Chapter 3: Interactive Web Applications

3.1 Web Server Interfaces
3.2 Server-Side Scripting (PHP)
3.3 Database Integration
3.4 Integration of Client-Side and Server-Side Scripts (AJAX)
3.5 Server-Side Programming with Java (Servlets, JSP)
Database Management Systems: 
A Quick Reminder

• Database:
  – Structured collection of data items
  – Stored persistently
  – Provides access to a common data pool for multiple users

• Database Management System (DBMS):
  – Collection of programs for administration and usage of a database
  – Various base models for DBMS:
    » Old: network model, hierarchical model
    » Dominant: relational model
    » Alternative: object-oriented model

• Relational databases:
  – Good methodological support for design of data schema
  – Standardized language interface SQL (Structured Query Language)
Prerequisites and Basic Architecture

Database server

Database 1
- Table 1.1
- Table 1.2

Database 2
- Table 2.1
- Table 2.2

Administration software

User programs

Other Servers (e.g. Web Server)
MySQL

- Open source software system
  - Frequently used also in commercial context
  - www.mysql.com
- Software package providing:
  - Database server (mysql)
  - Administration program (mysqladmin)
  - Command line interface (mysql)
  - Various utility programs
- Communication between programs on local host: socket interface
  - Bidirectional data stream exchange between programs
  - Similar to files
Before Creating Anything in the Database...

• Using a database requires careful *information design*.
• Which are the data to be stored?
• Are there existing data to connect to?
• What is the **schema** of the data to be stored?
  – Eg. Entity-Relationship diagrams as a tool
  – Transformation into relational database schema (table design)
• Once a database if filled with data and in use, it is rather difficult to modify!
  – Database schema design has to be carried out with great care!
• Most important rule: Avoid redundant storage of information
Creating Database Tables (1)

• Prerequisites:
  – Database server running
  – Socket connection between programs intact
  – User accounts with adequate privileges known

• First step: Create database
  – Container for many tables
  – Requires special privileges
  – Example SQL:
    create database music;

• Second step: Select database
  – Sets the context for further interactions
  – Example SQL:
    use music
Creating Database Tables (2)

• Third step: Create tables
  – According to earlier design
  – Each table should provide a unique identifier (primary key)
  – SQL Example:
    
    ```sql
    create table song (code VARCHAR(5), title VARCHAR(20),
    artist VARCHAR(20), composer VARCHAR(20), runtime INT);
    ```
  – Further steps: Defining keys, indices etc.

• Fourth step: Fill tables with data
  – Simplest case: Individual SQL commands
  – Better: Import from structured data file
  – Frequent: Special programs for importing and creating data
  – SQL Example:
    
    ```sql
    insert into song
    values ('1', 'One', 'U2', 'Adam Clayton, Bono, Larry Mullen & The Edge', 272);
    ```
SQL Monitor Output

```
mysql> describe song;
+----------------+-----------------+------+-----+-----------------+-------------------+
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>varchar(5)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>artist</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>composer</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>runtime</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>
+----------------+-----------------+------+-----+-----------------+-------------------+
5 rows in set (0.01 sec)
```

mysql>
Queries with SQL

mysql> select * from song;
+--------+-----------------+----------+-----------------+----------+
<table>
<thead>
<tr>
<th>code</th>
<th>title</th>
<th>artist</th>
<th>composer</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>One</td>
<td>U2</td>
<td>Adam Clayton, Bono</td>
<td>272</td>
</tr>
<tr>
<td>2</td>
<td>In the End</td>
<td>Linkin Park</td>
<td></td>
<td>219</td>
</tr>
<tr>
<td>3</td>
<td>Wheel in the Sky</td>
<td>Journey</td>
<td></td>
<td>252</td>
</tr>
<tr>
<td>4</td>
<td>Lady in Black</td>
<td>Uriah Heep</td>
<td></td>
<td>281</td>
</tr>
<tr>
<td>5</td>
<td>Smoke on the Water</td>
<td>Deep Purple</td>
<td></td>
<td>481</td>
</tr>
</tbody>
</table>
+--------+-----------------+----------+-----------------+----------+
5 rows in set (0.00 sec)

mysql> select title from song where runtime>250;
+--------+
<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
</tr>
<tr>
<td>Wheel in the Sky</td>
</tr>
<tr>
<td>Lady in Black</td>
</tr>
<tr>
<td>Smoke on the Water</td>
</tr>
</tbody>
</table>
+--------+
4 rows in set (0.00 sec)
Databases, PHP and MySQL

• Special libraries for database access:
  – "Database extensions"
  – Generic for all database systems
• For specific database systems:
  – "Vendor specific database extensions"
• For MySQL:
  – MySQL-specific database extensions to PHP
Connecting to a Database from PHP

• First step: **Connect** to server
  – Establish a connection for data exchange between Web Server/PHP plugin and database server
  – Often local (sockets), if both programs on same machine
  – Requires hostname, (database) username, password
  – PHP function: `mysql_connect()`
    » Returns a link (resource) which can be used for `mysql_close()`

• Second step: **Select** a database
  – Corresponds to the SQL command `use`
  – Requires database name (and possibly link to server)
  – PHP function: `mysql_select_db()`
    » Returns Boolean result (success)
Example: Connecting to Database

```php
<?php

$link = mysql_connect('localhost','root','demopw')
    or die ('Could not connect: '.mysql_error());
echo 'Connected.<br/>';

mysql_select_db('music')
    or die ('Could not select db.');
echo 'DB selected.<br/>';

...
?>
```
Sending Database Queries from PHP

• Basic idea (as in all programming language/database integrations):
  – SQL queries are given as strings to library functions

• Most important function in MySQL extensions to PHP:
  `mysql_query()`
  – Requires SQL query as parameter (optionally link to server as 2nd param.)
  – "Query" includes also INSERT, UPDATE, DELETE, DROP (SQL)!

• Return value in case of SELECT, SHOW, DESCRIBE and similar:
  – Result set represented by resource value
  – Special functions to retrieve result data as PHP data structures
    – `mysql_num_rows()`
      » Number of rows returned
    – `mysql_fetch_array()`
      » Reads one row of data and transforms it into an array
      » Makes the next row available
Example: Reading Data From a Query in PHP

```php
<?php
...
$query = 'SELECT * FROM song';
$result = mysql_query($query);

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    foreach ($row as $element) {
        echo $element;
        echo ',', ';
    }
    echo("<br/>");
...?
```

$dbaccess.php
Creating HTML Output From SQL Query (1)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
  <title>Database table in HTML</title>
</head>

<?php
$link = mysql_connect('localhost','root','demopw')
or die ('Could not connect: '.mysql_error());
mysql_select_db('music') or die ('Could not select db.');
?>

---

Ludwig-Maximilians-Universität München
Prof. Hußmann
Multimedia im Netz – 3 - 62
Creating HTML Output From SQL Query (2)

...<br>

<body><h1>The following table is retrieved from MySQL:</h1>
<table><tr><td><?php
$query = 'SELECT * FROM song';
$result = mysql_query($query)
or die ('Query failed'.mysql_error());
while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr><td>
    foreach ($row as $element) {
        echo "\t\t<td>"
        echo $element;
        echo "</td>\n"
    }
    echo "</tr><\n"
}
?></table>
Creating HTML Output From SQL Query (3)

...  
<?php  
    mysql_free_result($result);  
    mysql_close($link);  
?>  

</body>  
</html>
Chapter 3: Interactive Web Applications

3.1 Web Server Interfaces
3.2 Server-Side Scripting (PHP)
3.3 Database Integration
3.4 Integration of Client-Side and Server-Side Scripts (AJAX)
3.5 Server-Side Programming with Java (Servlets, JSP)

Asynchronous JavaScript + HTML (AJAX)

• James Garrett 2005: 

• Catchy name for an idea which was in use already at the time:
  – Google Suggest
  – Google Maps

• Basic idea:
  – Loading data from server is decoupled from changes in the presentation

• Advantages:
  – User can interact fluidly with the application
  – Information from server is fetched at regular intervals - display can always stay up-to-date

• AJAX is not a technology, it is a combination of known technologies
  – XHTML, CSS, DOM, XML, XSLT, JavaScript, XMLHttpRequest

• There are AJAX-like applications which use neither JavaScript nor HTML
  – E.g. using Flash and querying servers in the background
Asynchronicity

Examples:
Web mail access
Autocompletion of forms (e.g. City based on Zip code)
AJAX and Client-Side Scripting

- AJAX applications are programs executed in the Web browser
  - Require a runtime environment
  - Usually programmed in JavaScript
- AJAX applications need to modify or construct HTML to be displayed in the browser
  - Requires access to loaded/displayed HTML
  - *Domain Object Model (DOM)* is used for accessing and manipulating page content
JavaScript Object Tree

- Elements of the displayed document and other information can be accessed and manipulated
- Navigation:
  - Mostly selection by "id"
  - Starting point is often "document" object
DOM Reminder

• DOM is a collection of functions which make it possible to access and manipulate HTML and XML documents in the browser

• DOM ist a standardised API (Application Programming Interface)
  – Usable with several programming languages

• Examples of DOM object properties and methods:
  nodeName, nodeValue, nodeType, attributes
  getElementsByTagName()
  parentNode, hasChildNodes();
  childNodes, firstChild, lastChild, previousSibling, nextSibling;
  createElement(); createTextNode();
  insertBefore(), replaceChild(), removeChild(),
  appendChild();
AJAX and Server-Side Scripting

• AJAX applications make particular sense when the data loaded from the server changes dynamically
  – PHP scripts or other server-side dynamics
  – Database connectivity

• For ease of understanding (only!):
  – Most examples in the following deal with static Web pages only
Request Construction and Handling

• Main functionalities required:
  – Construction of a request to be sent to the server
  – Sending a request to the server
  – Waiting (asynchronously) until server responds
  – Calling functions to analyze server response

• All these functionalities are realized in one single object (in the sense of object-orientation):
  – XMLHttpRequest
Basic Control Flow

1. Invoke some javascript
2. Setup request object
3. Request callback
4. Send HTTP request
5. Return control
6. HTTP Response
7. Update page DOM

http://www.ibm.com/developerworks, Dojo framework
XMLHttpRequest

- Outlook Web Access for Internet Explorer 5 (end 90s):
  - XMLHttpRequest object invented at Microsoft
  - Realized as ActiveX object
- Mozilla 1.4 (Netscape 7.1) and derivates (including Firefox):
  - Native XMLHttpRequest object for JavaScript
  - Independent of Active X
- Other manufacturers:
  - Followed step by step: Konqueror, Apple Safari, Opera, iCab
- Since Internet Explorer 7 ActiveX no longer required
  - Just JavaScript
- Under W3C standardization (Working draft August 2009)
- Long term situation for creating XMLHttpRequest object will be:
  ```javascript
  var XMLHTTP = new XMLHttpRequest();
  ```
- Currently we have to fight with browser incompatibilities!
  - Frameworks like Prototype can help
Platform Independent Creation of XMLHttpRequest

```javascript
var XMLHTTP = null;
if (window.XMLHttpRequest) {
    XMLHTTP = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    try {
        XMLHTTP = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (ex) {
        try {
            XMLHTTP = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (ex) {}
    }
}
```
Construction of a HTTP Request

• open() method of XMLHttpRequest object
  – Note: There is no interaction with the server involved, despite the name

• Required parameters:
  – HTTP method: GET, POST or HEAD
  – URL to send the request to

• Optional parameters:
  – Asynchronous or synchronous treatment (default asynchronous = true)
  – Username and password for authentication

• Examples:
  XMLHttpRequest.open("GET", "fibonacci.php?fib=12")
  XMLHttpRequest.open("POST", "/start.html", false, un, pwd);
Sending a Request

• Before sending: `XMLHTTP.setRequestHeader()`
  – Setting headers for the request
  – Recommended: `Content-Type` (MIME type)

• `XMLHTTP.send()`
  – Sends request to server

• Parameter:
  – In the simplest case (in particular GET method): `null`
  – For more complex cases:
    "Request entity body" is given as parameter
    » Mainly for POST method
States of an XMLHttpRequest Object

0 UNSENT

Just created

1 OPENED

Request constructed, sending

2 HEADERS_RECEIVED

Header part of response arrived

3 LOADING

Body is being received

4 DONE

Response has been received completely
Asynchronous Reaction by Event Handler

- In order to react to the received response:
  - Function has to be called when state 4 is reached
- Registering an event handler:
  - Callback function, called when event takes place
  - Similar to event handling for user interfaces (Java Swing, Flash)
- For Ajax:
  - Callback method registered with XMLHttpRequest object
  - Event = State 4 is reached
  - More general: Called at any state change
- \( \text{XMLHTTP.onreadystatechange} = \text{function}; \)
  - \( \text{XMLHTTP.readystatechange} \) gives current state (as number)
Example: Very Simple Request

...<body>
<script type = "text/javascript">
    var XMLHttpRequest = new XMLHttpRequest();

    function dataOutput() {
        if (XMLHttpRequest.readyState == 4) {
            var d = document.getElementById("data");
            d.innerHTML += XMLHttpRequest.responseText;
        }
    }

    window.onload = function() {
        XMLHttpRequest.open("GET", "data.txt", true);
        XMLHttpRequest.onreadystatechange = dataOutput;
        XMLHttpRequest.send(null);
    }
</script>

</body>

<p id="data">Data from server: </p>
</html>
AJAX and XML

• The server response (essentially text) needs to be analysed
• XML
  – Supports arbitrarily structured information
  – Is fully supported by JavaScript and DOM
• Servers should return data as XML
• Problem (currently):
  – Browser incompatibilities
Example XML Data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet totalResultsAvailable="24900000"
  totalResultsReturned="10">
  <Result>
    <Title>AJAX - Wikipedia</Title>
    <Summary>Background about the web development technique for creating interactive web applications.</Summary>
  </Result>
  <Result>
    <Title>Ajax: A New Approach to Web Applications</Title>
    <Summary>Essay by Jesse James Garrett from Adaptive Path.</Summary>
    <Url>http://www.adaptivepath.com/p...s/000385.php</Url>
  </Result>
  <Result>
    <Title>AFC Ajax</Title>
    <Summary>Official site. Club information, match reports, news, and much more.</Summary>
    <Url>http://www.ajax.nl</Url>
  </Result>
</ResultSet>
```

From C.Wenz
AJAX Program Creating a HTML Table from XML

• Fixed HTML text:

```html
<body>
  <p>
    <span id="Anzahl">0</span> von
    <span id="Gesamt">0</span> Treffern:
  </p>

  <table id="Treffer">
    <thead>
      <tr><th>Titel</th><th>Beschreibung</th><th>URL</th></tr>
    </thead>
  </table>
</body>
```

Script has to fill the missing data from XML response. Basic structure of script as above.

From C. Wenz
Transformer Callback Function (1)

function DatenAusgeben() {
  if (XMLHTTP.readyState == 4) {
    var xml = XMLHttpRequest.responseTextXML;

    var anzahl = document.getElementById("Anzahl");
    var gesamt = document.getElementById("Gesamt");
    anzahl.innerHTML =
    xml.documentElement.getAttribute("totalResultsReturned");
    gesamt.innerHTML =
    xml.documentElement.getAttribute("totalResultsAvailable");

    var treffer = document.getElementById("Treffer");
    var tbody = document.createElement("tbody");

    var ergebnisse = xml.getElementsByTagName("Result");
    ...
  }
}
Transformer Callback Function (2)

... for (var i=0; i<ergebnisse.length; i++) {
    var zeile = document.createElement("tr";);
    var titel = document.createElement("td");
    var beschreibung = document.createElement("td");
    var url = document.createElement("td");
    var titeltext, beschreibungstext, urltext;
    for (var j=0; j<ergebnisse[i].childNodes.length; j++) {
        var knoten = ergebnisse[i].childNodes[j];
        switch (knoten.nodeName) {
            case "Title":
                titeltext = document.createTextNode(knoten.firstChild.nodeValue);
                break;
            case "Summary":
                beschreibungstext = document.createTextNode(knoten.firstChild.nodeValue);
                break;
            case "Url":
                urltext = document.createTextNode(knoten.firstChild.nodeValue);
                break;
        }
    }
}
Transformer Callback Function (2)

... for (var i=0; i<ergebnisse.length; i++) {
    ...
    for (var j=0; j<ergebnisse[i].childNodes.length; j++) {
        ...
        titel.appendChild(titeltext);
        beschreibung.appendChild(beschreibungtext);
        url.appendChild(urltext);

        zeile.appendChild(titel);
        zeile.appendChild(beschreibung);
        zeile.appendChild(url);
        tbody.appendChild(zeile);
    }
    treffer.appendChild(tbody);
}
AJAJ? – Simple Serialization with JSON

- XML Serialization of data
  - Tends to be long
  - Many redundant elements
  - Occupies a lot of bandwidth
- Alternative Serialization: JSON (JavaScript Object Notation)

```json
{
  "ResultSet":
  {
    "totalResultsAvailable": "24900000",
    "totalResultsReturned": 10,
    "Result":
    [
      {
        "Title": "AJAX - Wikipedia",
        "Url": "http:\/\/en.wikipedia.org\/wiki\/AJAX"
      },
      {
        "Title": "Ajax: A New Approach to Web Applications",
        "Url": "http:\/\/www.adaptivepath.com\/p.../000385.php"
      }
    ]
  }
}
```
Problems with AJAX

• Back button
  – Browsers do not store dynamically modified pages in history

• Polling
  – Browser send more requests at a more regular pace; i.e. the base assumptions for traffic engineering change

• Bookmarks
  – It is difficult to set a bookmark at a specific state of a dynamically created flow of pages
  – Solution attempts use the document-internal anchors (#)

• Indexing by search engines
Example: Bookmarking Support

When processing response:

```
location.hash = "#" + escape(url);
```

```
window.onload = function() {
    if (location.hash.length > 1) {
        url = unescape(location.hash.substring(1));
        ladeURL();
    }
}
```
Sajax: Framework for AJAX (in PHP)

- Example for a framework supporting Ajax
- Sajax (Simple Ajax)
  - [http://www.modernmethod.com/sajax](http://www.modernmethod.com/sajax)
  - Open Source
  - Framework (library) for several scripting languages, including PHP
- Abstracts from technical details of AJAX
  - Write AJAX applications without knowing about XMLHttpRequest
- Basic idea:
  - Create a server-side dynamic function (in PHP)
  - "Export" this function with Sajax (sajax_export('functionname'))
  - In the JavaScript section of the page, call sajax_show_javascript() (a PHP function generating JavaScript)
  - Corresponding to the server-side function, now a JavaScript function exists (x_functionname) which calls the server-side function asynchronously (i.e. a callback function is given as parameter)
Examples of AJAX Applications

- Maps: Google Maps, OpenStreetMap
- Office: AjaxWrite, nexImage
- Social software: Flickr, Del.icio.us, Last.fm, *VZ Netzwerke
- Mail: Google Mail
- Web search: Google Suggest
- CRM: 24SevenOffice