Ajax: Handling Different Server Data Formats

XML, JSON, and String

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

Topics in This Section

- **Building HTML tables in JavaScript**
- **XML data**
  - Parsing results
  - Building XML data on server with MVC
- **JSON data**
  - Parsing results
  - Building JSON data on server with MVC
- **String data**
  - Parsing results
  - Building String data on server with MVC
- **Combination data**
  - Deciding what data format to use at run time

Building HTML Tables

Motivation

- **In many cases, the server data is intimately tied to a specific HTML form on the client**
  - In that case, it makes good sense for the server to send HTML tags and for the client to merely insert them
- **In other cases, the same server data may be used in several forms or in different pages**
  - And the data may be used in different ways by different applications
  - In that case, it makes sense for the server to send some standard data format
    - The client must parse (extract info from) this data format
    - The client must build HTML based upon the data

Utility: Building HTML Tables

```
function getTable(headings, columns) {
  var table = "<table border='1'>\n" +
              getTableHeadings(headings) +
              getTableBody(columns) +
              "</table>";
  return(table);
}
```

- **Note**
  - The first argument contains the headings
    - To be inserted into th elements
  - The second argument is an array-of-arrays, where each sub-array is a table *column*
    - The elements in the sub-arrays will be inserted into td elements

Utility: Building HTML Tables (Continued)

```
function getTableHeadings(headings) {
  var firstRow = "  <tr>";
  for(var i=0; i<headings.length; i++) {
    firstRow += "<th>" + headings[i] + "</th>";
  }
  firstRow += "</tr>\n";
  return(firstRow);
}

function getTableBody(columns) {
  var numRows = columns[0].length;
  var numCols = columns.length;
  var body = "";
  for(var row=0; row<numRows; row++) {
    body += "  <tr>";
    for(var col=0; col<numCols; col++) {
      body += "<td>" + columns[col][row] + "</td>";
    }
    body += "</tr>\n";
  }
  return(body);
}
```

Other Utilities (From Last Section)

```
// Insert the html data into the element
// that has the specified id.

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

// Return escaped value of textfield that has given id.
// The builtin "escape" function converts < to &lt;, etc.

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

Example Usage (JavaScript)

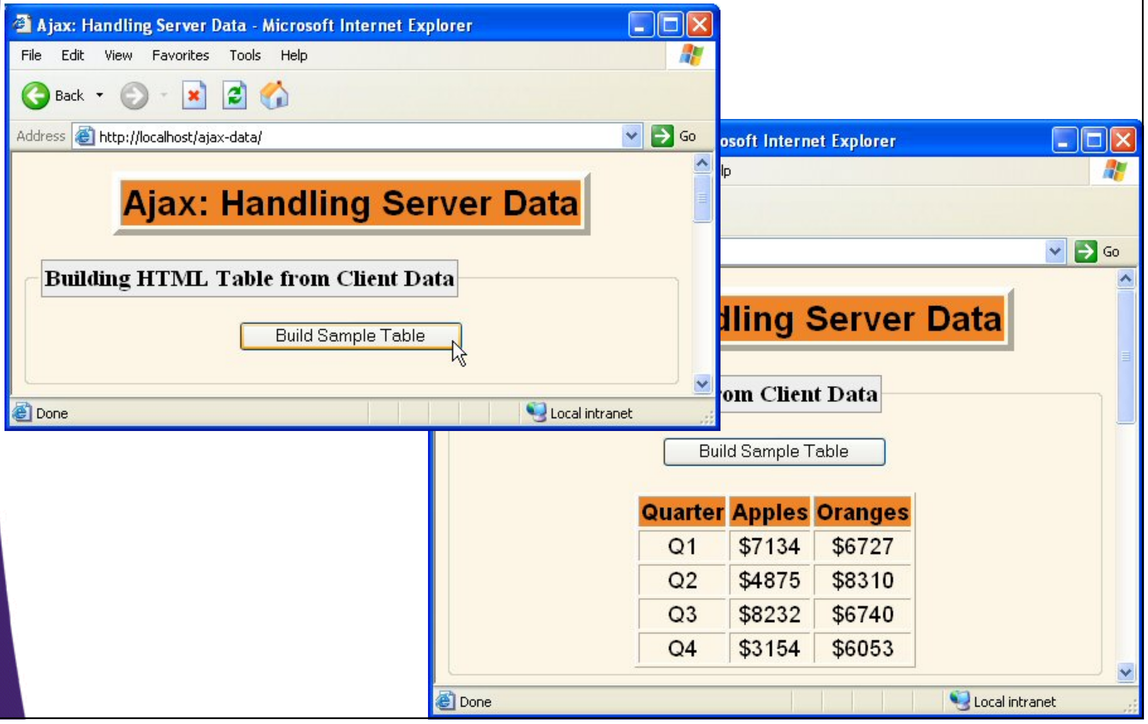
```
function clientTable(displayRegion) {
  var headings = ["Quarter", "Apples", "Oranges"];
  var columns = [["Q1", "Q2", "Q3", "Q4"],
                 [randomSales(), randomSales(),
                  randomSales(), randomSales()],
                 [randomSales(), randomSales(),
                  randomSales(), randomSales()]];
  var table = getTable(headings, columns);
  htmlInsert(displayRegion, table);
}

function randomSales() {
  var sales = 1000 + (Math.round(Math.random() * 9000));
  return("$" + sales);
}
```

Example Usage (HTML)

```
...
<fieldset>
  <legend>Building HTML Table from Client Data</legend>
  <form action="#">
   <input type="button" value="Build Sample Table"
          onclick='clientTable("client-table")'/>
  </form>
  <p/>
  <div id="client-table"></div>
</fieldset>
...
```

Example Usage (Result)


Ajax: Handling Server Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost/ajax-data/
Go
Done
Local intranet

Ajax: Handling Server Data

Building HTML Table from Client Data

Build Sample Table

Quarter	Apples	Oranges
Q1	$7134	$6727
Q2	$4875	$8310
Q3	$8232	$6740
Q4	$3154	$6053

Handling XML Data

Basic Tasks

- **Getting the raw XML data**
  - var xmlDocument = request.responseXML;
- **Finding array of XML elements**
  - xmlDocument.getElementsByTagName
    (xmlElementName);
- **Finding the text between start and end tags**
  - someElement.childNodes[0].nodeValue
- **Note**
  - In a later section we will give much more detail on XML manipulation in JavaScript

Basic Tasks: Details

- **Getting the raw XML data**
  - var xmlDocument = request.responseXML;
    - Instead of request.responseText
- **Finding array of XML elements**
  - var elementArray =
    xmlDocument.getElementsByTagName
    (xmlElementName);
  - For example, if XML is

```
<a><b>foo</b>
    <c>bar</c>
    <b>baz</b>
</a>
```

    Then getElementsByTagName("b") returns a two-element array containing objects representing the two b tags and their contents

Basic Tasks: Details (Continued)

- **Finding the text between start and end tags**
  - someElement.childNodes[0].nodeValue
    - Elements containing text are treated as having a hidden sub-element called a text node, and it is that sub-element's value that is the text you want
- **Example**
  - If XML is
    ```
    <a><b>foo</b>
        <c>bar</c>
        <b>baz</b>
    </a>
    ```
  - Then the following makes value be "foo"
    - var elementArray = xmlDocument.getElementsByTagName("b");
    - var value = elementArray[0].childNodes[0].nodeValue;

XML Utility Function

```
// Given the name of an XML element, returns an array
// of the values of all elements with that name.
// E.g., for
//    <foo><a>one</a><q>two</q><a>three</a></foo>
// getXmlValues(doc, "a") would return
//    ["one", "three"].

function getXmlValues(xmlDocument, xmlElementName) {
  var elementArray =
    xmlDocument.getElementsByTagName(xmlElementName);
  var valueArray = new Array();
  for(var i=0; i<elementArray.length; i++) {
    valueArray[i] =
      elementArray[i].childNodes[0].nodeValue;
  }
  return(valueArray);
}
```

General Utility Function (Update from Previous Section)

```
// Generalized version of ajaxResultPost. In this
// version, you pass in a response handler function
// instead of just a result region.

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

General Utility Function (Same as in Previous Sections)

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

Handling XML Data: Example

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
      - Get text from XML with getElementsByTagName and childNodes[0].nodeValue
      - Build HTML table or other HTML data out of the text
    - 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

Initiate Request

```
function xmlCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=xml";
  ajaxPost(address, data,
           function(request) {
             showXmlCityInfo(request, resultRegion);
           });
}
```

Handle Response

```
function showXmlCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var xmlDocument = request.responseXML;
    var headings = ["City", "Time", "Population"];
    var columns = [getXmlValues(xmlDocument, "name"),
                   getXmlValues(xmlDocument, "time"),
                   getXmlValues(xmlDocument, "population")];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML Code

```
...
<fieldset>
  <legend>Getting XML Data from Server, Building HTML Table</legend>
  <form action="#">
   <label for="city-type-1">City Type:</label>
   <select id="city-type-1">
     <option value="top-5-cities">Largest Five US Cities</option>
     <option value="second-5-cities">Second Five US Cities</option>
     <option value="cities-starting-with-s">
       US Cities Starting with 'S'</option>
     <option value="superbowl-hosts">
       Most Recent Superbowl Hosts</option>
    </select>
   <br/>
   <input type="button" value="Show Cities"
          onclick='xmlCityTable("city-type-1", "xml-city-table")'/>
  </form>
  <p/>
  <div id="xml-city-table"></div>
</fieldset>
...
```

Server Design: MVC

- **Logic**
  - Set the headers, read the request parameters, compute the results
  - Do this in Java (called by a servlet)
- **Presentation**
  - Build an XML file
  - Do this in JSP
    - Use the JSP expression language to access the results
- **Minor Variation**
  - So that you can set headers and Content-Type, use RequestDispatcher.include instead of RequestDispatcher.forward
- **Reminder**
  - Details on MVC and on the JSP expression language are given in other sections.
    - From the servlet and JSP tutorials

Servlet Code

```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    String cityType = request.getParameter("cityType");
    List<City> cities = findCities(cityType);
    request.setAttribute("cities", cities);
    String format = request.getParameter("format");
    String outputPage;
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } ...
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

Servlet Code (Continued)

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

- **I will use POST from the JavaScript**
  - But having GET support makes it easier to test interactively
  - So support both

Servlet Code (Continued)

```
private Map<String,String[]> cityTypeMap;

public void init() {
  cityTypeMap = new HashMap<String,String[]>();
  cityTypeMap.put("top-5-cities",
                  CityUtils.TOP_5_CITIES);
  cityTypeMap.put("second-5-cities",
                  CityUtils.SECOND_5_CITIES);
  cityTypeMap.put("cities-starting-with-s",
                  CityUtils.CITIES_STARTING_WITH_S);
  cityTypeMap.put("superbowl-hosts",
                  CityUtils.SUPERBOWL_HOSTS);
}
```

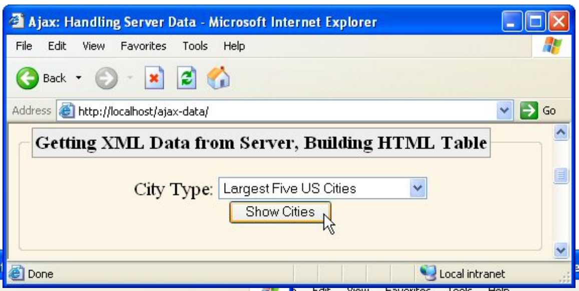
Supporting Code

```
private List<City> findCities(String cityType) {
  String[] cityNames = cityTypeMap.get(cityType);
  if (cityNames == null) {
    cityNames = CityUtils.TOP_5_CITIES;
  }
  List<City> cities = new ArrayList<City>();
  for(String cityName: cityNames) {
    cities.add(CityUtils.getCity(cityName));
  }
  return(cities);
}
```

JSP Code (/WEB-INF/results/cities-xml.jsp)

```
<?xml version="1.0" encoding="UTF-8"?>
<cities>
  <city>
    <name>${cities[0].name}</name>
    <time>${cities[0].shortTime}</time>
    <population>${cities[0].population}</population>
  </city>
  <city>
    <name>${cities[1].name}</name>
    <time>${cities[1].shortTime}</time>
    <population>${cities[1].population}</population>
  </city>
  ...
  <city>
    <name>${cities[4].name}</name>
    <time>${cities[4].shortTime}</time>
    <population>${cities[4].population}</population>
  </city>
</cities>
```

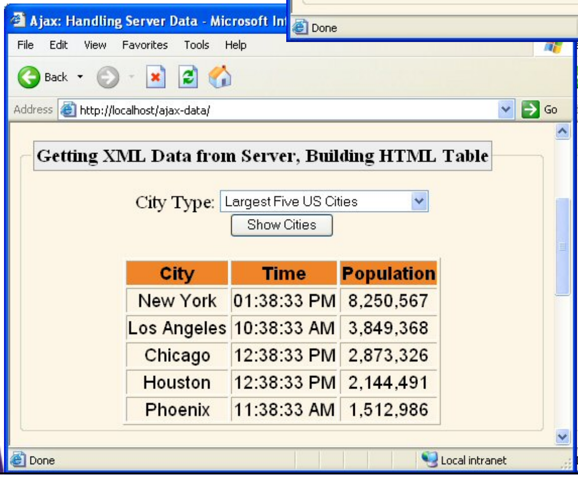
XML Data: Results


Ajax: Handling Server Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost/ajax-data/
Go
Done
Local intranet

Getting XML Data from Server, Building HTML Table

City Type: Largest Five US Cities

Show Cities

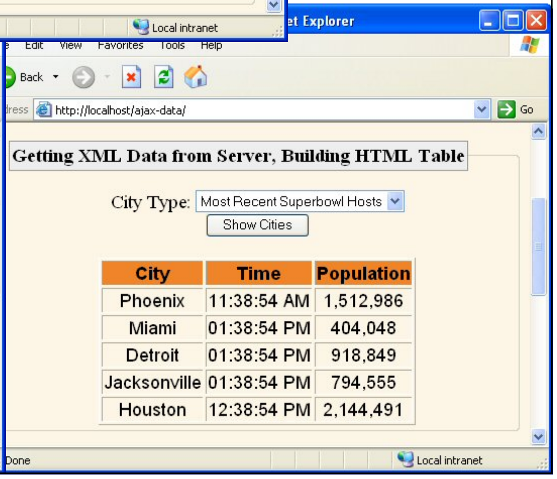

Ajax: Handling Server Data - Microsoft Int
File Edit View Favorites Tools Help
Back
Address http://localhost/ajax-data/
Go
Done
Local intranet

Getting XML Data from Server, Building HTML Table

City Type: Largest Five US Cities

Show Cities

City	Time	Population
New York	01:38:33 PM	8,250,567
Los Angeles	10:38:33 AM	3,849,368
Chicago	12:38:33 PM	2,873,326
Houston	12:38:33 PM	2,144,491
Phoenix	11:38:33 AM	1,512,986



Getting XML Data from Server, Building HTML Table

City Type: Most Recent Superbowl Hosts

Show Cities

City	Time	Population
Phoenix	11:38:54 AM	1,512,986
Miami	01:38:54 PM	404,048
Detroit	01:38:54 PM	918,849
Jacksonville	01:38:54 PM	794,555
Houston	12:38:54 PM	2,144,491

Handling JSON Data

Basic Tasks

- **JSON**
  - JavaScript Object Notation. A simple textual representation of JavaScript objects that is already directly supported in JavaScript.
  - More details will be provided in later section
- **Directly in JavaScript**
  - var someObject =
    { "field1": "value1",
      "field2": "value2", ... };
- **In a string (e.g., when coming in on network)**
  - Surround object representation in parens
  - Pass to the builtin "eval" function

Basic Tasks: Details

- **Main object**
  - Surround entire value in curly braces
  - Put field names in single or double quotes
  - Use colons between field names and values
  - Put commas after each fieldname: fieldvalue pair.
- **Field values**
  - Strings: use single or double quotes
  - Numbers: no quotes needed
  - Arrays: use comma-separated values inside *square* braces
- **Putting JSON in strings**
  - Enclose in parens and quotes
    - Use single quotes on the outside if you have double quotes inside
  - Pass result to "eval" to get an object back

Basic Tasks: Example

```
var firstObject =
  { "field1": "string-value1",
    "field2": 3,
    "field3": ["a", "b", "c"]
   };
var someString =
  '({ "f1": "val1", "f2": "val2" })';
var secondObject = eval(someString);
```

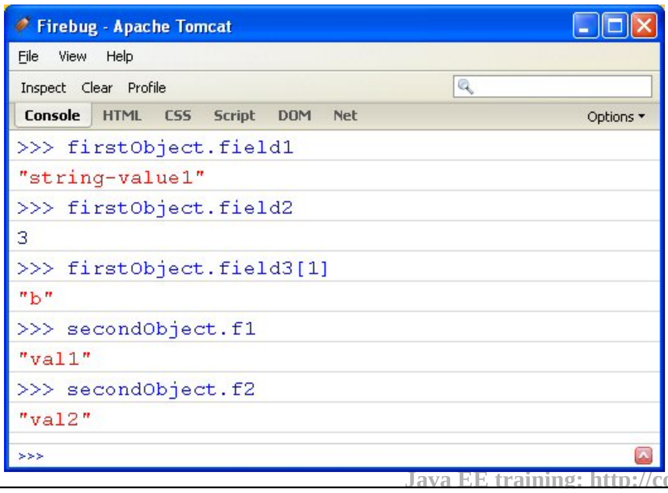
- **Results**
  - firstObject.field1 → "string-value1"
  - firstObject.field2 → 3
  - firstObject.field3[1] → "b"
  - secondObject.f1 → "val1"
  - secondObject.f2 → "val2"

Testing

- **Don't use HTML: use Firebug**
  – Open Firebug
    - F12 or Tools → Firebug → Open Firebug
  – Go to the Console
  – Cut/paste the expressions into the command line
    - Either at the bottom or the right, depending on Options
- **Reminder**
  – Firebug is indispensible for Ajax development and testing
  – Download from http://getfirebug.com/
  – Disable by default for most sites
    - Right-click on the Firebug logo at bottom right
    - Select "Disable Firebug"
    - Right-click on logo again
    - Select "Allowed Sites"
    - Enter localhost and other sites you test your code on
  – You can still temporarily enable Firebug to trace how some random site works. Try it on gmail or Yahoo mail to see what is happening.

Testing in Firebug: Example

- **Steps**
  – Opened Firebug with F12
  – Cut/pasted code from earlier slide
  – Interactively entered the expressions shown in blue



```
>>> firstObject.field1
"string-value1"
>>> firstObject.field2
3
>>> firstObject.field3[1]
"b"
>>> secondObject.f1
"val1"
>>> secondObject.f2
"val2"
>>>
```

Handling JSON Data: Example

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
      - Pass string to "eval" to get a real JavaScript object
      - Access fields, array elements, etc., with normal JavaScript syntax
    - 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

Initiate Request

```
function jsonCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=json";
  ajaxPost(address, data,
           function(request) {
             showJsonCityInfo(request, resultRegion);
           });
}
```

Handle Response

```
function showJsonCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var rawData = request.responseText;
    var data = eval(rawData);
    var headings = ["City", "Time", "Population"];
    var columns = [data.names,
                   data.times,
                   data.populations];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML Code

```
...
<fieldset>
  <legend>Getting JSON Data from Server, Building HTML Table
  </legend>
  <form action="#">
   <label for="city-type-2">City Type:</label>
   <select id="city-type-2">
     <option value="top-5-cities">Largest Five US Cities</option>
     <option value="second-5-cities">Second Five US Cities</option>
     <option value="cities-starting-with-s">
       US Cities Starting with 'S'</option>
     <option value="superbowl-hosts">
       Most Recent Superbowl Hosts</option>
    </select>
   <br/>
   <input type="button" value="Show Cities"
          onclick='jsonCityTable("city-type-2",
                                 "json-city-table")'/>
  </form>
  <p/>
  <div id="json-city-table"></div>
</fieldset>...
```

Server Design: MVC

- **Logic**
  - No changes to basic logic
  - Only addition is logic to decide which results page applies
- **Presentation**
  - Build a plain-text page instead of an XML page
  - Embed data in JSON format
    - Put parens around data because it will be passed to eval at the other end

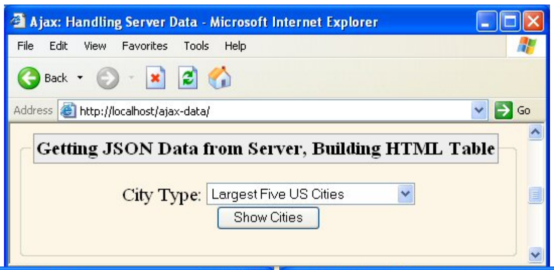
Servlet Code

```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    ...
    request.setAttribute("cities", cities);
    String format = request.getParameter("format");
    String outputPage;
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } else if ("json".equals(format)) {
      response.setContentType("text/javascript");
      outputPage = "/WEB-INF/results/cities-json.jsp";
    } ...
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

JSP Code (/WEB-INF/results/cities-json.jsp)

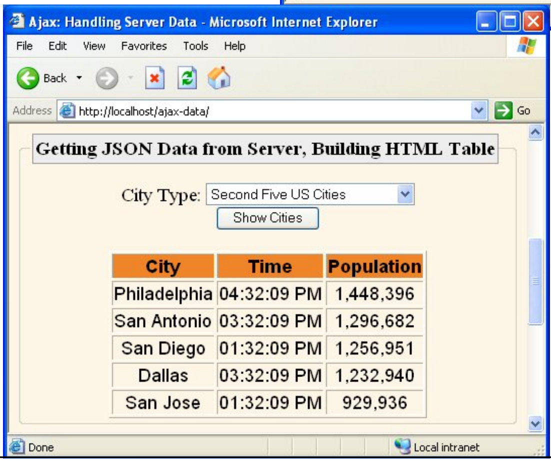
```
({ "names": ["${cities[0].name}",
             "${cities[1].name}",
             "${cities[2].name}",
             "${cities[3].name}",
             "${cities[4].name}"],
   "times": ["${cities[0].shortTime}",
             "${cities[1].shortTime}",
             "${cities[2].shortTime}",
             "${cities[3].shortTime}",
             "${cities[4].shortTime}"],
  "populations": ["${cities[0].population}",
                  "${cities[1].population}",
                  "${cities[2].population}",
                  "${cities[3].population}",
                  "${cities[4].population}"]
})
```

JSON Data: Results


Ajax: Handling Server Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/ajax-data/
Show Cities

Getting JSON Data from Server, Building HTML Table

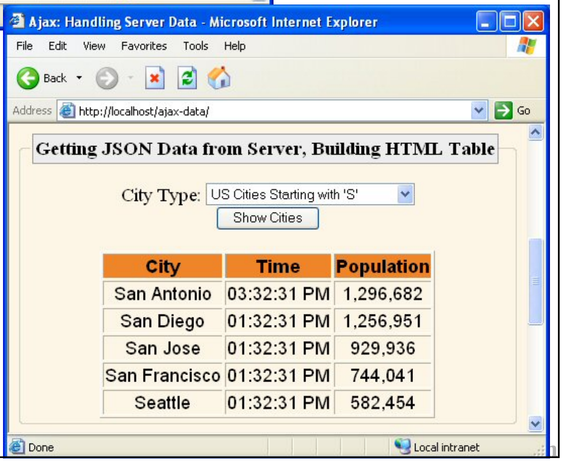
City Type: Largest Five US Cities


Ajax: Handling Server Data - Microsoft Internet Explorer
Address http://localhost/ajax-data/
Show Cities
Done
Local intranet

Getting JSON Data from Server, Building HTML Table

City Type: Second Five US Cities

City	Time	Population
Philadelphia	04:32:09 PM	1,448,396
San Antonio	03:32:09 PM	1,296,682
San Diego	01:32:09 PM	1,256,951
Dallas	03:32:09 PM	1,232,940
San Jose	01:32:09 PM	929,936


Ajax: Handling Server Data - Microsoft Internet Explorer
Address http://localhost/ajax-data/
Show Cities
Done
Local intranet

Getting JSON Data from Server, Building HTML Table

City Type: US Cities Starting with 'S'

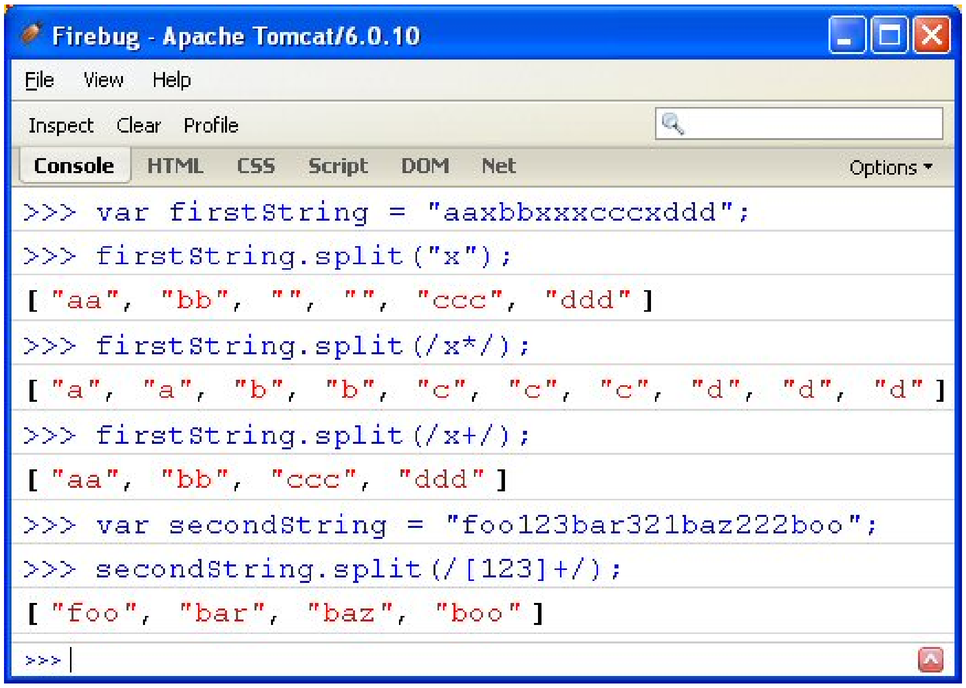
City	Time	Population
San Antonio	03:32:31 PM	1,296,682
San Diego	01:32:31 PM	1,256,951
San Jose	01:32:31 PM	929,936
San Francisco	01:32:31 PM	744,041
Seattle	01:32:31 PM	582,454

Handling String Data

Basic Tasks

- **General Approach**
  - Server-side code invents a custom data format
  - Client-side code parses it
- **Specific Common Approach**
  - Server-side code sends delimited strings
  - Client-side code uses String.split to break strings into arrays
- **String.split in JavaScript**
  - Quite similar to String.split in Java
  - With a one-char delimiter, use single or double quotes
  - With a regular expression, use slashes
    - JavaScript regex's similar to Perl (and Java) regular expressions
    - More details will be given in a later section
- **Online references**
  - http://www.evolt.org/article/Regular_Expressions_in_JavaScript/17/36435/
  - http://www.javascriptkit.com/javatutors/re.shtml

String.split: Example


Firebug - Apache Tomcat/6.0.10
File View Help
Inspect Clear Profile
Console HTML CSS Script DOM Net Options

```
>>> var firstString = "aaxbbxxxcccxddd";
>>> firstString.split("x");
[ "aa", "bb", "", "", "ccc", "ddd" ]
>>> firstString.split(/x*/);
[ "a", "a", "b", "b", "c", "c", "c", "d", "d", "d" ]
>>> firstString.split(/x+/);
[ "aa", "bb", "ccc", "ddd" ]
>>> var secondString = "foo123bar321baz222boo";
>>> secondString.split(/[123]+/);
[ "foo", "bar", "baz", "boo" ]
>>>
```

Handling String Data: Example

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
      - Break it into array with String.split and regular expression delimiters
      - Access array elements (perhaps using String.split again)
    - 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

Initiate Request

```
function stringCityTable(inputField, resultRegion) {
  var address = "show-cities";
  var data = "cityType=" + getValue(inputField) +
             "&format=string";
  ajaxPost(address, data,
           function(request) {
             showStringCityInfo(request, resultRegion);
           });
}
```

Handle Response

```
function showStringCityInfo(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var rawData = request.responseText;
    var columnStrings = rawData.split(/\n/);
    var names = columnStrings[0].split("#");
    var times = columnStrings[1].split("#");
    var populations = columnStrings[2].split("#");
    var headings = ["City", "Time", "Population"];
    var columns = [names, times, populations];
    var table = getTable(headings, columns);
    htmlInsert(resultRegion, table);
  }
}
```

HTML Code

```
...
<fieldset>
  <legend>Getting String Data from Server, Building HTML Table
  </legend>
  <form action="#">
   <label for="city-type-3">City Type:</label>
   <select id="city-type-3">
     <option value="top-5-cities">Largest Five US Cities</option>
     <option value="second-5-cities">Second Five US Cities</option>
     <option value="cities-starting-with-s">
       US Cities Starting with 'S'</option>
     <option value="superbowl-hosts">
       Most Recent Superbowl Hosts</option>
    </select>
   <br/>
   <input type="button" value="Show Cities"
          onclick='stringCityTable("city-type-3",
                                   "string-city-table")'/>
  </form>
  <p/>
  <div id="string-city-table"></div>
</fieldset>...
```

Server Design: MVC

- **Logic**
  - No changes to basic logic
  - Only addition is logic to decide which results page applies
- **Presentation**
  - Build a plain-text page
  - Embed data between delimiters

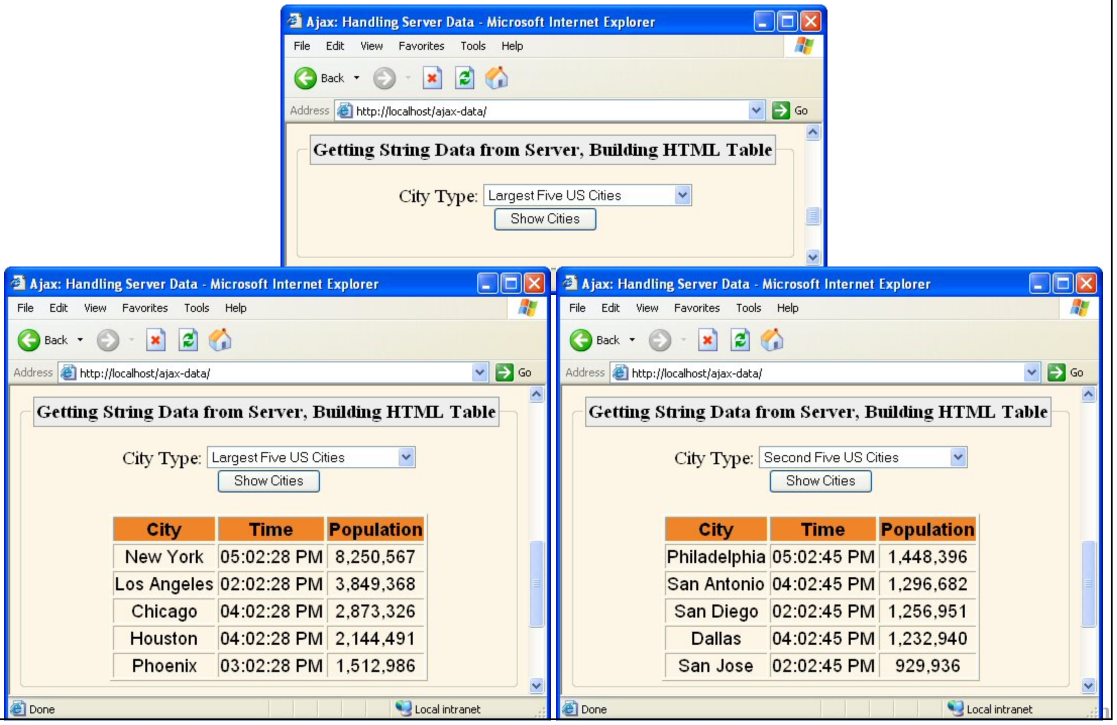
Servlet Code

```
public class ShowCities extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    ...
    if ("xml".equals(format)) {
      response.setContentType("text/xml");
      outputPage = "/WEB-INF/results/cities-xml.jsp";
    } else if ("json".equals(format)) {
      response.setContentType("text/javascript");
      outputPage = "/WEB-INF/results/cities-json.jsp";
    } else {
      response.setContentType("text/plain");
      outputPage = "/WEB-INF/results/cities-string.jsp";
    }
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
  }
```

JSP Code (/WEB-INF/results/cities-string.jsp)

```
${cities[0].name}#${cities[1].name}#...#${cities[4].name}
${cities[0].shortTime}#...#${cities[4].shortTime}
${cities[0].population}#...#${cities[4].population}
```

String Data: Results



Ajax: Handling Server Data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost/ajax-data/
Go
Done
Local intranet

Getting String Data from Server, Building HTML Table

City Type: Largest Five US Cities

Show Cities

Getting String Data from Server, Building HTML Table

City Type: Largest Five US Cities

Show Cities

City	Time	Population
New York	05:02:28 PM	8,250,567
Los Angeles	02:02:28 PM	3,849,368
Chicago	04:02:28 PM	2,873,326
Houston	04:02:28 PM	2,144,491
Phoenix	03:02:28 PM	1,512,986

Getting String Data from Server, Building HTML Table

City Type: Second Five US Cities

Show Cities

City	Time	Population
Philadelphia	05:02:45 PM	1,448,396
San Antonio	04:02:45 PM	1,296,682
San Diego	02:02:45 PM	1,256,951
Dallas	04:02:45 PM	1,232,940
San Jose	02:02:45 PM	929,936

Combination Data

Idea

- **Earlier**
  - Server
    - Decided what datatype to return based on "format" parameter
  - Client
    - Hardcoded "format" value
    - Hardcoded response handler function
- **Now**
  - Server
    - No change. Still uses "format" param in same way.
  - Client
    - Gets "format" value from textfield
    - Decides on response handler function based on textfield value

JavaScript

```
function cityTable(cityTypeField, formatField,
                   resultRegion) {
  var address = "show-cities";
  var cityType = getValue(cityTypeField);
  var format = getValue(formatField);
  var data = "cityType=" + cityType +
             "&format=" + format;
  var responseHandler = findHandler(format);
  ajaxPost(address, data,
           function(request) {
             responseHandler(request, resultRegion);
           });
}

function findHandler(format) {
  if (format == "xml") {          // == is ok for strings!
    return(showXmlCityInfo);
  } else if (format == "json") {
    return(showJsonCityInfo);
  } else {
    return(showStringCityInfo);
  }
}
```

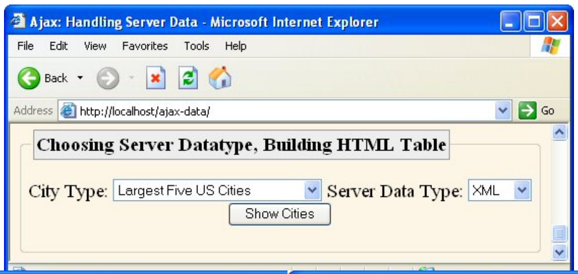
HTML

```
<fieldset>
  <legend>Choosing Server Datatype...</legend>
  <form action="#">
   <label for="city-type-4">City Type:</label>
   <select id="city-type-4">
     <option value="top-5-cities">Largest Five ...</option>
     ...
    </select>
  <label for="data-type">Server Data Type:</label>
   <select id="data-type">
     <option value="xml" selected="selected">XML</option>
     <option value="json">JSON</option>
     <option value="string">String</option>
    </select>
   <br/>
   <input type="button" value="Show Cities"
          onclick='cityTable("city-type-4", "data-type",
                                   "city-table")'/>
  </form>
  <p/>
  <div id="city-table"></div>
</fieldset>
```

Server-Side Code

- No changes whatsoever

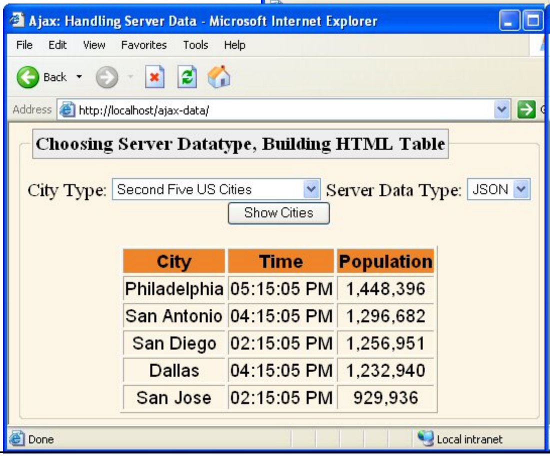
Combination Data: Results



Choosing Server Datatype, Building HTML Table

City Type: Largest Five US Cities   Server Data Type: XML

Ajax: Handling Server Data - Microsoft Internet Explorer
http://localhost/ajax-data/
Show Cities

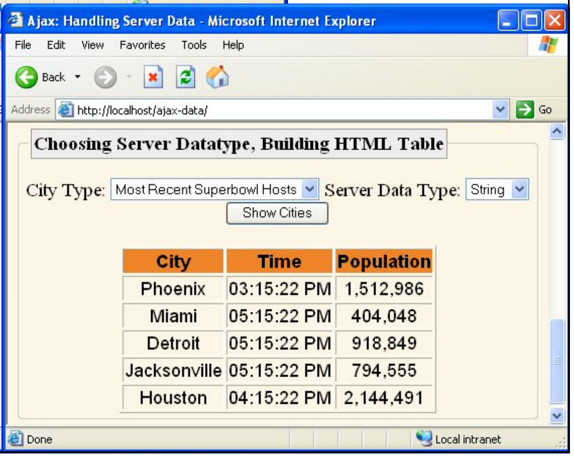


Choosing Server Datatype, Building HTML Table

City Type: Second Five US Cities   Server Data Type: JSON

City	Time	Population
Philadelphia	05:15:05 PM	1,448,396
San Antonio	04:15:05 PM	1,296,682
San Diego	02:15:05 PM	1,256,951
Dallas	04:15:05 PM	1,232,940
San Jose	02:15:05 PM	929,936

Ajax: Handling Server Data - Microsoft Internet Explorer
http://localhost/ajax-data/
Show Cities
Done
Local intranet



Choosing Server Datatype, Building HTML Table

City Type: Most Recent Superbowl Hosts   Server Data Type: String

City	Time	Population
Phoenix	03:15:22 PM	1,512,986
Miami	05:15:22 PM	404,048
Detroit	05:15:22 PM	918,849
Jacksonville	05:15:22 PM	794,555
Houston	04:15:22 PM	2,144,491

Ajax: Handling Server Data - Microsoft Internet Explorer
http://localhost/ajax-data/
Show Cities
Done
Local intranet

Wrapup

Preview of Next Section

- **Problems with JSP pages in this section**
  - Repeated identical information for each of the five entries in the list of cities
  - Cannot handle city lists that are not five entries long
- **Handling variable-length data in JSP**
  - Bean
  - Custom tag
  - JSTL loop
  - Scripting loop

Summary

- **Parsing XML data**
  - Call request.responseXML
  - Call getElementsByTagName
  - Get body text via someElement.childNodes[0].nodeValue
- **Parsing JSON data**
  - Pass to eval
  - Treat as normal JavaScript object
- **Parsing string data**
  - Use String.split and (possibly) regular expressions
- **Server**
  - Use MVC

Questions?