INTERNET & WEB APPLICATION DEVELOPMENT

SWE 444

Fall Semester 2008-2009 (081)

Module7: AJAX

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives/Outline

- Objectives
  - Understand the role of AJAX
  - Learn how to use AJAX in your web applications
- Outline
  - Rich User Experience
  - AJAX
  - Why AJAX?
  - Real-Life Examples of AJAX Apps
  - Technologies Used In AJAX
  - Examples

Rich User Experience

- Take a look at a typical desktop application (Spreadsheet app, etc.)
- The program responses intuitively and quickly
- The program gives a user meaningful feedback's instantly
  - A cell in a spreadsheet changes color when you hover your mouse over it
  - Icons light up as mouse hovers them
- Things happen naturally
  - No need to click a button or a link to trigger an event

Characteristics of Conventional Web Applications

- "Click, wait, and refresh" user interaction
  - Page refreshes from the server needed for all events, data submissions, and navigation
- Synchronous "request/response" communication model
  - The user has to wait for the response
- Page-driven: Workflow is based on pages
  - Page-navigation logic is determined by the server

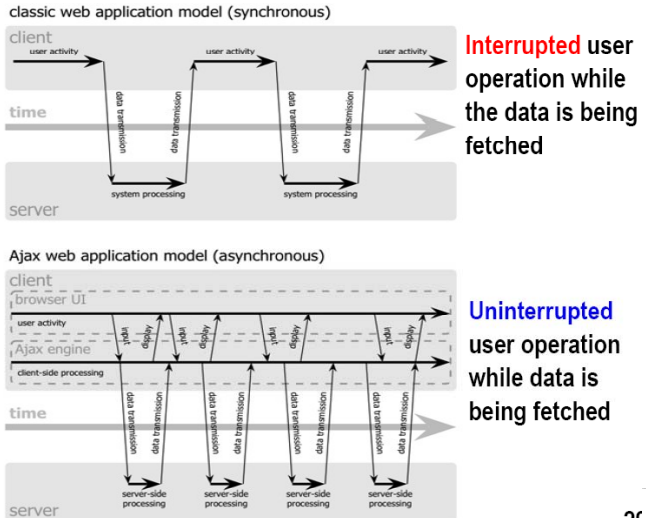
Issues of Conventional Web Application

- Interruption of user operation
  - Users cannot perform any operation while waiting for a response
- Loss of operational context during refresh
  - Loss of information on the screen
  - Loss of scrolled position
- No instant feedback's to user activities
  - A user has to wait for the next page
- Constrained by HTML
  - Lack of useful widgets
- These are the reasons why Rich Internet Application (RIA) technologies were born

AJAX

- AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML
- AJAX is a type of programming made popular in 2005 by Google (with Google Suggest)
- AJAX is not a new programming language, but a new way to use existing standards
- With AJAX you can create better, faster, and more user-friendly web applications
- AJAX is using JavaScript, namely the **XmlHttpRequest** object, to communicate asynchronously with a server-side component and dynamically update the source of an HTML page based on the resulting XML/Text response

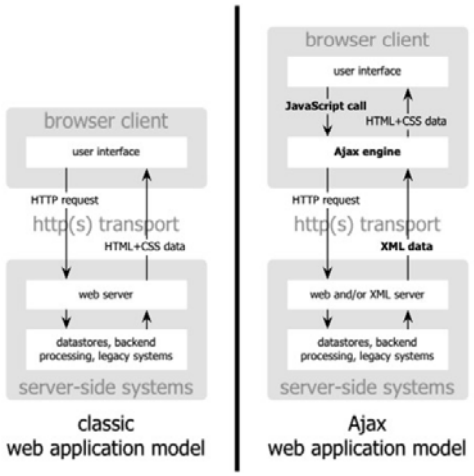
AJAX



Interrupted user operation while the data is being fetched

Uninterrupted user operation while data is being fetched

AJAX



AJAX

- DHTML plus Asynchronous communication capability through XMLHttpRequest
- Pros
  - Most viable RIA technology so far
  - Tremendous industry momentum
  - Several toolkits and frameworks are emerging
  - No need to download code & no plug-in required
- Cons
  - Still browser incompatibility
  - JavaScript is hard to maintain and debug

Why AJAX?

- Intuitive and natural user interaction
  - No clicking required
  - Mouse movement is a sufficient event trigger
- "Partial screen update" replaces the "click, wait, and refresh" user interaction model
  - Only user interface elements that contain new information are updated (fast response)
  - The rest of the user interface remains displayed without interruption (no loss of operational context)
- Data-driven (as opposed to page-driven)
  - UI is handled in the client while the server provides data
- Asynchronous communication replaces "synchronous request/response model."
  - A user can continue to use the application while the client program requests information from the server in the background
  - Separation of displaying from data fetching

Real-Life Examples of AJAX Apps

- Google maps
  - http://maps.google.com/
- Goolgle Suggest
  - http://www.google.com/webhp?complete=1&hl=en
- Gmail
  - http://gmail.com/
- Yahoo Maps
  - http://maps.yahoo.com/
- Many more are popping everywhere

Technologies Used In AJAX

- JavaScript
  - JavaScript function is called when an event in a page occurs
  - Glue for the whole AJAX operation
- DOM
  - API for accessing and manipulating structured documents
  - Represents the structure of XML and HTML documents
- CSS
  - Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript
- XMLHttpRequest
  - JavaScript object that performs asynchronous interaction with the server

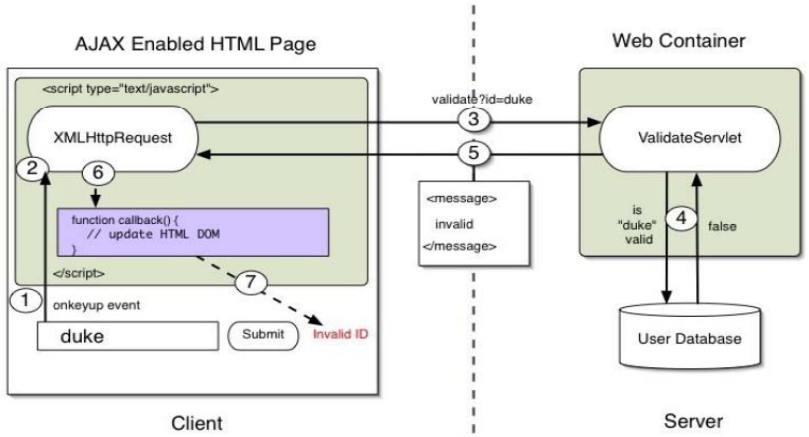
XMLHttpRequest

- JavaScript object
- Adopted by modern browsers
  - Mozilla, Firefox, Safari, and Opera
  - Communicates with a server via standard HTTP GET/POST
- `XMLHttpRequest` object works in the background for performing asynchronous communication with the backend server
  - Does not interrupt user operation

Server-Side AJAX Request Processing

- Server programming model remains the same
  - It receives standard HTTP GETs/POSTs
  - Can use Servlet, JSP, ASP, PHP...
- With minor constraints
  - More frequent and finer-grained requests from client
  - Response content type can be
    - text/xml
    - text/plain
    - text/json
    - text/javascript

AJAX: Sample App



Steps of AJAX Operation

1. A client event occurs
2. An XMLHttpRequest object is created
3. The XMLHttpRequest object is configured
4. The XMLHttpRequest object makes an async. request
5. The ValidateServlet returns an XML document containing the result
6. The XMLHttpRequest object calls the callback() function and processes the result
7. The HTML DOM is updated

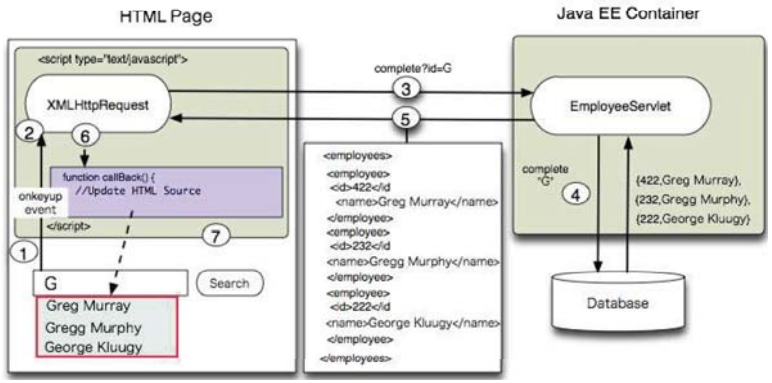
XMLHttpRequest Properties

- onreadystatechange
  - Set with a JavaScript event handler that fires at each state change
- readyState – current status of request
  - 0 = uninitialized
  - 1 = loading
  - 2 = loaded
  - 3 = interactive (some data has been returned)
  - 4 = complete
- status
  - HTTP Status returned from server: 200 = OK

XMLHttpRequest Properties

- responseText
  - String version of data returned from the server
- responseXML
  - XML document of data returned from the server
- statusText
  - Status text returned from server

AJAX: Another App



AJAX: Coding Involved

- index.jsp Page Auto-Complete Form

```
<form name="autofillform" action="autocomplete"
  method="get">
  <input type="text" size="20" autocomplete="off"
  id="completeField" name="id"
  onkeyup="doCompletion();">
  <input id="submit_btn" type="Submit"
  value="Lookup Employee">
</form>
```

AJAX: Coding Involved (2)

- AutoComplete Event Handler

```
function doCompletion() {
   if (completeField.value == "") {
        clearTable();
   } else {
        var url = "autocomplete?action=complete&id=" +
                  escape(completeField.value);
        var req = initRequest(url);
        req.onreadystatechange = function() {
           if (req.readyState == 4) {
                if (req.status == 200) {
                      parseMessages(req.responseXML);
                } else if (req.status == 204){
                      clearTable();
                        }
                }
           };
        req.open("GET", url, true);
        req.send(null);
   }
}
```


AJAX: Coding Involved (3)

- AutoComplete XMLHttpRequest

```
function initRequest(url) {
  if (window.XMLHttpRequest) {
      return new XMLHttpRequest();
  } else if (window.ActiveXObject) {
      isIE = true;
      return new ActiveXObject("Microsoft.XMLHTTP");
  }
}
```


AJAX: Coding Involved (4)

- **AutoComplete Servlet doGet()**

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException {
...
String targetId = request.getParameter("id");
Iterator it = employees.keySet().iterator();
while (it.hasNext()) {
   EmployeeBean e = (EmployeeBean)employees.get((String)it.next());
   if ((e.getFirstName().toLowerCase().startsWith(targetId) ||
        e.getLastName().toLowerCase().startsWith(targetId)) &&
   !targetId.equals("")) {
        sb.append("<employee>");
        sb.append("<id>" + e.getId() + "</id>");
        sb.append("<firstName>" + e.getFirstName() + "</firstName>");
        sb.append("<lastName>" + e.getLastName() + "</lastName>");
        sb.append("</employee>");
        namesAdded = true; } // if
} // while
if (namesAdded) {
   response.setContentType("text/xml");
   response.setHeader("Cache-Control", "no-cache");
   response.getWriter().write("<employees>" + sb.toString() + "</employees>"
} else {
   response.setStatus(HttpServletResponse.SC_NO_CONTENT);
}
} // doGet
```


AJAX: Coding Involved (5)

- Processing the response

```
function parseMessages(responseXML) {
clearTable();
var employees =
  responseXML.getElementsByTagName("employees")[0];
if (employees.childNodes.length > 0) {
  completeTable.setAttribute("bordercolor", "black");
  completeTable.setAttribute("border", "1");
} else {
  clearTable();
}
for (loop = 0; loop < employees.childNodes.length; loop++)
  {
  var employee = employees.childNodes[loop];
  var firstName =
  employee.getElementsByTagName("firstName")[0];
  var lastName =
  employee.getElementsByTagName("lastName")[0];
  var employeeId = employee.getElementsByTagName("id")[0];
  appendEmployee(firstName.childNodes[0].nodeValue,lastName
  .childNodes[0].nodeValue,
  employeeId.childNodes[0].nodeValue);
}
}
```

Q & A

Resources

- W3Schools AJAX Tutorial
  - http://www.w3schools.com/ajax/default.asp
- "AJAX Programming" [Sang Shin]
  - http://www.javapassion.com/ajaxcodecamp/
- http://developer.mozilla.org/en/docs/AJAX:Getting_Started
- Ajax and XMLHttpRequest Tutorial
- Another Ajax Tutorial
- Ajax (programming) - Wikipedia
- Ajax Help and Tutorials
- CodeProject AJAX and PHP Building Responsive Web Applications - Chapter 1 AJAX and the Future of Web Applications. Free source code and programming help
- Ajax Resource Center
- Open Directory - Computers Programming Languages JavaScript AJAX
- Safari Books Online - 9780137142309 - Deitel® Developer Series AJAX, Rich Internet Applications, and Web Development for Programmers