



# h:dataTable – Building HTML Tables from Collections

Originals of Slides and Source Code for Examples:  
<http://www.coreservlets.com/JSF-Tutorial/jsf2/>

**Customized Java EE Training:** <http://courses.coreservlets.com/>  
Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live training on JSF 2.x, please see courses at <http://courses.coreservlets.com/>.**



**Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. 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, servlets/JSP, Ajax, jQuery, Android development, Java 6 or 7 programming, custom mix of topics
    - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, etc.) or survey several
  - Courses developed and taught by coreservlets.com experts (edited by Marty)
    - Spring, Hibernate/JPA, EJB3, GWT, Hadoop, SOAP-based and RESTful Web Services
- Contact [hall@coreservlets.com](mailto: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

© 2012 Marty Hall



## Overview

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


Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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 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., <ul> 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 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 containing a collection

10



# Basics

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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="#{someCollection}"
             border="...">
  <h:column>#{someVar.property1}</h:column>
  <h:column>#{someVar.property2}</h:column>
  ...
</h:dataTable>
```
- **Legal types for “value” attribute**
  - **Array**, **List**, ResultSet, Result, DataModel
    - Result is from JSTL ; DataModel is part of JSF

# 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 // This data never changes, so you might as make it application scoped.
@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", "Balmer", "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"));
    }
}
```

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://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/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

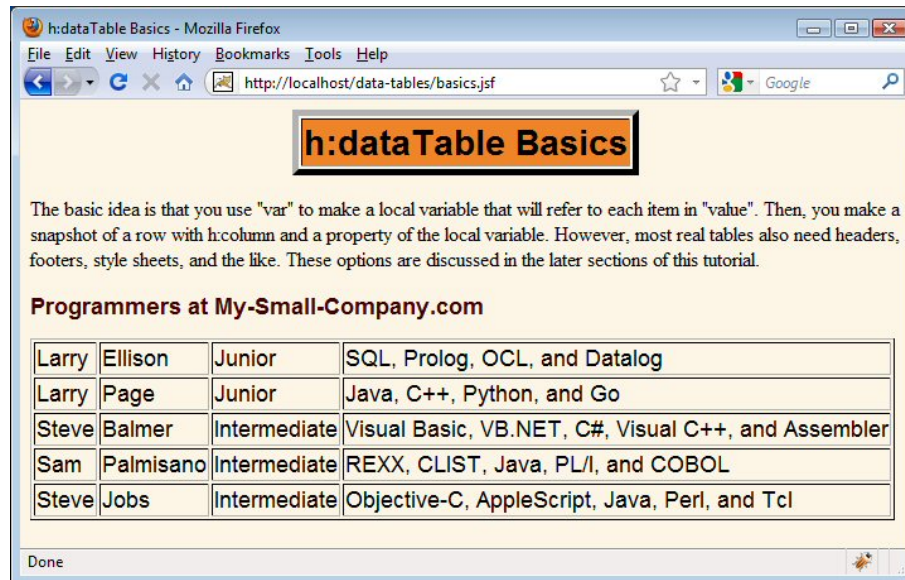
# Facelets Page: Main Code

```
<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



h:dataTable Basics

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	Balmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tcl

Done

19

© 2012 Marty Hall



# Headers and Captions

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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>
```

- **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

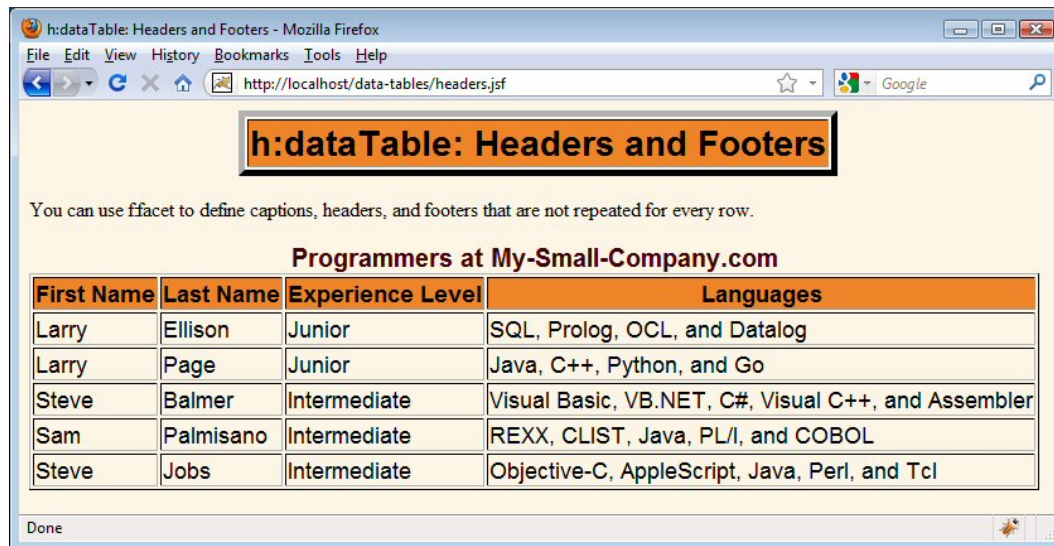
```
<h:dataTable var="programmer"
             value="#{company1.programmers}" border="1">
  <f:facet name="caption">Programmers at
                        #{company1.companyName}</f:facet>

  <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 facet 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	Balmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tcl

Done

24



# Style Sheets

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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.

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 body as last example)*

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

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.

## 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; }
```

# 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	Balmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tel

**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	Balmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tel

Done

32



# Ajax-Enabled Tables

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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<SelectedItem> companyChoices;
    private Map<String, Company> companyMap;

    public CompanyInfo() {
        companyChoices =
            new ArrayList<SelectedItem>();
        companyMap =
            new HashMap<String, Company>();
        companyName = companies[0].getCompanyName();
        for(Company c: companies) {
            SelectItem menuChoice =
                new SelectItem(c.getCompanyName());
            companyChoices.add(menuChoice);
            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.

JSF comboboxes (i.e., h:selectOneMenu) take an array or List of SelectItems.

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	Balmer	Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	Intermediate	Objective-C, AppleScript, Java, Perl, and Tcl

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

39

© 2012 Marty Hall



# Editable Table Cells

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.  
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?</i>  
        <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 and 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? <input type="checkbox"/> ) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? <input checked="" type="checkbox"/> ) Junior <input type="button" value="Update"/>	Java, C++, Python, and Go
Steve	Balmer	(Edit? <input type="checkbox"/> ) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	(Edit? <input checked="" type="checkbox"/> ) Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	(Edit? <input checked="" type="checkbox"/> ) Intermediate <input type="button" value="Update"/>	Objective-C, AppleScript, Java, Perl, and Tcl

Table Cells

45

### Programmers at My-Small-Company.com

First Name	Last Name	Experience Level	Languages
Larry	Ellison	(Edit? <input type="checkbox"/> ) Junior	SQL, Prolog, OCL, and Datalog
Larry	Page	(Edit? <input checked="" type="checkbox"/> ) Remedial	Java, C++, Python, and Go
Steve	Balmer	(Edit? <input type="checkbox"/> ) Intermediate	Visual Basic, VB.NET, C#, Visual C++, and Assembler
Sam	Palmisano	(Edit? <input checked="" type="checkbox"/> ) Intermediate	REXX, CLIST, Java, PL/I, and COBOL
Steve	Jobs	(Edit? <input type="checkbox"/> ) iGenius	Objective-C, AppleScript, Java, Perl, and Tcl

© 2012 Marty Hall



# Wrap-Up

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

Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android. 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

© 2012 Marty Hall



## Questions?

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

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