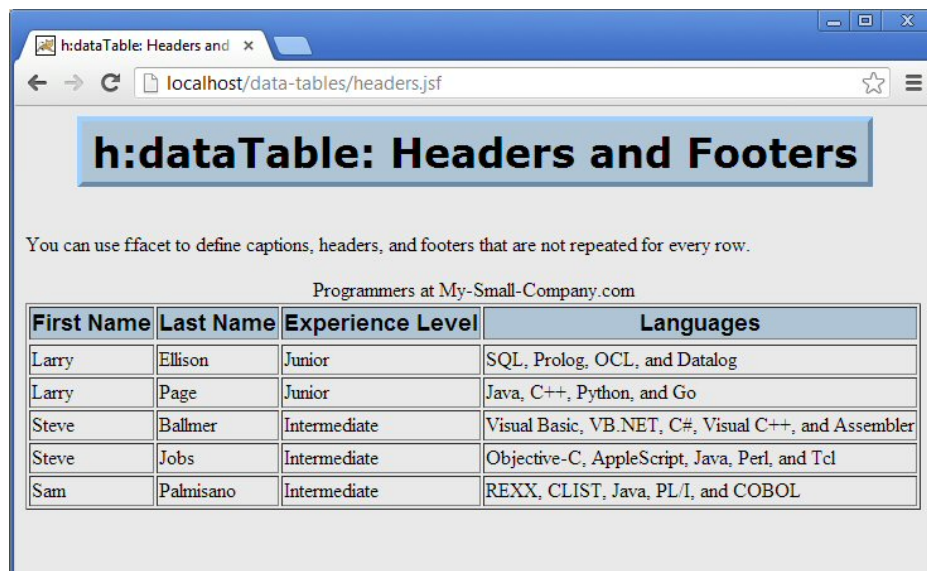
```
<h:dataTable var="programmer"
             value="#{company1.programmers}" border="1">
  <f:facet name="caption">Programmers at
                        #{company1.companyName}</f:facet>

  <h:column>
    <f:facet name="header">First Name</f:facet>
    #{programmer.firstName}
  </h:column>
  <h:column>
    <f:facet name="header">Last Name</f:facet>
    #{programmer.lastName}
  </h:column>
  <h:column>
    <f:facet name="header">Experience Level</f:facet>
    #{programmer.level}
  </h:column>
  <h:column>
    <f:facet name="header">Languages</f:facet>
    #{programmer.languageList}
  </h:column>
</h:dataTable>
```

Note: same managed bean (Company1) and supporting class (Programmer) as previous example.

23

Results



h:dataTable: Headers and Footers

You can use ffacet to define captions, headers, and footers that are not repeated for every row.

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	Junior	Java, C++, Python, and Go
Steve	Ballmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tcl
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL

24



Style Sheets



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **Problem**
 - Using a general style sheet for the entire table does not always work
 - Might be multiple tables in the page that want different styles
 - Different rows might want different styles
- **Solution**
 - Explicitly supply CSS style names.
- **Summary**
 - styleClass, captionClass, headerClass, footerClass
 - CSS name that will apply to table as a whole, caption (if there is one), first row (if it is a header), and last row (if it is a footer)
 - rowClasses
 - Comma-separated list of names. Apply first name to first non-header row or column, then next name, etc. *When you get to end of list, repeat.* For instance, two names will apply to odd/even rows.
 - columnClasses
 - Comma-separated list of names. Apply until you run out, then stop. *Do not repeat as with rowClasses.* Thus, you normally supply exactly as many entries as you have columns.

Example: Summary

```
<h:dataTable var="someVar" value="#{someValue}"
    styleClass="tableStyle"
    captionClass="captionStyle"
    headerClass="headerStyle"
    footerClass="footerStyle"
    rowClasses="row1,row2"
    columnClasses="col1,col2,col3">
    ...
</h:dataTable>
```

Overall table style. I.e., `<table class="tableStyle" ...>`

Style for caption (if any).

Style for first table row, if it is a heading (i.e., there is at least one f:facet with name="header").

Style for last table row, if it is a footer.

Apply row1 to first, third, fifth... non-heading rows.
Apply row2 to second, fourth, sixth... non-heading rows.

Apply col1 to first column, col2 to second column, col3 to third column.
If there are more than three columns, then no custom class is given to the later columns. Unlike rowClasses, these do not repeat.

27

Example: Facelets Part 1

```
<h:dataTable var="programmer"
    value="#{company1.programmers}"
    border="1"
    styleClass="mainTable"
    captionClass="caption1"
    headerClass="heading"
    rowClasses="even, odd">
    (Same body as last example)
</h:dataTable>
```

Fewer entries for rowClasses than rows in the table.
Will repeat once the end of the list of styles is reached.

Despite the simplicity and popularity of rowClasses, sometimes the same effect can be accomplished by using a single style sheet for the table as a whole, and applying the positional CSS selectors `nth-child(even)`, `nth-child(odd)`, or `nth-child(xn+y)`. These selectors are now supported consistently by all major browsers. See the CSS lecture earlier in this JSF tutorial series.

28

Example: Facelets Part 2

```
<h:dataTable var="programmer"
    value="#{myCompany.programmers}"
    border="1"
    styleClass="mainTable"
    captionClass="caption2"
    headerClass="heading"
    columnClasses="even, odd, even, odd">
```

Same number of entries for columnClasses as there are columns in the table. Unlike with rowClasses, will not repeat once the end of the list of styles is reached.

(Same body as last example)

```
</h:dataTable>
```

29

Example: Style Sheet Part 1

```
.mainTable {
    margin-left: auto;
    margin-right: auto;
}
.caption1 {
    font-family: sans-serif;
    font-weight: bold;
    font-size: 24px;
    color: blue;
}
.caption2 {
    font-family: sans-serif;
    font-weight: bold;
    font-size: 24px;
    color: red;
}
.heading {
    font-family: sans-serif;
    font-weight: bold;
    font-size: 20px;
    color: black;
    background-color: silver;
    text-align: center; }
```

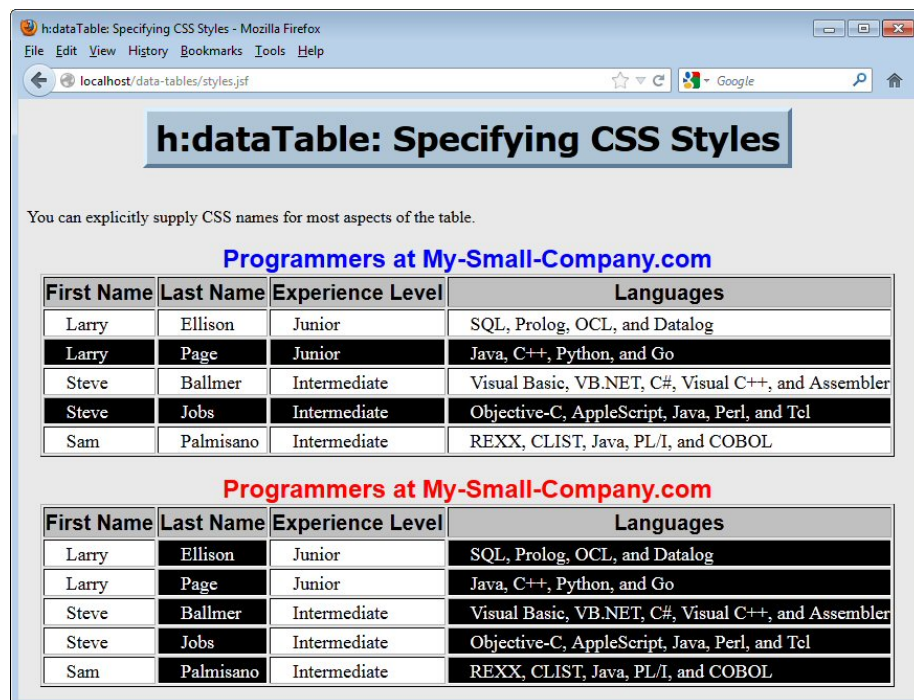
30

Example: Style Sheet Part 2

```
.even {
    font-family: Times New Roman, serif;
    font-size: 18px;
    color: black;
    background-color: white;
    text-indent: 20px;
}
.odd {
    font-family: Times New Roman, serif;
    font-size: 18px;
    color: white;
    background-color: black;
    text-indent: 20px;
}
```

31

Results



h:dataTable: Specifying CSS Styles

You can explicitly supply CSS names for most aspects of the table.

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	Junior	Java, C++, Python, and Go
Steve	Ballmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	Junior	Java, C++, Python, and Go
Steve	Ballmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL

32



Ajax-Enabled Tables



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **Problem**
 - Wasteful to reload entire page when switching among various tables
 - Especially if there is a lot of non-table output on the page
- **Solution**
 - Use f:ajax to rebuild the table without reloading the entire page.
- **Notes**
 - Only basic code shown here.
 - Full details on f:ajax are given in separate tutorial section.

Example Overview

- **CompanyInfo class**
 - Has list of available companies
 - Maps company names to companies
- **Company class**
 - Stores list of programmers working for the company
 - Shown earlier in this tutorial
- **Programmer class**
 - Stores name, skill level, and programming languages
 - Shown earlier in this tutorial
- **Desired behavior**
 - User chooses a company name and sees table of programmers that work for that company.
 - Table should be updated without reloading entire page

35

CompanyInfo

```
@ManagedBean
@ApplicationScoped
public class CompanyInfo {
    private String companyName;
    private Company[] companies = {
        new Company1(), new Company2(), new Company3() };
    private List<String> companyChoices;
    private Map<String, Company> companyMap;

    public CompanyInfo() {
        companyChoices = new ArrayList<>();
        companyMap = new HashMap<>();
        companyName = companies[0].getCompanyName();
        for(Company c: companies) {
            companyChoices.add(c.getCompanyName());
            companyMap.put(c.getCompanyName(), c);
        }
    }
}
```

This information does not change based on user input, so it is application scoped.

Once the user chooses a company name, it will be associated with the corresponding Company.

36

CompanyInfo (Continued)

Simple accessor methods.

```
public String getCompanyName() {
    return(companyName);
}

public void setCompanyName(String companyName) {
    this.companyName = companyName;
}

public List<SelectItem> getCompanyChoices() {
    return (companyChoices);
}

public Company getCompany() {
    return(companyMap.get(companyName));
}
}
```

37

Facelets Code

```
<div align="center"><b>Company:</b>
<h:selectOneMenu value="#{companyInfo.companyName}"
    <f:selectItems value="#{companyInfo.companyChoices}"/>
    <f:ajax render="programmerTable"/>
</h:selectOneMenu>
</div>
<p/>
<h:dataTable var="programmer"
    value="#{companyInfo.company.programmers}"
    border="1"
    id="programmerTable"
    styleClass="mainTable"
    captionClass="caption1"
    headerClass="heading"
    rowClasses="even,odd">
    (Table body almost identical to previous two examples)
</h:dataTable>
```

When the combobox value changes, fire off a behind-the-scenes JavaScript request to the server. Update the model data for the combobox (i.e., call setCompanyName). Then, return the data needed to rebuild the element with the id "programmerTable". Replace value for that element only, without reloading entire page.

38

Results

h:dataTable: Ajax-Enabled Tables

You can update a table without reloading the entire page.

Company: My-Small-Company.com

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	Junior	Java, C++, Python, and Go
Steve	Ballmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	Emeritus	REXX, CLIST, Java, PL/I, and COBOL

h:dataTable: Ajax-Enabled Tables

You can update a table without reloading the entire page.

Company: Some-Other-Company.com

Programmers at Some-Other-Company.com

First Name	Last Name	Experience Level	Languages
Harry	Hacker	Junior	Java, C++, and JavaScript
Polly	Programmer	Intermediate	Java, C++, JavaScript, and Ruby
Codie	Coder	Senior	C#, Java, Python, and JavaScript

h:dataTable: Ajax-Enabled Tables

You can update a table without reloading the entire page.

Company: Third-Company.com

Programmers at Third-Company.com

First Name	Last Name	Experience Level	Languages
Stephen	Harper	Intermediate	C#, Java, JavaScript, and Perl
Barack	Obama	Junior	Lisp, Java, and C++
Enrique	Peña Nieto	Senior	Python, Java, Ruby, and JRuby

39

© 2015 Marty Hall



Editable Table Cells



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **Problem**

- You would like to let end user edit some table cells. However, you only want certain cells editable, and want a nearby “Update” button when it is.

- **Solution**

- When user clicks “Edit” checkbox, change the output from simple text to a textfield followed by an update button. Use Ajax to prevent reloading entire page.

- **Main point of example**

- h:dataTable cell values can use normal JSF constructs
 - Conditional text
 - h:inputText, h:commandButton and any other elements. Not limited to simple #{blah} output

41

Java Classes

- **CompanyInfo**

- Shown earlier

- **Company**

- Shown earlier

- **Programmer**

- Partially shown earlier.
 - Added a boolean property to say whether the skill level is editable.
 - Added code to setLevel that resets the boolean property to false.

42

Programmer Class (New Parts)

```
public class Programmer {  
    ...  
    private boolean isLevelEditable;  
    ...  
    public void setLevel(String level) {  
        isLevelEditable = false;  
        this.level = level;  
    }  
    ...  
    public boolean isLevelEditable() {  
        return(isLevelEditable);  
    }  
  
    public void setLevelEditable(boolean isLevelEditable) {  
        this.isLevelEditable = isLevelEditable;  
    }  
}
```

43

Facelets Code

```
<h:selectOneMenu ...>...</h:selectOneMenu>  
...  
<h:dataTable ...>  
    ... (Most columns same as previous examples)  
    <h:column>  
        <f:facet name="header">Experience Level</f:facet>  
        (<i>Edit?  
        <h:selectBooleanCheckbox value="#{programmer.levelEditable}">  
            <f:ajax render="@form"/>  
        </h:selectBooleanCheckbox></i>)  
        <h:inputText value="#{programmer.level}" size="12"  
            rendered="#{programmer.levelEditable}"/>  
        <h:commandButton value="Update" rendered="#{programmer.levelEditable}">  
            <f:ajax render="@form" execute="@form"/>  
        </h:commandButton>  
        <h:outputText value="#{programmer.level}"  
            rendered="#{!programmer.levelEditable}"/>  
    </h:column>  
    ... (Most columns same as previous examples)  
</h:dataTable>
```

Core h:selectOneMenu and h:dataTable code is about the same as the previous example.

If the checkbox is checked, these two elements (input field and Ajax-enabled submit button) are shown.

If the checkbox is not checked, this one element (giving simple output) is shown instead.

44

Results

h:dataTable: Conditional Output & Editable Table Cells

You can use the "rendered" attribute to switch between output and input elements so as to make cells editable.

Company: My-Small-Company.com

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	(Edit? ☐) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? ☒) Junior	Java, C++, Python, and Go
Steve	Ballmer	(Edit? ☐) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	(Edit? ☒) Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	(Edit? ☐) Intermediate Update	REXX, CLIST, Java, PL/I, and COBOL

After clicking on checkbox

After typing in "Emeritus"
and clicking "Update"

h:dataTable: Conditional Output & Editable Table Cells

You can use the "rendered" attribute to switch between output and input elements so as to make cells editable.

Company: My-Small-Company.com

Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	(Edit? ☐) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? ☒) Junior	Java, C++, Python, and Go
Steve	Ballmer	(Edit? ☐) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Steve	Jobs	(Edit? ☒) Intermediate	Objective-C, AppleScript, Java, Perl, and Tel
Sam	Palmisano	(Edit? ☐) Emeritus	REXX, CLIST, Java, PL/I, and COBOL

45

© 2015 Marty Hall



Wrap-Up



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Syntax Summary

```
<h:dataTable var="row"           Variable name to refer to each entry in collection
              value="#{someBean.someData}" Collection (List, array, or a few other types)
              border="1"         It takes effort to use CSS to give table borders, so "border" is supplied as shortcut
              styleClass="css-table-style" CSS style for main table (<table class="...">)
              headerClass="css-heading-style" CSS for 1st row, if there is a header defined below
              rowClasses="evenRow, oddRow"> CSS styles for remaining rows, applied alternately

<h:column>
  <f:facet name="header">Col 1 Title</f:facet> Text for first col in first row
  #{row.someProperty} Value for first col in all subsequent rows
</h:column>
<h:column>
  <f:facet name="header">Col 2 Title</f:facet> Text for second col in first row
  #{row.someOtherProperty} Value for second col in all subsequent rows.
                           You can also use other JSF constructs here.
</h:column>
...
</h:dataTable>
```

47

© 2015 Marty Hall



Questions?

More info:

<http://www.coreservlets.com/JSF-Tutorial/jsf2/> – JSF 2.2 tutorial

<http://www.coreservlets.com/JSF-Tutorial/primefaces/> – PrimeFaces tutorial

<http://courses.coreservlets.com/jsf-training.html> – Customized JSF and PrimeFaces training courses

<http://coreservlets.com/> – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training



Customized Java EE Training: <http://courses.coreservlets.com/>

Java 7, Java 8, JSF 2, PrimeFaces, Android, JSP, Ajax, jQuery, Spring MVC, RESTful Web Services, GWT, Hadoop

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.