The AjaxTags Library: Advanced Features

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

---

more SERVLETS and JavaServer Pages

core SERVLETS and JavaServer Pages
2ND EDITION
Volume 1: Core Technologies

MARTY HALL
Java 2 Platform, Enterprise Edition

MARTY HALL · LARRY BROWN
Java 2 Platform, Enterprise Edition Series

For live Ajax training, please see training 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
  – Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, customized mix of topics
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  – Spring, Hibernate, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- **Regions that display temporarily while server-side resource runs**
- **Autocomplete textboxes with associated values displayed in other textboxes**
- **Prefunctions and postfunctions:**
  - Arbitrary JavaScript that runs before or after server-side resource
- **Functions that run when server has error**
- **Multiple triggers for server-side resources**

Basic Setup: advanced.jsp

```
<!DOCTYPE ...>
<%@ taglib uri="http://ajaxtags.org/tags/ajax"
           prefix="ajax" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type"
      content="text/html;charset=UTF-8"" />
<jsp:include page="/WEB-INF/includes/header.jsp"/>
...
<title>Advanced AjaxTags Examples</title>
</head>
<body>
  ...
</body></html>
```

Basic Setup: header.jsp

```
<% request.setAttribute("contextPath",
                        request.getContextPath()); %>
<script src="${contextPath}/scripts/prototype-1.4.0.js"
        type="text/javascript"></script>
<script src="${contextPath}/scripts/scriptaculous.js"
        type="text/javascript"></script>
<script src="${contextPath}/scripts/overlibmws.js"
        type="text/javascript"></script>
<script src="${contextPath}/scripts/ajaxtags-1.2-beta2.js"
        type="text/javascript"></script>
<link rel="stylesheet"
      href="${contextPath}/css/styles.css"
      type="text/css"/>
```

---

ajax:autocomplete with indicator

Basic Idea

- **Approach**
  - indicator attribute designates section (usually div or span) that will be enabled while results are being loaded
    - It is disabled to start with, and disabled again when results come back
    - Often contains animated GIF showing progress
- **Main use**
  - Gives user feedback when server-side resource takes a long time

JSP Example

```
<fieldset>
  <legend>ajax:autocomplete with indicator</legend>
  <form>
    <label for="language">Programming language:</label>
    <input type="text" name="language" id="language"/>
    <span id="indicatorRegion" style="display:none;">
      <img src="${contextPath}/images/busy-indicator.gif"/>
      Loading...
    </span>
  </form>
  <ajax:autocomplete
    source="language"
    target="language"
    baseUrl="${contextPath}/slow-language-completer.ajax"
    className="autocomplete"
    minimumCharacters="1"
    indicator="indicatorRegion"/>
</fieldset>
```
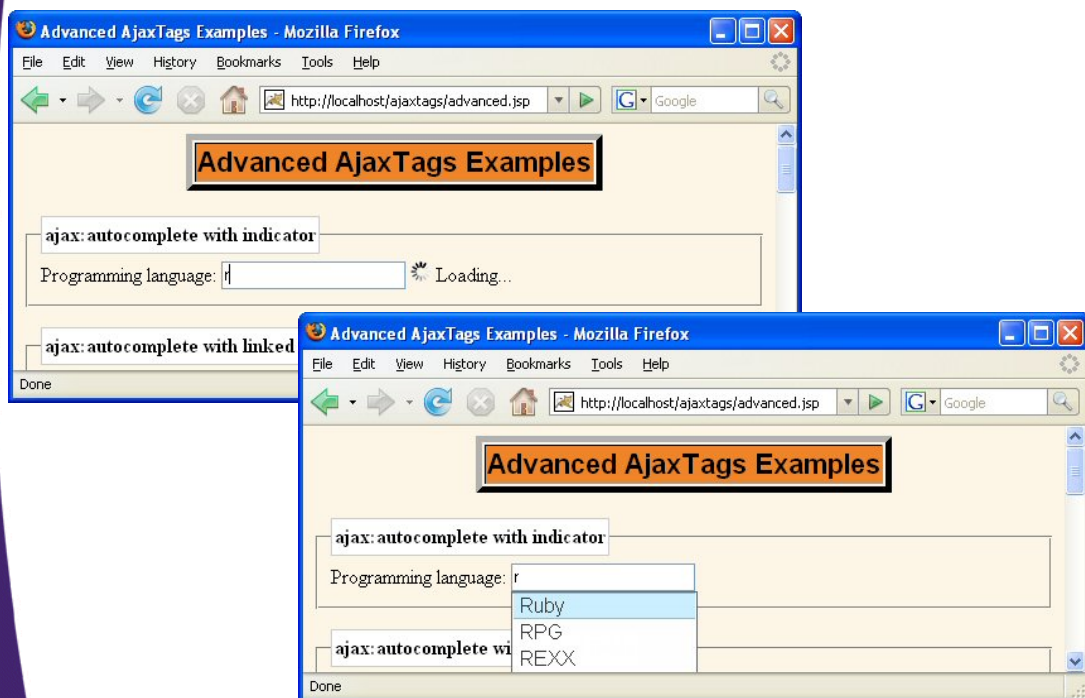
Server-Side Code

```
public class SlowLanguageCompleter
    extends LanguageCompleter {
  @Override
  public String getXmlContent(HttpServletRequest request,
                              HttpServletResponse response)
      throws Exception {
    try { Thread.sleep(5000); } catch(Exception e) {}
    return(super.getXmlContent(request, response));
  }
}
```

Results

ajax:autocomplete with Linked Fields

Basic Idea

- **Approach**
  - Server-side resource returns a list with names distinct from values
    - xmlBuilder.addItem("name1", "value1");
  - Names are displayed in dropdown box under main textfield
  - When a value is selected from dropdown, secondary value placed into a different textfield
- **Main uses**
  - When you have auxiliary information to display to user
    - E.g., show description when catalog ID selected
  - When you have associated values
    - E.g., word in another language

13

JSP Example

```
<fieldset>
  <legend>ajax:autocomplete with linked fields</legend>
  <form>
    <label for="englishAnimal">English animal:</label>
    <input type="text" id="englishAnimal"/>
    <label for="spanishAnimal">Spanish animal:</label>
    <input type="text" id="spanishAnimal"/>
  </form>
  <ajax:autocomplete
    source="englishAnimal"
    target="spanishAnimal"
    baseUrl="${contextPath}/animal-completer.ajax"
    parameters="englishAnimal={englishAnimal}"
    className="autocomplete"
    minimumCharacters="1"/>
</fieldset>
```

Server-Side Code

```
import javax.servlet.http.*;
import org.ajaxtags.helpers.*;
import org.ajaxtags.servlets.*;

public class AnimalCompleter extends BaseAjaxServlet {
  @Override
  public String getXmlContent(HttpServletRequest request,
                              HttpServletResponse response)
      throws Exception {
    String animalPrefix =
      request.getParameter("englishAnimal");
    String animalList = makeAnimalList(animalPrefix);
    return(animalList);
  }
```
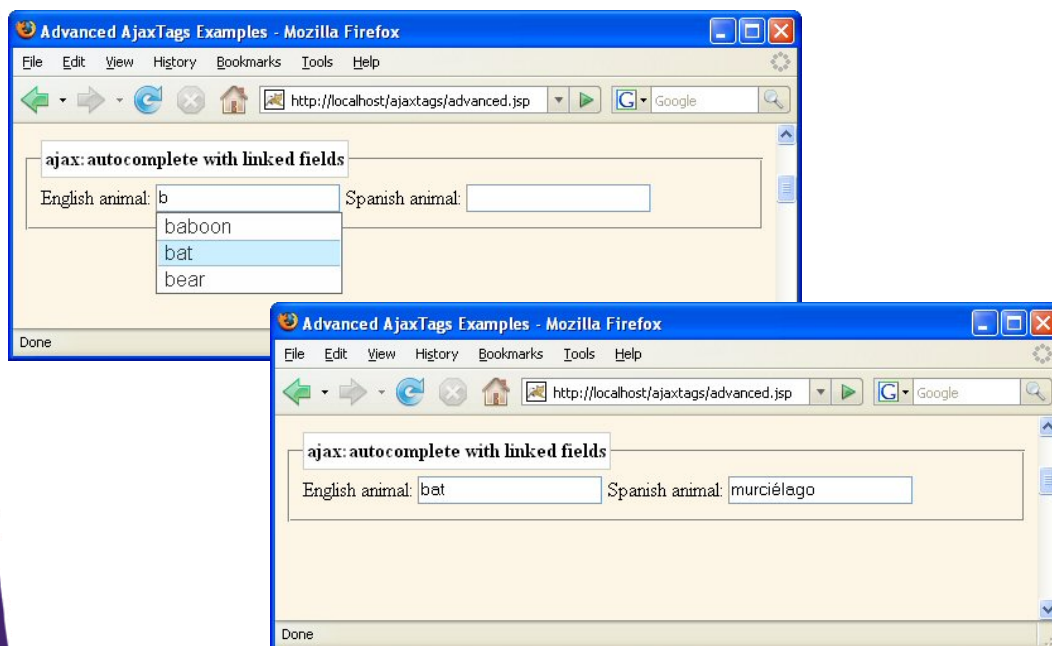
```
private static final String[][] animals =
{ {"aardvark", "aardvark"},{"alligator", "cocodrilo"},
  {"antelope", "antílopo"}, {"baboon", "babuino"},
  {"bat", "murciélago"}, {"bear", "oso"}, {"cat", "gato"},
  {"camel", "camello"}, {"conch", "concha"}
};
private String makeAnimalList(String animalPrefix) {
  animalPrefix = animalPrefix.toUpperCase();
  AjaxXmlBuilder builder = new AjaxXmlBuilder();
  for(String[] animalPair: animals) {
    String englishAnimal = animalPair[0];
    String spanishAnimal = animalPair[1];
    if(englishAnimal.toUpperCase().startsWith(animalPrefix)) {
      builder.addItem(englishAnimal, spanishAnimal);
    }
  }
  return(builder.toString());
}
}
```

Results

ajax:updateField with postFunction

Basic Idea

- **Approach**
  - You list a regular JavaScript function as the value of the postFunction or preFunction attribute
    - Almost all AjaxTags tags support postFunction and preFunction
  - The postFunction fires after the server-side resource is finished
  - The preFunction first just before the server-side resource is called
- **Main uses**
  - preFunction
    - To make an "I am working" message or image
    - Form field validation
  - postFunction
    - To highlight fields or show a success message

19

Script

- ## scripts/postFunction.js
  - Note that it is using scriptaculous calls

```
function showConversionSuccess() {
  // Temporarily highlight Celsius and Kelvin fields
  new Effect.Highlight('c');
  new Effect.Highlight('k');

  // Turn on successMessage div, then fade it out.
  Element.show('successMessage');
  setTimeout("Effect.DropOut('successMessage');", 2000);
}
```

- ## advanced.jsp

Load scriptaculous and Prototype. Defines contextPath variable.

```
<head>
...
<jsp:include page="/WEB-INF/includes/header.jsp"/>
<script src="${contextPath}/scripts/postFunction.js"
        type="text/javascript"></script>
<title>Advanced AjaxTags Examples</title>
</head>
```

JSP Example

```
<fieldset>
  <legend>ajax:updateField with postFunction</legend>
  <form>
    <label for="f">Enter temperature in Fahrenheit:</label>
    <input type="text" id="f"/>
    <input type="button" id="convertButton" value="Convert"/>
    <hr width="500" align="left"/>
    <label for="c">Temperature in Celsius:</label>
    <input type="text" id="c"/>
    <label for="k">Temperature in Kelvin:</label>
    <input type="text" id="k"/><p/>
    <div id="successMessage" style="display:none"
         class="temporaryMessage">
    Calculation complete</div>
  </form>
```

JSP Example (Continued)

```
<ajax:updateField
  source="f"
  target="c,k"
  baseUrl="${contextPath}/temperature-converter.ajax"
  action="convertButton"
  parameters="f={f}"
  parser="new ResponseXmlParser()"
  postFunction="showConversionSuccess"/>
</fieldset>
```
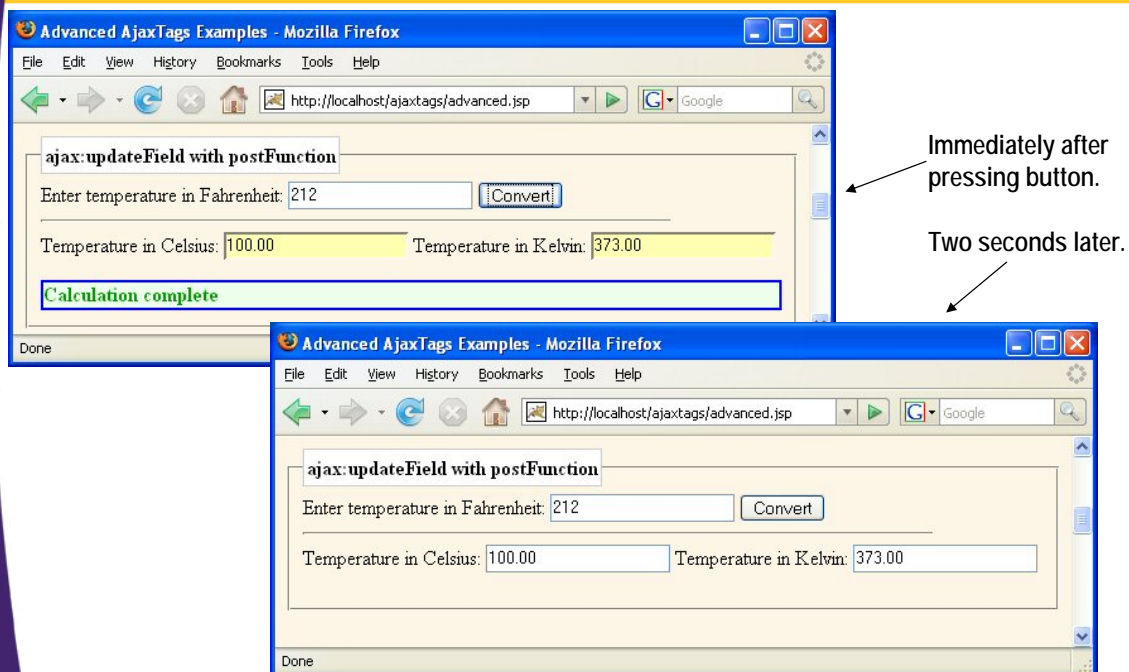
Server-Side Code

- **Same TemperatureConverter as in previous section**
  - Given a temperature in Fahrenheit, returns a list of corresponding temperatures in Celsius and Kelvin
  - Use of postFunction does not change how server-side resource works

Style Sheet Entries

```
.temporaryMessage {
  border:2px solid blue;
  background-color:#eeffee;
  color:#009900;
  font-weight: bold;
}
```

Results

Immediately after pressing button.

Two seconds later.

ajax:tabPanel with errorFunction

Basic Idea

- **Approach**
  - You list a regular JavaScript function as the value of the errorFunction attribute
    - Almost all AjaxTags tags support errorFunction
  - If server-side resource returns an HTTP status code other than 200, the errorFunction is triggered
- **Main uses**
  - Error messages
  - Default values when server-side resource is unavailable

Script

- ## scripts/errorFunction.js

```javascript
function warn() {
  alert("Server error!\n\n" +
        "Did you forget to specify\n" +
        "a default tab?");
}
```
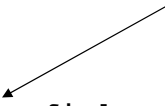
- ## error.jsp

```html
<head>
...
<jsp:include page="/WEB-INF/includes/header.jsp"/>
<script src="${contextPath}/scripts/errorFunction.js"
        type="text/javascript"></script>
<title>Advanced AjaxTags Examples</title>
</head>
```

JSP Example

```html
<fieldset>
  <legend>ajax:tabPanel</legend>
  <h2>Largest Cities in Selected Northeast States</h2>
  <div class="tabPanelWrapper">
    <ajax:tabPanel
      panelStyleId="panel"
      contentStyleId="content"
      panelStyleClass="tabPanel"
      contentStyleClass="tabContent"
      currentStyleClass="currentTab"
      errorFunction="warn">
      <ajax:tab
        caption="Maryland"
        baseUrl="${contextPath}/population-finder.ajax"
        parameters="state=Maryland,city=Baltimore"/>
      <ajax:tab
        caption="Virginia"
        baseUrl="${contextPath}/population-finder.ajax"
        parameters="state=Virginia,city=Virginia Beach"/>
        ...
    </ajax:tabPanel>
  </div></fieldset>
```

Should say defaultTab="true".
Without a default tab, tabPanel tries
to load contextPath/null.

Server-Side Code

- **Same PopulationFinder as in previous section**
  - Given a state and a city, returns the population
  - Use of errorFunction does not change how server-side resource works

Results

The page at http://localhost says:

⚠ Server error!

Did you forget to specify a default tab?

[OK]

ajax:htmlContent with Multiple Triggers

Basic Idea

- **Approach**
  – In previous section, we used source to designate button
    - Gives the ID of the element that will trigger the action
  – If you use sourceClass instead, then any element that has that CSS classname can be a trigger
    - The element must support onclick (push buttons, links, radio buttons, checkboxes, etc.)
- **Main use**
  – Multiple ways of triggering an action

JSP Example

```
<fieldset>
  <legend>ajax:htmlContent with Multiple Triggers</legend>
  <form>
    <label for="state2">State:</label>
    <select id="state2">
      <option value="">Select State</option>
      <option value="Maryland">Maryland</option>
      ...
      <option value="New York">New York</option>
    </select>
    <label for="city2">City:</label>
    <select id="city2" disabled="disabled">
      <option value="">Select City</option>
    </select>
    <br/><hr align="left" width="500"/>
    <input type="button" value="Show Population" class="trigger"/>
    <a href="javascript://no-op" class="trigger">Show Population</a>
    <input type="radio" class="trigger"/>Show Population
    <br/><hr align="left" width="500"/>
    <span id="population"></span>
  </form>
```

JSP Example (Continued)

```
<ajax:select
  baseUrl="${contextPath}/city-finder.ajax"
  source="state2"
  target="city2"
  parameters="state={state2}"/>
<ajax:htmlContent
  baseUrl="${contextPath}/population-finder.ajax"
  sourceClass="trigger"
  target="population"
  parameters="state={state2},city={city2}"/>
</fieldset>
```
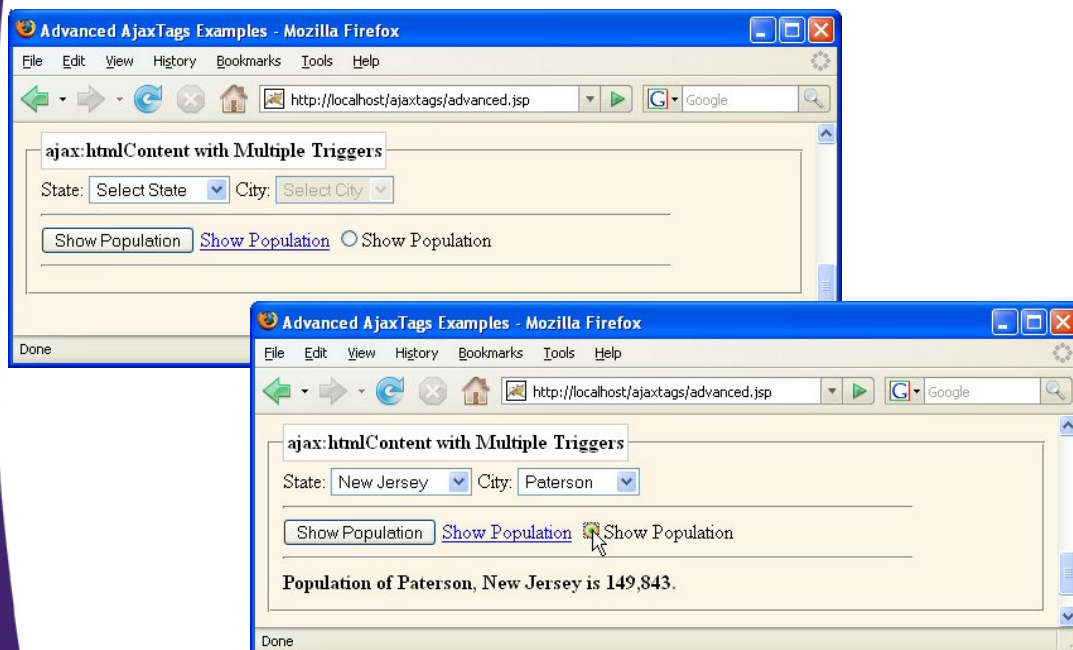
Server-Side Code

- **Same PopulationFinder as in previous examples**
  - Given a state and a city, returns the population
  - Type of trigger does not change how server-side resource works

Results

Summary

- **indicator attribute**
  - Specifies region that displays temporarily while server-side resource runs
- **Returning a list for ajax:autocomplete with names different from values**
  - Names get shown in the dropdown list
  - Name gets inserted in source textfield
    - I.e., the textfield that actually had dropdown list
  - Value gets inserted in target textfield
- **preFunction and postFunctions attributes**
  - Arbitrary JavaScript that runs before or after server-side resource
- **errorFunction attribute**
  - Function that runs when server has error
- **sourceClass attribute**
  - Designates elements that trigger server-side resources

Questions?