

# Java Server Faces (JSF)

# Table of Contents

# Chapter 6

# Data Tables

# Data Tables

The h:dataTable tag iterates over data to create an HTML table.

```
<h:dataTable value="#{items}" var="item">
    <h:column>
        <!-- left column components -->
        #{item.aPropertyName}
    </h:column>
    <h:column>
        <!-- next column components -->
        <h:commandLink value="#{item.anotherPropertyName}"
          action="..."/>
    </h:column>
    <!-- add more columns, as desired -->
</h:dataTable>
```

# Data Tables (cont'd)

- **This data (items) must be one of the following:**
  - An array
  - java.util.List
  - java.sql.ResultSet
  - javax.servlet.jsp.jstl.sql.Result
  - javax.faces.model.DataModel

# Simple Table Example

# Simple Table Example Cont.

# Simple Table Example Cont.

simple/web/index.xhtml

```
<h:body>
      #{msgs.pageTitle}
      <h:form>
            <h:dataTable value="#{tableData.names}" var="name">
                  <h:column>
                  #{name.last}
                  </h:column>
                  <h:column>
                  #{name.first}
                  </h:column>
            </h:dataTable>
      </h:form>
</h:body>
```

# Simple Table Example Cont.

simple/src/java/com/corejsf/Name.java

```
public class Name implements Serializable {
      private String first;
      private String last;

      public Name(String first, String last) {
      this.first = first;
      this.last = last;
       }
      // setters and getters
      public void setFirst(String newValue) { first = newValue; }
      public String getFirst() { return first; }
      public void setLast(String newValue) { last = newValue; }
      public String getLast() { return last; }
}
```

# Simple Table Example Cont.

simple/src/java/com/corejsf/TableData.java

```
@Named  // or @ManagedBean
@SessionScoped

public class TableData implements Serializable {

      private static final Name[ ] names = new Name[ ] {
      new Name("William", "Dupont"),
      new Name("Anna", "Keeney"),
      new Name("Mariko", "Randor"),
      new Name("John", "Wilson")
       };

      public Name[ ] getNames() { return names;}
}
```

# <h:dataTable> Attributes

| Attribute | Description |
|---|---|
| bgcolor | Background color for the table |
| border | Width of the table's border |
| captionClass | The CSS class for the table caption |
| captionStyle | A CSS style for the table caption |
| cellpadding | Padding around table cells |
| cellspacing | Spacing between table cells |
| columnClasses | Comma-separated list of CSS classes for columns |
| dir | Text direction for text that does not inherit directionality; valid values: LTR (left to right) and RTL (right to left) |
| first | A zero-relative index of the first row shown in the table |
| footerClass | CSS class for the table footer |
| frame | Specification for sides of the frame surrounding the table; valid values: none, above, below, hsides, vsides, lhs, rhs, box, border |
| headerClass | CSS class for the table header |

# <h:dataTable> Attributes

| Attribute | Description |
|---|---|
| rowClasses | Comma-separated list of CSS classes for rows |
| rows | The number of rows displayed in the table, starting with the row specified with the first attribute; if you set this value to zero, all table rows will be displayed |
| rules | Specification for lines drawn between cells; valid values: groups, rows, columns, all |
| summary | Summary of the table's purpose and structure used for nonvisual feedback such as speech |
| var | The name of the variable created by the data table that represents the current item in the value |
| binding, id, rendered, styleClass, value | Basic |
| lang, style, title, width | HTML 4.0 |
| onclick, ondblclick, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup | DHTML events |

# <h:column> Attributes

- footerClass
- binding
- id
- Rendered
- styleClass
- value

# Headers and Footers

- We use facets to achieve this goal

```
<h:dataTable>
    <h:column>
        <f:facet name="header">
                <%-- header components go here --%>
        </f:facet>
        <%-- column components go here --%>
        <h:column>#{name.last}</h:column>
        <f:facet name="footer">
                <%-- footer components go here --%>
        </f:facet>
    </h:column>
    ...
</h:dataTable>
```

# Example

```
<h:dataTable value="#{tableData.names}" var="name">
  <h:column>
     <f:facet name="header">
         <h:outputText value="#{msgs.lastnamecol}"/>
     </f:facet>
     <h:outputText value="#{name.last}"/>
     <f:facet name="footer">
         <h:outputText value="#{msgs.alphanumeric}"/>
     </f:facet>
  </h:column>
```

- **TIP: To place multiple components in a table header or footer, you must group them in a container (h:panelGroup tag, h:panelGrid or h:dataTable)**

# <ui:repeat> Tag

- Instead of the h:dataTable tag, you can use the ui:repeat tag.
- xmlns:ui="http://java.sun.com/jsf/facelets"

```
<table>
    <ui:repeat value="#{tableData.names}" var="name">
        <tr>
            <td>#{name.last},</td>
            <td>#{name.first}</td>
        </tr>
    </ui:repeat>
</table>
```

# <ui:repeat> Tag Attributes

- The following attributes let you iterate over a subset of the collection:
  - offset is the index at which the iteration starts (default: 0)
  - step is the difference between successive index values (default: 1)
  - size is the number of iterations
    - (default: (size of the collection – offset) /step)

- Example:
  - <ui:repeat ... offset="10" step="2" size="5">

# <ui:repeat> Tag Attributes

- varStatus: sets a variable that reports on the iteration status.
- The iteration status has these properties Boolean properties even, odd, first, and last, which are useful for selecting styles.
- Integer properties:
  - index
  - begin
  - step
  - end.

# <ui:repeat> Tag Attributes

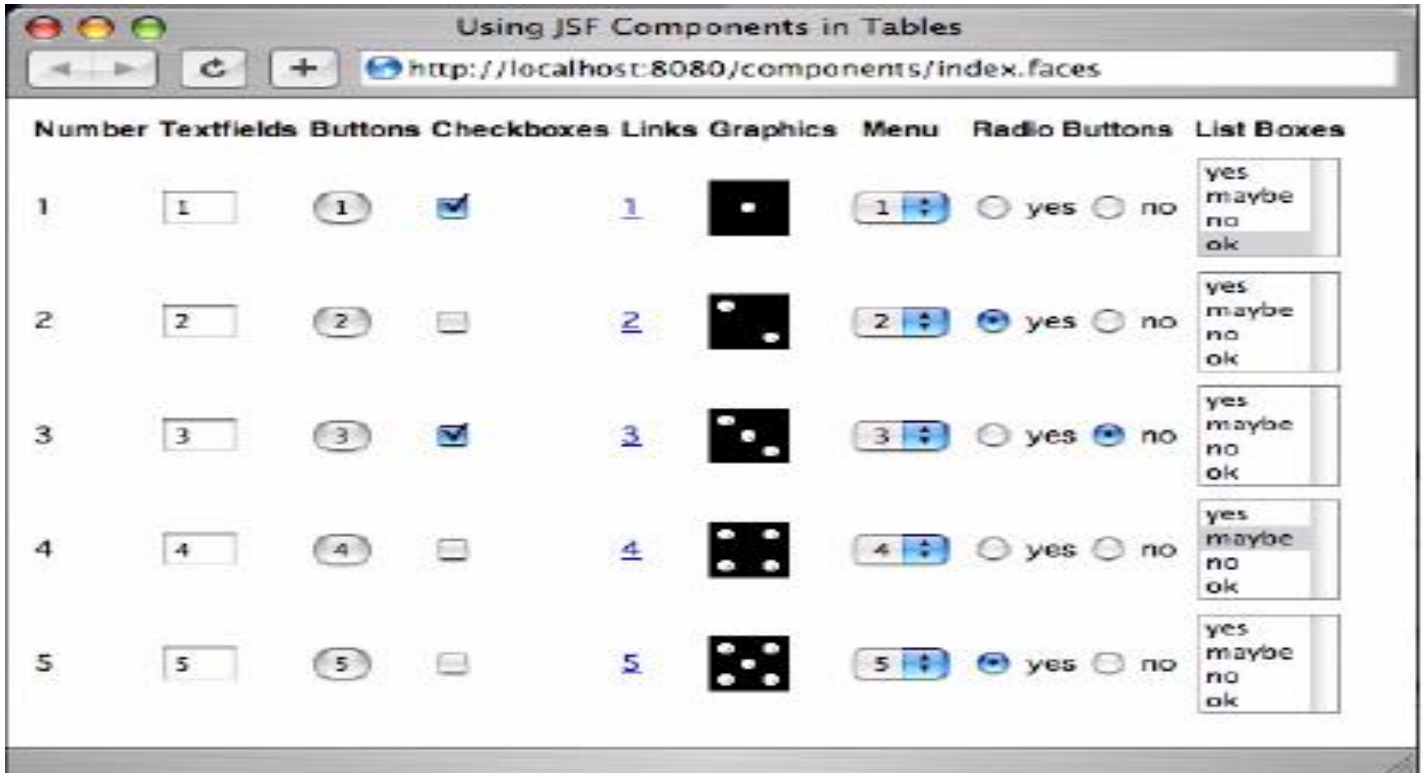
```
<table>
    <ui:repeat value="#{tableData.names}" var="name" varStatus="status">
        <tr>
            <td>#{status.index + 1}</td>
            <td>#{name.last},</td>
            <td>#{name.first}</td>
        </tr>
    </ui:repeat>
</table>
```

# JSF Components in Table Cells

- Any JSF component be placed in a table cell
  - outputText,
  - selectOneMenu,
  - selectOneRadio, graphicImage,
  - gridPanel

- JSF component in a cell can be manipulated just like a component outside of the table
  - associating an event handler
  - Associating validations

# JSF Components in Table Cells (cont'd)



# Styling of a Table

- Specify CSS classes as attributes of h:dataTable
  - **styleClass:** Table as a whole
  - **headerClass, footerClass:** Column headers and footers
  - **columnClasses:** Individual columns
  - **rowClasses:** Individual rows

# Database Tables

- Databases store information in tables, so the JSF data table component is a good fit for showing data stored in a database.

# Database Tables (Cont.)

```
<h:dataTable value="#{customerBean.all}" var="customer"
        styleClass="customers“ headerClass="customersHeader"
        columnClasses="custid,name">
    <h:column>
        <f:facet name="header">#{msgs.customerIdHeader}</f:facet>
        #{customer.Cust_ID}
    </h:column>
    <h:column>
        <f:facet name="header">#{msgs.nameHeader}</f:facet>
        #{customer.Name}
    </h:column>
    ...
</h:dataTable>
```

# Database Tables (Cont.)

- The value for h:dataTable is an instance of:
  - java.sql.ResultSet
  - javax.servlet.jsp.jstl.Result.
- Don’t use a result set returned from the Statement.executeQuery method.

# Database Tables (Cont.)

```
public ResultSet getAll() throws SQLException {
    Connection conn = ds.getConnection();
    try {
        Statement stmt = conn.createStatement();
        ResultSet result = stmt.executeQuery("SELECT * FROM Customers");
        // return ResultSupport.toResult(result);
        CachedRowSet crs = new com.sun.rowset.CachedRowSetImpl();
        // or use an implementation from your database vendor
        crs.populate(result);
        return crs;
    } finally {
        conn.close();
    }
}
```

# Table Models

- h:dataTable wraps java objects (*array, list, result set, or JSTL result* ) in a model that extends the javax.faces.model.DataModel class.

- Model classes:
  - ArrayDataModel
  - ListDataModel
  - ResultDataModel
  - ResultSetDataModel

# javax.faces.model.DataModel<E>

- int getRowCount()
  - Returns the total number of rows, if known; otherwise, it returns –1.
- int getRowIndex()
  - Returns the index of the current row.
- void setRowIndex(int index)
  - Sets the current row index and updates the scoped variable representing the current item in the collection (that variable is specified with the var attribute of h:dataTable).
- E getRowData()
  - Returns the data associated with the current row.

# Table Models Cont.

```
public class TableData implements Serializable {
    private static final Name[] names = new Name[] {
                    new Name("William", "Dupont"),
                    new Name("Anna", "Keeney"),
                    new Name("Mariko", "Randor"),
                    new Name("John", "Wilson")
            };

    private DataModel<Name> model = new ArrayDataModel<Name>(names);

    public DataModel<Name> getNames() {
                    return model;
            }
}
```

# Table Models Cont.

- **Rendering Row Numbers**
  - the DataModel class has a method getRowIndex that yields the current row number. You can access this method from a JSF page, as long as your application provides a table model instead of a collection.

```
<h:dataTable value="#{tableData.names}" var="name">
        <h:column>
                #{tableData.names.rowIndex + 1}
        </h:column>
        <h:column>#{name.last},</h:column>
        <h:column>#{name.first}</h:column>
</h:dataTable>
```

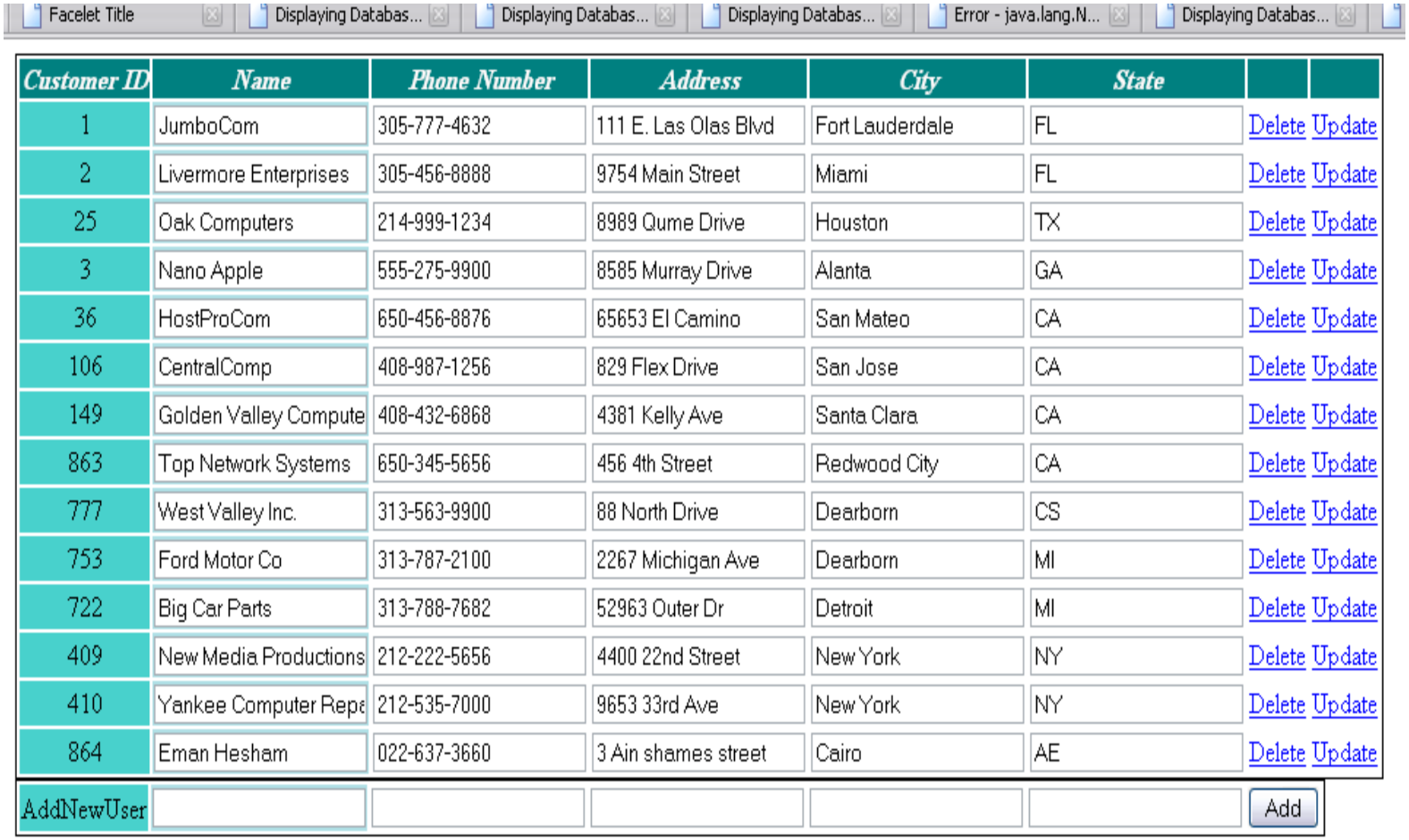
# Table Models Cont.

- Finding the Selected Row
  - you can retrieve the current item by calling the getRowData() method of the DataModel class.

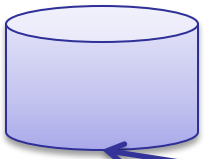
```
<h:dataTable value="#{tableData.names}" var="name">
        <h:commandLink value="Delete" action="{tableData.deleteRow}"/>
</h:dataTable>

public String deleteRow() {
        Name nameToDelete = model.getRowData();
        names.remove(nameToDelete);
        return null;
}
```

# Assignment

- Select data from database and display it into Table.
- Add functionality to add, edit and delete data from table.



# Assignment

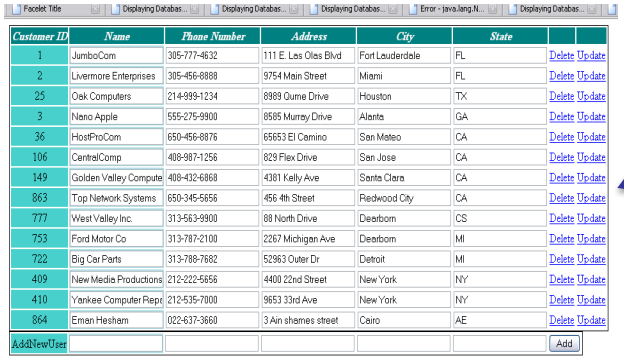


## Class Customer

{all fields as in Customer Table with setters & getters}

## Class CustomerDAO

- connect()
- ArrayList<Customer> getAllCustomers()
- int deleteCustomer(int id)
- int updateCustomer(Customer c)
- int addCustomer(Customer c)



| Customer ID | Name | Phone Number | Address | City | State | | |
|---|---|---|---|---|---|---|---|
| 1 | JumboCom | 305-777-4632 | 111 E. Las Olas Blvd | Fort Lauderdale | FL | Delete | Update |
| 2 | Livermore Enterprises | 305-456-8888 | 9754 Main Street | Miami | FL | Delete | Update |
| 25 | Oak Computers | 214-999-1234 | 8989 Gurne Drive | Houston | TX | Delete | Update |
| 3 | Nano Apple | 555-275-9900 | 8585 Murray Drive | Alanta | GA | Delete | Update |
| 36 | HostProCom | 650-456-8876 | 65653 El Camino | San Mateo | CA | Delete | Update |
| 106 | CentralComp | 408-987-1256 | 829 Flex Drive | San Jose | CA | Delete | Update |
| 149 | Golden Valley Compute | 408-432-6868 | 4381 Kelly Ave | Santa Clara | CA | Delete | Update |
| 863 | Top Network Systems | 650-345-5656 | 456 4th Street | Redwood City | CA | Delete | Update |
| 777 | West Valley Inc. | 313-563-9900 | 88 North Drive | Dearborn | CS | Delete | Update |
| 753 | Ford Motor Co | 313-787-2100 | 2267 Michigan Ave | Dearborn | MI | Delete | Update |
| 722 | Big Car Parts | 313-789-7682 | 52963 Outer Dr | Detroit | MI | Delete | Update |
| 409 | New Media Productions | 212-222-5656 | 4400 22nd Street | New York | NY | Delete | Update |
| 410 | Yankee Computer Repc | 212-535-7000 | 9653 33rd Ave | New York | NY | Delete | Update |
| 864 | Eman Hesham | 022-637-3660 | 3 Ain shames street | Cairo | AE | Delete | Update |
| AddNewUser | | | | | | Add | |

## Class CustomerBean (Session)

- DataModel<Customer> customers
- CustomerDAO dao
- Customer newUser

- update()
- delete()
- addCustomer()