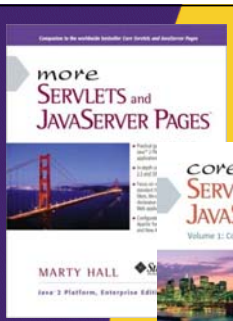
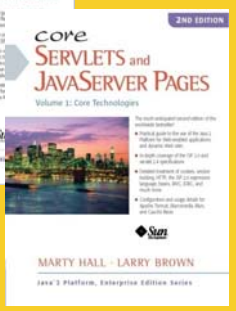
JSF 2 Composite Components
Part IV – Wrapping JavaScript GUIs

JSF 2.2 Version

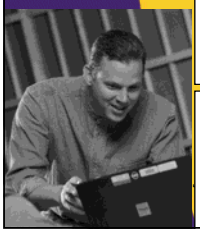
Originals of slides and source code for examples: http://www.coreservlets.com/JSF-Tutorial/jsf2/
Also see the PrimeFaces tutorial – http://www.coreservlets.com/JSF-Tutorial/primefaces/
and customized JSF2 and PrimeFaces training courses – http://courses.coreservlets.com/jsf-training.html





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

- Loading JavaScript libraries
- Wrapping up a jQuery-UI element as a JSF2 component
- Finding the real HTML id
- Escaping colons in JavaScript
- Passing arguments to the JavaScript

Review: Input Components

Input Components

- **Idea**
  - A composite component that will go inside h:form and be used to gather input from the user
- **Features**
  - Must be associated with a bean value
    - Use “type” attribute to enforce the type of the value
  - Often has its own style sheet
    - Load with h:outputStylesheet
  - May need to assemble final value out of submitted pieces
  - May need to call component setter methods based on attribute values
  - May have embedded Ajax behavior
  - May be built out of a rich JavaScript GUI element

Wrapping Up GUI Elements from JavaScript Libraries

Wrapping Up JavaScript GUI Elements: Idea

- **Motivation**
  - If you need a rich GUI element (slider, popup calendar, etc.) that is not already part of your JSF library
- **Options**
  - Look at existing component libraries
    - PrimeFaces, RichFaces, IceFaces, OpenFaces, Tomahawk, Trinidad, etc.
  - Use an element from jQuery UI, Ext/JS, Dojo, YUI etc.
    - But wrap it up so that it can be used like a normal JSF component.

Wrapping Up JavaScript GUI Elements: Steps

- **Load the core JavaScript library**
  - Use h:outputScript with target="head"
- **Load your own JavaScript file**
  - Again, use h:outputScript.
  - To avoid name conflicts, use namespaces for functions
    - Covered in next section
- **Designate the input element of interest**
  - By class, if all input elements will be treated the same
    - Covered in this section
  - By id, if you will pass custom arguments to each element
    - Must compute real HTML ID and handle the fact that JSF ids contain colons. Covered in next section.
- **Convert values**
  - Use standard converter (e.g., f:convertDateTime)
  - Make your own converter
    - Implement Converter, override getAsObject & getAsString

Example: Popup Calendar (Date Picker) from jQuery UI

- **Load jQuery and jQuery UI**

  <h:outputStylesheet name="jquery-ui.css" library="css/jquery-ui"/>
  <h:outputScript name="jquery.js" library="scripts" target="head"/>
  <h:outputScript name="jquery-ui.js" library="scripts" target="head"/>
- **Load custom JavaScript file**

  <h:outputScript name="jquery-ui-setup.js" library="scripts" target="head"/>
- **Give custom CSS class to h:inputText**
  - <h:inputText styleClass="date" …>
  - In setup file for on load: $(".date").datepicker();
- **Use Date/Time converter**

  <h:inputText styleClass="date" value="#{cc.attrs.value}">
  <f:convertDateTime pattern="MM/dd/yyyy"/>
  </h:inputText>

Component File

```
<cc:interface>
  <cc:attribute name="value" type="java.util.Date"/>
</cc:interface>

<cc:implementation>
  <h:outputStylesheet name="jquery-ui.css"
                      library="css/jquery-ui"/>
  <h:outputScript name="jquery.js" library="scripts"
                  target="head"/>
  <h:outputScript name="jquery-ui.js" library="scripts"
                  target="head"/>
  <h:outputScript name="jquery-ui-setup.js" library="scripts"
                  target="head"/>
  <h:inputText styleClass="date" value="#{cc.attrs.value}">
    <f:convertDateTime pattern="MM/dd/yyyy"/>
  </h:inputText>
</cc:implementation>
```

Custom JavaScript File

```
$(function() {
    $(".date").datepicker();
});
```

In jQuery, $(function() {…}) is like a smart window.onload function. It is smart in the sense that it does not overwrite existing onload functions, and it runs after the DOM is parsed but potentially before images are loaded. jQuery and jQuery UI are covered in separate sections in the Ajax/jQuery tutorial at coreservlets.com.

This means to find all HTML elements (textfields, in this case) that have class="date" and turn them into date pickers (i.e., textfields that contain dates and pop up calendars when you click in them).

Example Usage: Reservations at the JSF Resort

- **Idea**
  - Same as previous: let user make a hotel reservation.
    - But, use a jQuery date picker (popup calendar) instead of three separate select menus
- **Managed Bean**
  - Extends previous example, two changes:
    - setStartDate and setEndDate add one day to the date, since days of month in SimpleDateFormat start at 0, not 1
    - Adds a "2" to the end of the return value, since results page uses a different managed bean
- **Page that uses component**
  - Same as last example: collects name, checkin date, and checkout date.
- **Results page**
  - Same as previous example except for managed bean name

Managed Bean (First Half)

```
@ManagedBean
public class ResortBean2 extends ResortBean {

  @Override
  public void setStartDate(Date startDate) {
    super.setStartDate(DateUtils.nextDay(startDate));
  }

  @Override
  public void setEndDate(Date endDate) {
    super.setEndDate(DateUtils.nextDay(endDate));
  }
```

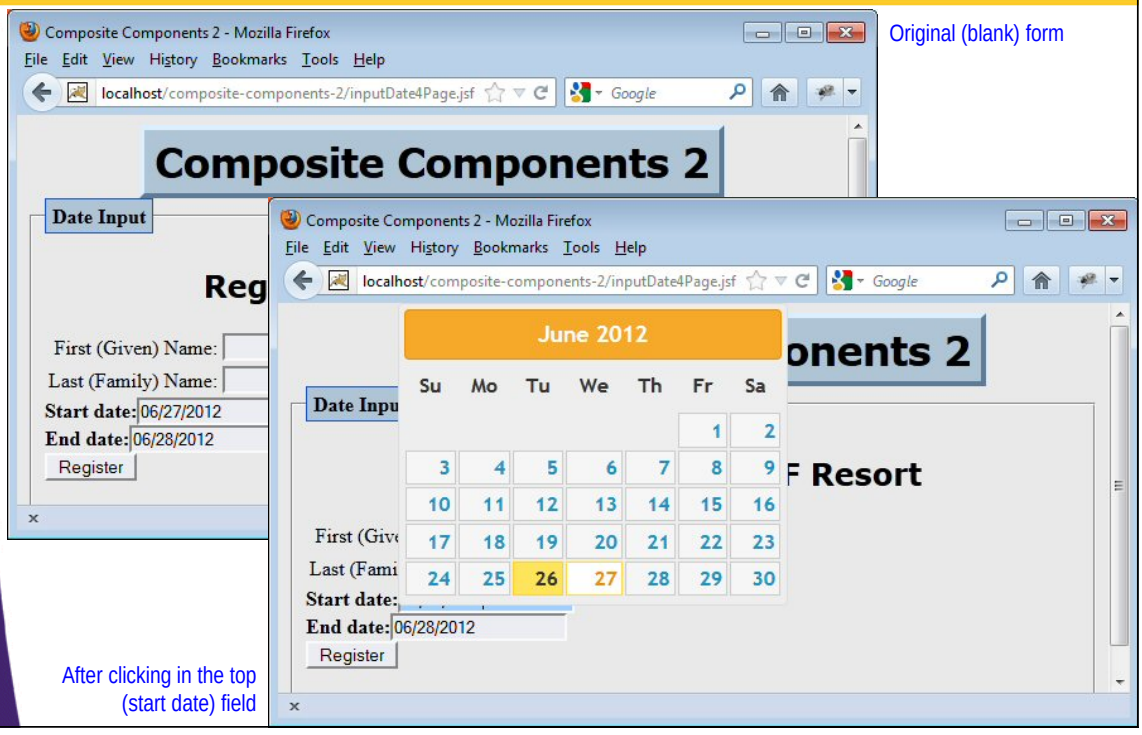

Managed Bean (Second Half)

```
  @Override
  public String register() {
    String returnValue = super.register();
    if (returnValue != null) {
      returnValue = returnValue + "2";
    }
    return(returnValue);
  }
}
```

Facelets Page that Uses Component

```
...
<h:form id="registrationForm">
   <h2>Register for the JSF Resort</h2>
   <utils:inputName2 value="#{resortBean2}"
                     firstNamePrompt="First (Given) Name"
                     lastNamePrompt="Last (Family) Name"/>
   <b>Start date:</b>
   <utils:inputDate4 value="#{resortBean2.startDate}"/><br/>
   <b>End date:</b>
   <utils:inputDate4 value="#{resortBean2.endDate}"
                     id="checkoutDate"/>
   <h:message for="registrationForm:checkoutDate"
              styleClass="error"/><br/>
   <h:commandButton action="#{resortBean2.register}"
                    value="Register"/><p/>
</h:form>
...
```


Result: Page that Uses Component



Original (blank) form

After clicking in the top (start date) field

Result: Results Page

Passing Arguments to JavaScript

Passing Arguments to JavaScript

- **Motivation**
  - Rather than making all date pickers look the same, you may want to collect component attributes (start date, end date, number of months to display, etc.) and pass those values to the underlying JavaScript.
- **Issues**
  - You need to find the real HTML id
    - Concatenate #{cc.clientId} with a colon and the id used in component file
  - In jQuery, you must escape the colons
    - Colons have special meaning in jQuery selector patterns, so you must replace each ":" with "\\:" before using the string as a jQuery selector

Passing Arguments: Code Summary

- **JavaScript code**

```
var coreservlets = {};
coreservlets.escapeColons = function(string) {
  return(string.replace(/:/g, "\\:"));
};
coreservlets.jQueryIdString = function(fieldId) {
  return("#" + coreservlets.escapeColons(fieldId));
};
```

- **Composite component**

```
<h:inputText value="#{cc.attrs.value}" id="date">…
<script type="text/javascript">
    $(function() {
        var fieldId = coreservlets.jQueryIdString("#{cc.clientId}" + ":date");
        $(fieldId).datepicker({ … });
    });
</script>
```

Options for jQuery UI GUI Elements

- **Without options**
  - $("some-selector").somemethod();
    - E.g., $("textfield-id").datepicker();
- **With options**
  - $("some-selector").somemethod({ option1: value1,
    option1: value2,
    …
    optionN: valueN });
- **Warnings**
  - Don't forget the curly braces
  - Misspelled option names are ignored – no error message
  - No trailing comma after the last option

Quick Example

```
$("#some-id").datepicker({
    changeMonth: true,
    changeYear: true,
    numberOfMonths: 2
});
```

Main Options for datepicker({…})

- **defaultDate (default: today)**
  - Initially selected date
- **changeMonth, changeYear (default: false)**
  - Should jQuery include a dropdown list to let you choose the month or year?
- **dayNames (default Sunday…Saturday) monthNames (default January…December)**
  - Names to use for the days and months. For other languages. There are also various options for short versions of days.
- **numberOfMonths (default: 1)**
  - How many months to show at a time

Component File: Interface Section

```
<cc:interface>
  <cc:attribute name="value" type="java.util.Date"/>
  <cc:attribute name="changeMonth" default="false"/>
  <cc:attribute name="changeYear" default="false"/>
  <cc:attribute name="numberOfMonths" default="1"/>
</cc:interface>
```

Component File: Implementation Section

```
<cc:implementation>
  <h:outputStylesheet name="jquery-ui.css" library="css/jquery-ui"/>
  <h:outputScript name="jquery.js" library="scripts" target="head"/>
  <h:outputScript name="jquery-ui.js" library="scripts" target="head"/>
  <h:outputScript name="jquery-ui-setup.js" library="scripts" target="head"/>
  <h:inputText value="#{cc.attrs.value}" id="date">
    <f:convertDateTime pattern="MM/dd/yyyy"/>
  </h:inputText>
  <script type="text/javascript">
   $(function() {
      var fieldId = coreservlets.jQueryIdString("#{cc.clientId}" + ":date");
      $(fieldId).datepicker({ changeMonth: #{cc.attrs.changeMonth},
                              changeYear: #{cc.attrs.changeYear},
                              numberOfMonths: #{cc.attrs.numberOfMonths}});
   });
  </script>
</cc:implementation>
```


Custom JavaScript File

```
var coreservlets = {};
```

This sets up an empty object to use as a namespace, to reduce the chance of name conflicts. The functions will be properties in that object. This is similar syntax to static methods in Java. See the coreservlets.com JavaScript tutorial for details on this technique.

```
coreservlets.escapeColons = function(string) {
  return(string.replace(/:/g, "\\:"));
};
```

Given "foo:bar:baz", returns "foo\:bar\:baz". For use in inputDate5, where each input element has its own id, but the id will contain ":" due to the JSF conventions. In jQuery, if you have <... id="foo:bar"/>, you cannot do $("#foo:bar") because ":" has special meaning. So, you must do $("#foo\:bar") instead.

```
coreservlets.jQueryIdString = function(fieldId) {
  return("#" + coreservlets.escapeColons(fieldId));
};
```

Given "foo:date", returns "#foo\:date". That is, prepends "#" on the front, then escapes colons. This yields an escaped version of the CSS id selector string that can be used by jQuery.

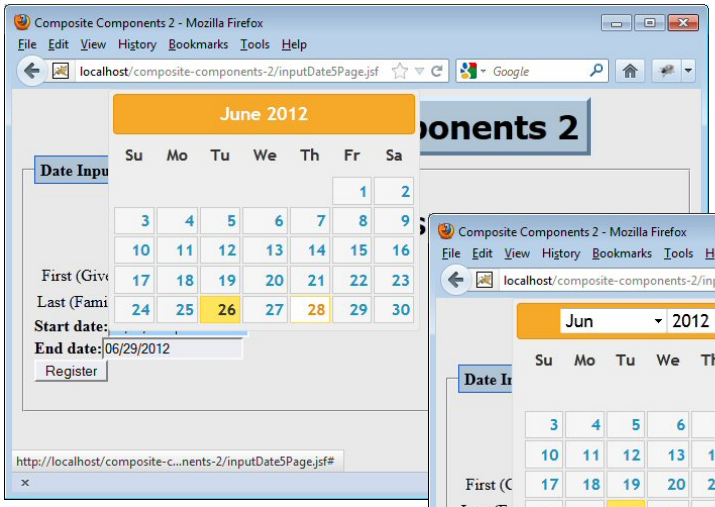
Example Usage: Reservations at the JSF Resort

- **Idea**
  - Same as previous: let user make a hotel reservation.
    - But, pass options to the datepicker function
- **Managed Bean**
  - Unchanged from previous example
- **Page that uses component**
  - Same as last example: collects name, checkin date, and checkout date.
    - But, supplies attributes that customize the number of months shown and whether a month or year menu is shown
- **Results page**
  - Unchanged from previous example

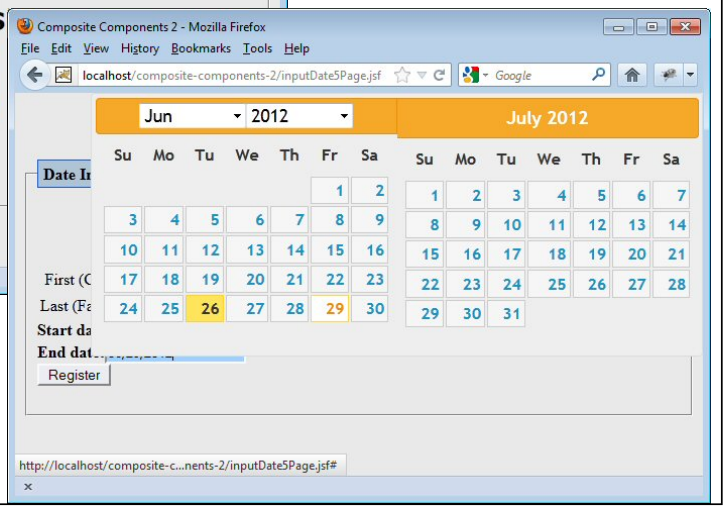
Facelets Page that Uses Component

```
...
<h:form id="registrationForm">
   <h2>Register for the JSF Resort</h2>
   <utils:inputName2 value="#{resortBean2}"
                     firstNamePrompt="First (Given) Name"
                     lastNamePrompt="Last (Family) Name"/>
   <b>Start date:</b>
   <utils:inputDate5 value="#{resortBean2.startDate}"/><br/>
   <b>End date:</b>
   <utils:inputDate5 value="#{resortBean2.endDate}"
                     changeMonth="true" changeYear="true"
                     numberOfMonths="2" id="checkoutDate"/>
   <h:message for="registrationForm:checkoutDate"
              styleClass="error"/><br/>
   <h:commandButton action="#{resortBean2.register}"
                    value="Register"/><p/>
</h:form>
...
```

Result: Page that Uses Component



Default date picker.



Date picker with number of months displayed set to 2, and month/year menus enabled.

Result: Results Page

Wrap-Up

Summary

- **Basics**
  - Load JavaScript via h:outputScript with target="head"
- **Passing arguments to JavaScript**
  - Find real (HTML) id with "#{cc.clientId}" + ":local-id"
  - Replace colons with "\\:"
  - Pass result to jQuery $ function
- **Caution**
  - Consider using existing component libraries (e.g., PrimeFaces, RichFaces) before wrapping your own components

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