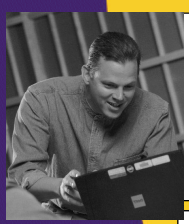
Ajax: The Basics Part II

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

For live Ajax & GWT training, 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 6, servlets/JSP (intermediate and advanced), Struts, JSF 1.*x*, JSF 2.0, Ajax, GWT 2.0 (with GXT), custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, Google Closure) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, Web Services, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- Sending GET data
- Reading textfield values
- Sending POST data
- Ajax toolkits and libraries
- Ajax books

Sending GET Data

Servlet Example: Design Deficiencies

- **No data sent from HTML page to servlet**
  - Solution: attach data to end of the URL (GET data)
    - Use normal GET format:
      - *mainaddress?var1=val1&var2=val2*
- **Notes**
  - In next examples we will get data from input fields, but for now we will embed it directly in the URL
  - If you have an "&" in a URL in xhtml, you are required to replace it with "&"
  - To read the data on the server, servlet does request.getParameter as usual

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Initiate a GET or POST request to a servlet
      - URL has GET data attached to the end
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Use innerHTML to insert result into designated element
- **HTML**
  - Load JavaScript from centralized directory
  - Designate control that initiates request
  - Give ids to input elements
  - Define a blank placeholder element with a known id

Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

No changes from version in previous section.
The getRequestObject function stays the same
in all sections.

Initiate Request

```
function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

No changes from version of ajaxResult in previous section.
The difference is the address, but the address gets passed
in from the HTML page.

Handle Response

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}

function htmlInsert(id, htmlData) {
  document.getElementById(id).innerHTML = htmlData;
}
```

Changed slightly from previous section: now uses htmlInsert helper function.

HTML Code

```
...
<fieldset>
  <legend>
    Sending Fixed GET Data, Result Shown in HTML
  </legend>
  <input type="button" value="Show Chicago Time"
         onclick='ajaxResult
                    ("show-time-in-city?city=Chicago",
                     "chicago-time")'/>
  <div id="chicago-time"></div>
</fieldset>
...
```

Servlet Code

- **Servlet reads the "city" parameter**
  - Uses this to determine time zone
  - Unknown or missing city results in error message
- **Servlet generates HTML markup**
  - Rather than returning an ordinary string to be inserted into the div, returns text containing xhtml tags
    - Provides more control over the final result
    - Remember to be check that the final result is legal in xhtml
      - Tags should be lowercase
      - Be aware of nesting rules (e.g., block-level elements allowed inside div and td, but only inline elements allowed inside p or span)

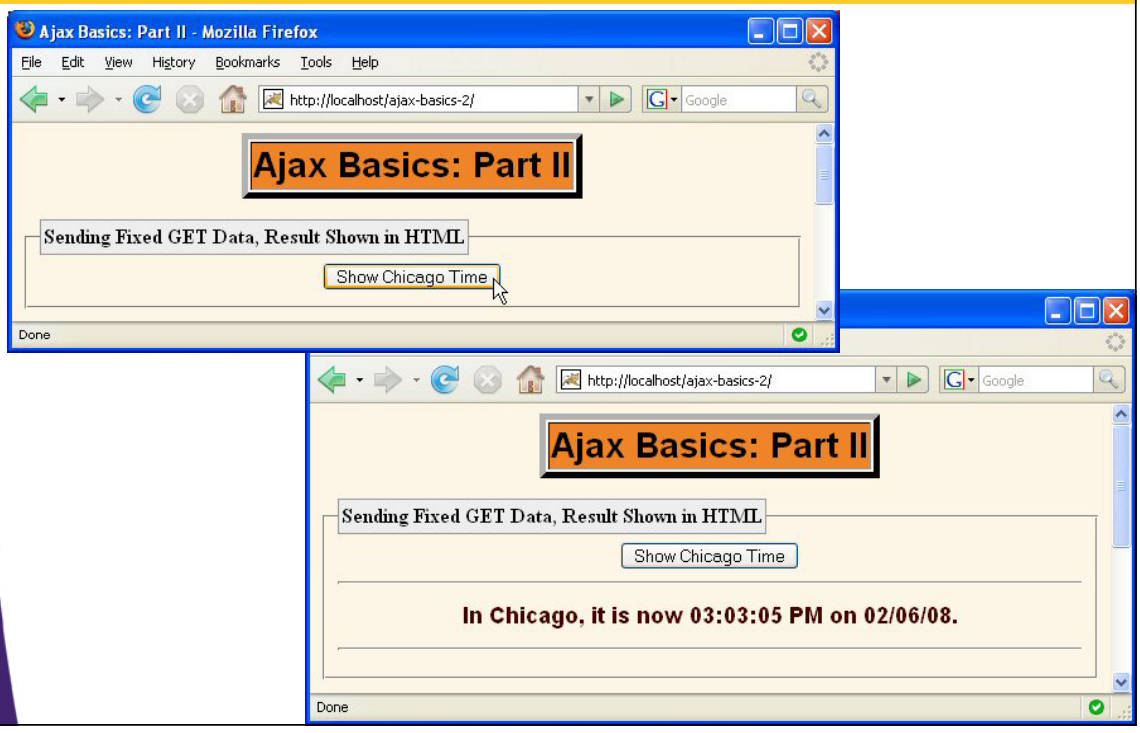
Servlet Code

```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    String cityName = request.getParameter("city");
    String message = CityUtils.getTime(cityName);
    if (message.startsWith("In")) { // Found city
      message =
        String.format("<hr/><h2>%s</h2><hr/>", message);
    } else { // No city or unknown city
      message =
        String.format("<h2 class='error'>%s</h2>", message);
    }
    out.print(message);
  }
}
```

CityUtils Class

- **Maintains a table of cities and associated time zones (populations also – used later)**
  - Given the name of a city, it finds the difference in hours between that city's time and server time (east coast USA)
- **Computes server time**
  - Using standard GregorianCalendar class
- **Converts to time in that city**
  - By calling the "add" method with the time zone offset
- **Formats the time and day**
  - Using String.format with %tr and %tD
  - Reminder: full source code is online
    - http://courses.coreservlets.com/Course-Materials/ajax.html

Sending GET Data: Results



Reading User Input (and Embedding it in GET Data)

GET Example: Design Deficiencies

- **City name was always Chicago**
  - Solution: send user data
    - Give an id to textfield
      - <input type="text" id="someID"/>
      - The id is not the same as the textfield name. The name is used only in regular form submissions, which does not apply to Ajax.
    - Read data from textfield
      - document.getElementById(someID).value
- **Textfields can have spaces and other characters that are illegal in URLs**
  - Solution: filter (escape) textfield data
    - Call "escape" to convert spaces and other special chars
    - Put result into the GET data
    - Request URL with embedded GET data

Summary of New Features

- **Read a textfield value**
  - userInput = document.getElementById(someID).value;
    - The value property works for textfields and most other input elements. But for radio buttons and checkboxes, the value is either "on" or empty. We will show examples of this later.
- **Escape spaces and other special characters**
  - userInput = escape(userInput);
- **Put data into the URL**
  - url = baseURL + "?someName=" + userInput;
    - For multiple input elements, put "&" between name/value pairs.
- **Send out request and process result**
  - No changes from previous examples
- **Helper function**

```
function getValue(id) {
  return(escape(document.getElementById(id).value));
}
```


Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Initiate a GET or POST request to a servlet
      - URL has GET data attached to the end
      - Data based on document.getElementById(id).value of some textfield
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Use innerHTML to insert result into designated element
- **HTML**
  - Load JavaScript from centralized directory
  - Designate control that initiates request
  - Give ids to input elements
  - Defines a blank placeholder element with a known id

Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

No changes from previous examples

Initiate Request

```
function showTimeInCity(inputField, resultRegion) {
  var baseAddress = "show-time-in-city";
  var data = "city=" + getValue(inputField);
  var address = baseAddress + "?" + data;
  ajaxResult(address, resultRegion);
}

function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}

function getValue(id) {
  return(escape(document.getElementById(id).value));
}
```

.The ajaxResult function is unchanged from previous example.

Handle Response

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}
```

No changes from previous example

HTML Code

```
...
<fieldset>
  <legend>
    Sending Dynamic GET Data, Result Shown in HTML
  </legend>
  <label>City: <input type="text" id="city-1"/>
  </label><br/>
  <input type="button" value="Show City Time"
         onclick='showTimeInCity("city-1",
                                 "city-1-time")'/>
  <div id="city-1-time"></div>
</fieldset>
...
```

Style Sheet Code

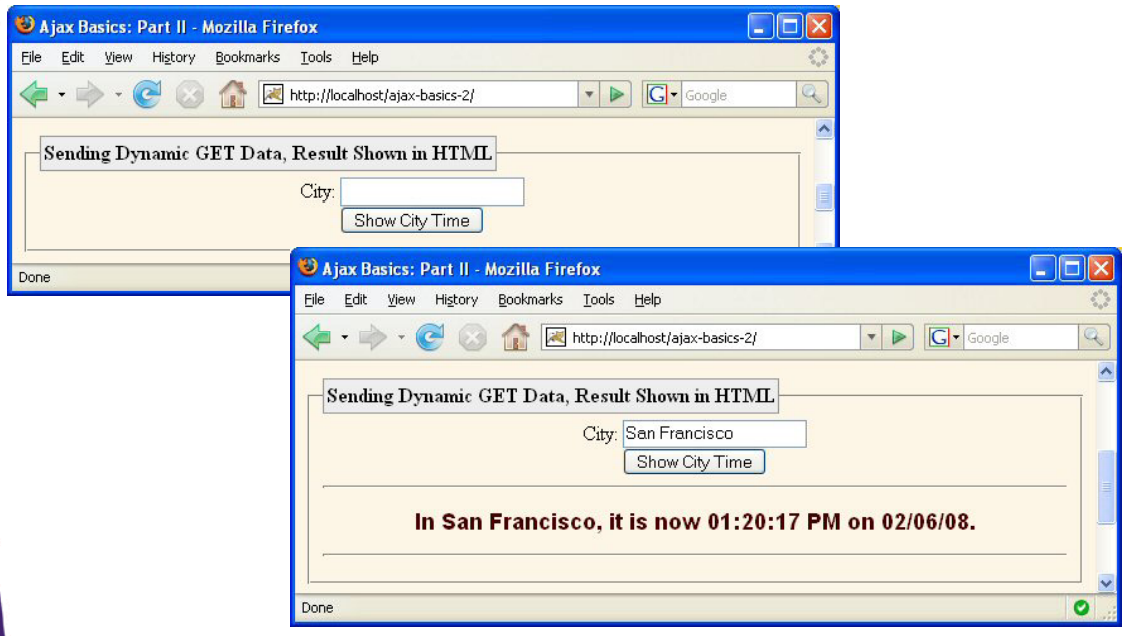
```
h2 { color: #440000;
     font-weight: bold;
     font-size: 18px;
     font-family: Arial, Helvetica, sans-serif;
}
.error { background-color: yellow;
         color: red;
         font-weight: bold;
         font-size: 20px;
         font-family: Arial, Helvetica, sans-serif;
         border-style: inset;
}
```


Servlet Code

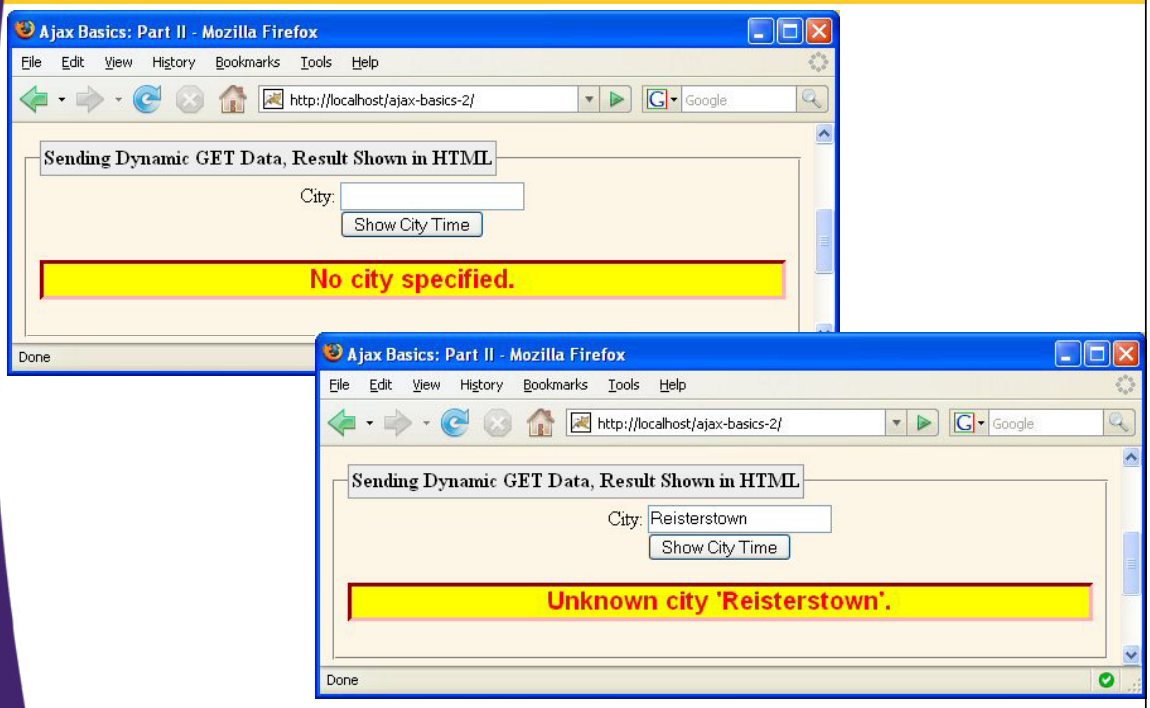
```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    String cityName = request.getParameter("city");
    String message = CityUtils.getTime(cityName);
    if (message.startsWith("In")) { // Found city
      message =
        String.format("<hr/><h2>%s</h2><hr/>", message);
    } else { // No city or unknown city
      message =
        String.format("<h2 class='error'>%s</h2>", message);
    }
    out.print(message);
  }
}
```

No changes from previous example

Reading User Input: Results (Good Data)



Reading User Input: Results (Bad Data)



Sending POST Data

GET vs. POST

- **In normal Web pages, there are compelling reasons for choosing POST or GET**
  - POST
    - URL is simpler
    - Data is hidden from people looking over your shoulder
    - Larger amounts of data can be sent
    - Can send special characters (e.g., in uploaded files)
    - Should always be used if the requests changes data on server
  - GET
    - Can bookmark results page
- **With Ajax, end users don't see URL, so choice is relatively arbitrary**
  - Unless there is a very large amount of data or the request changes the server state, in which case POST is preferred
    - Some JS libraries use GET by default, others use POST

Sending POST Data in JavaScript

- **Collect data from form**
  - Give ids to input elements
    <input type="text" id="*some-id*"/>
  - Read data
    var value1 = document.getElementById("*some-id*").value;
  - URL-encode data and form into query string
    var data = "var1=" + escape(value1);
- **Specify POST instead of GET for "open"**
  request.open("POST", address, true);
- **Specify form encoding type**
  request.setRequestHeader("Content-Type",
  "application/x-www-form-urlencoded");
- **Supply data in "send"**
  request.send(data);

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Initiate a POST request to a servlet
      - Put POST data in the send method
      - Data based on document.getElementById(id).value of some textfield
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Use innerHTML to insert result into designated element
- **HTML**
  - Load JavaScript from centralized directory
  - Designate control that initiates request
  - Give ids to input elements
  - Define a blank placeholder element with a known id

Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

No changes from previous examples

Initiate Request

```
function showTimeInCityPost(inputField, resultRegion) {
  var address = "show-time-in-city";
  var data = "city=" + getValue(inputField);
  ajaxResultPost(address, data, resultRegion);
}

function ajaxResultPost(address, data, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("POST", address, true);
  request.setRequestHeader
                ("Content-Type",
                 "application/x-www-form-urlencoded");
  request.send(data);
}
```

Handle Response

```
function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}
```

No changes from previous example

HTML Code

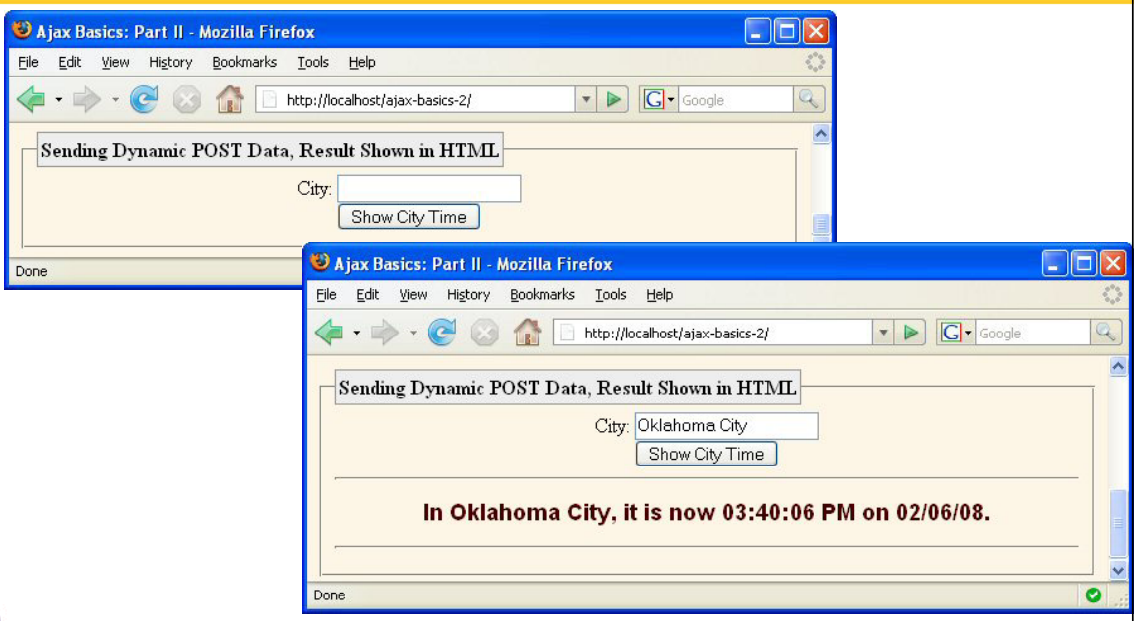
```
...
<fieldset>
  <legend>
    Sending Dynamic POST Data, Result Shown in HTML
  </legend>
  <label>City: <input type="text" id="city-2"/>
  </label><br/>
  <input type="button" value="Show City Time"
         onclick='showTimeInCityPost("city-2",
                                     "city-2-time")'/>
  <div id="city-2-time"></div>
</fieldset>
...
```

Servlet Code

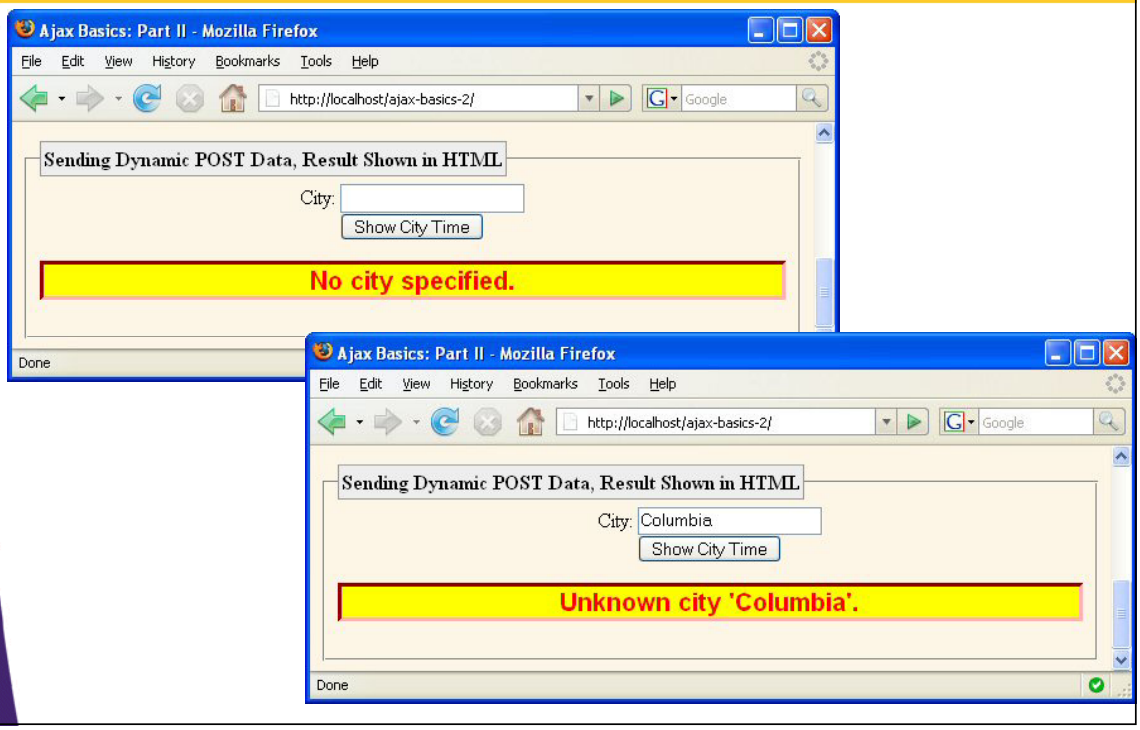
```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    // Shown in previous examples
  }

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

No changes from previous example

Reading User Input: Results (Good Data)



Reading User Input: Results (Bad Data)



Ajax Toolkits and Libraries

Client-Side Tools (JavaScript Libraries with Ajax Support)

- **jQuery**
  - http://jquery.com/
- **Prototype**
  - http://www.prototypejs.org/
- **script.aculo.us**
  - http://script.aculo.us/
- **Dojo**
  - http://www.dojotoolkit.org/
- **Ext-JS**
  - http://extjs.com/
- **Yahoo User Interface Library (YUI)**
  - http://developer.yahoo.com/yui/
- **Google Closure Library**
  - http://code.google.com/closure/library/

Preview of Coming Attractions: Ajax with and without Toolkits

- **With basic JavaScript**

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else { return(null); }
}
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { someFunct(request); };
  request.open("GET", "some-url", true);
  request.send(null);
}
```

Preview of Coming Attractions: Ajax with and without Toolkits

- **Prototype (handler passed response object)**

```
new Ajax.Request("address",
                 {onSuccess: handlerFunct});
```

- **jQuery (handler passed response text)**

```
$.ajax({url: "address",
        success: handlerFunct});
```

- **Ext (handler passed response object)**

```
Ext.Ajax.request({url: "address",
                  success: handlerFunct});
```

- **Dojo (handler passed response text)**

```
dojo.xhrGet({url: "address",
             load: handlerFunct});
```


Server-Side Tools

- **Direct Web Remoting**
  - Lets you call Java methods semi-directly from JavaScript
  - http://getahead.ltd.uk/dwr/
- **JSON/JSON-RPC**
  - For sending data to/from JavaScript with less parsing
  - http://www.json.org/
  - http://json-rpc.org/

Hybrid Client/Server Tools

- **JSP custom tag libraries**
  - Create tags that generate HTML and JavaScript
  - http://courses.coreservlets.com/Course-Materials/msajsp.html
- **AjaxTags (built on top of script.aculo.us)**
  - JSP custom tags that generate Ajax functionality
    - Supports many powerful Ajax capabilities with very simple syntax
    - http://ajaxtags.sourceforge.net
- **Google Web Toolkit**
  - Write code in Java, translate it to JavaScript
    - http://code.google.com/webtoolkit/
  - Also see https://ajax4jsf.dev.java.net/
    - GWT/JSF Integration Toolkit

JSF-Based Tools

- **Trinidad (formerly Oracle ADF)**
  - http://myfaces.apache.org/trinidad/
- **Tomahawk**
  - http://myfaces.apache.org/tomahawk/
- **Ajax4JSF (and RichFaces)**
  - http://labs.jboss.com/jbossajax4jsf/
- **IceFaces**
  - http://www.icefaces.org/
- **Backbase Ajax/JSF Library**
  - http://www.backbase.com/products/enterprise-ajax-for-jsf/overview/
  - Main version is not free, but scaled-down community version is free
- **Infragistics NetAdvantage**
  - http://www.infragistics.com/java/netadvantage/ (not free)
- **JSF 2.0 has integrated Ajax support**
  - http://www.coreservlets.com/JSF-Tutorial/

Books (In Rough Order of Preference)

- **Ajax in Practice**
  - Crane et al. Manning.
- **JavaScript: The Definitive Guide**
  - Flanagan. O'Reilly.
- **Foundations of Ajax**
  - Asleson and Schutta. APress.
- **Ajax in Action**
  - Crane, Pascarello, and James. Manning.
- **GWT in Action**
  - Hanson and Tacy. Manning.
- **Pro JSF and Ajax**
  - Jacobi and Fallows. APress.
- **Prototype and Scriptaculous in Action**
  - Crane et al. Manning.
- **Pro Ajax and Java Frameworks**
  - Schutta and Asleson. APress.
- **Professional Ajax, Second Edition**
  - Zakas, et al. Wrox.

Wrap-up

Summary

- **JavaScript**
  - Define request object
    - Check for both Microsoft and non-MS objects. Identical in all apps.
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Read user data with getElementById(id).value, then escape it
    - Initiate a GET or POST request
      - If POST, set Content-Type and put data in send method
      - If GET, append data on end of address after "?"
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText
    - Do something with result
      - Use innerHTML to insert result into designated element
- **HTML**
  - Give ids to input elements and to result region. Initiate process.
- **Java**
  - Use JSP, servlet, or combination (MVC) as appropriate.
  - Prevent browser caching.

Questions?