Slide 11.10: Server-side PHP scripts
Slide 12.2: AJAX (Asynchronous JavaScript and XML)
Home

PRINT VERSION

Programming Exercise III: A Location-Based Service (LBS) Using AJAX and HTML5 Technologies

(An Industry-Level, Second-to-None Comprehensive Specification)

Absolutely no copying others' works

Development Requirements

When start developing the exercise, follow the two requirements below:

- Have to use the LAMP and AJAX technologies.
- The system entry page must be located at `http://people.aero.und.edu/~userid/457/3/` and all pages must be hosted by `http://people.aero.und.edu/~userid/` .

Due Date and Submission Methods

Due on or before Wednesday, May 03, 2017 in class and have completed the following two tasks:

- Turn in
  - printouts of all source code including C/C++, (X)HTML, Java, JavaScript, JDBC, Perl, PHP, PL/SQL, Unix shell, SQL (especially `create` commands), XML, and whatever languages†,
  - the password for displaying the source code online (only one password for all interfaces and all exercises and the code will be examined during the demonstration too), and
  - a report (single spacing and no more than five pages) with the following sections:

    1. Title
    2. Introduction
    3. Literature review
    4. System structure
    5. The methods applied
    6. Experiment results
    7. Conclusion
    8. References
    9. Appendix: User manual

- Send an email to the instructor at wenchen@cs.und.edu to set up an appointment to demonstrate your exercise to the instructor individually, so misunderstanding would be minimized. The mail lists the times you will be available.

†Note that you are allowed to use any languages and tools for this exercise, but the exams will focus on LAMP and AJAX technologies unless otherwise specified.

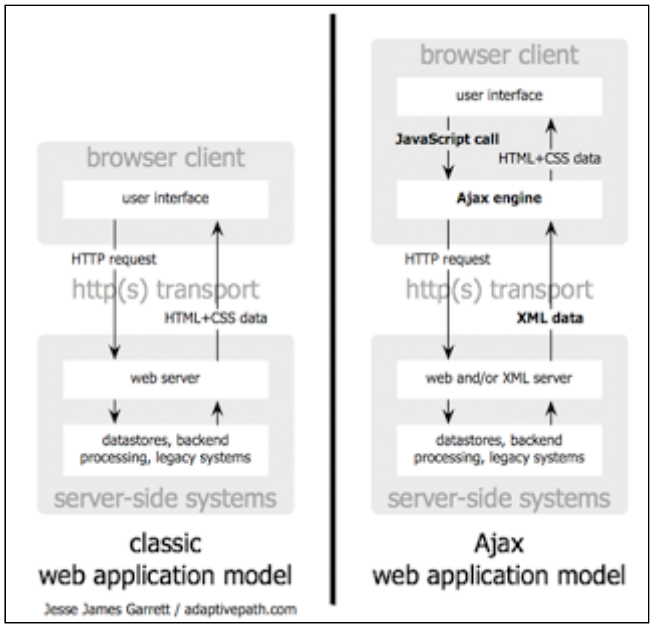
Objectives

Design and implement a location-based service (LBS) using HTML5, LAMP, and AJAX technologies. Two themes of this exercise are (i) using HTML5 and LAMP to create a location-based service and (ii) using AJAX to make the LBS more interactive, interesting, and user-friendly.

AJAX (Asynchronous JavaScript and XML)

The AJAX is used to create advanced web applications. It is the art of exchanging data with a server, and updating parts of a web page—without reloading the whole page. AJAX is not a new programming language, but a new way to use existing standards. The technologies include:

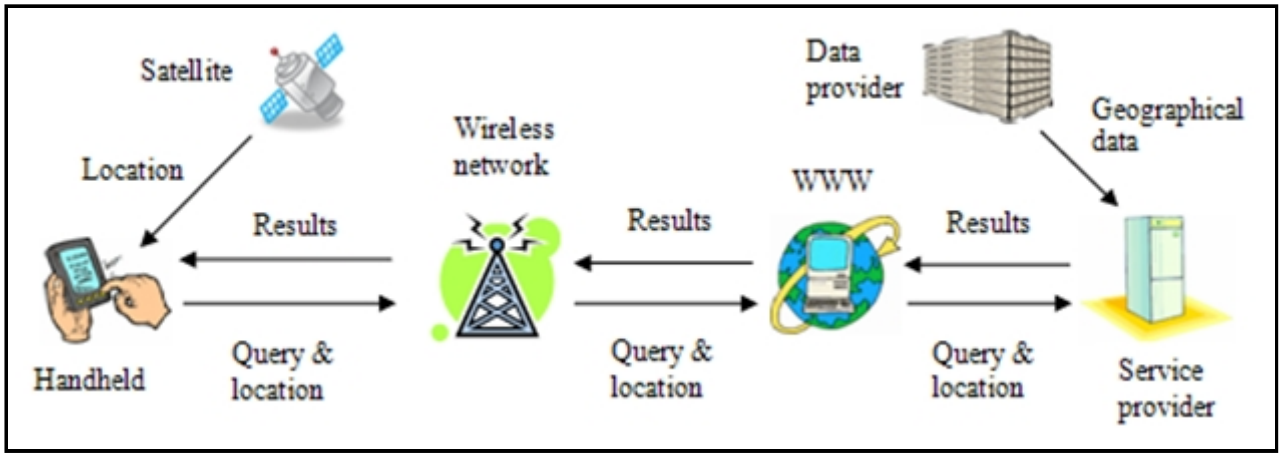
- HTML,
- JavaScript,
- XML, and
- MySQL database.



Location-Based Service (LBS)

A system structure of generic location-based services includes five major components:

a. *Mobile handheld devices*, which are small computers that can be held in one hand. For most cases, they are smartphones. However, desktop computers will be used to grade and demonstrate this exercise instead.
b. *Positioning system*, which is a navigation satellite system that provides location and time information to anyone with a receiver. This exercise uses HTML5 Geolocation to find the users' locations.
c. *Mobile and wireless networks*, which relay the query and location information from devices to service providers and send the results from the providers to devices.
d. *Service providers*, which provide the location-based services such as location-aware advertisements.
e. *Geographical data providers*, which are databases storing a huge amount of geographical data such as information about restaurants and gas stations. One example is GeoNames.



One example of LCSs is Foursquare.

HTML5 Geolocation

The HTML5 Geolocation API allows the web page to receive the geographical position of a mobile user pending permission. The following demonstration shows how to (i) get the user's latitude and longitude, (ii) mark the user's position on a map, and (iii) draw a route between two locations.

- Find latitude and longitude!
- Mark the map!
- Draw a direction!

The following program shows how to find the user's current latitude and longitude:

getCoord.html

```
01 <!DOCTYPE HTML>
02 <html>
03  <body>
04   Click the button to get your coordinates:
05   <button onclick="getLocation( )">Get Them!</button>
06   <p id="demo"></p>
07
08   <script>
```

```
    var  x = document.getElementById( "demo" );

    function  getLocation( ) {
     if ( navigator.geolocation )
      navigator.geolocation.getCurrentPosition( showPosition, showError );
     else
      x.innerHTML = "Geolocation is not supported by this browser.";
    }

    function  showPosition( position ) {
     x.innerHTML  = "Latitude: "  + position.coords.latitude;
     x.innerHTML += "Longitude: " + position.coords.longitude;
    }

    function  showError( error ) {
     switch ( error.code ) {
      case error.PERMISSION_DENIED:
       x.innerHTML = "User is denied the request for Geolocation.";
       break;
      case error.POSITION_UNAVAILABLE:
       x.innerHTML = "Location information is unavailable.";
       break;
      case error.TIMEOUT:
       x.innerHTML = "The request to get user location timed out.";
       break;
      case error.UNKNOWN_ERROR:
       x.innerHTML = "An unknown error occurred.";
       break;
     }
    }
   </script>
  </body>
 </html>
```

The following program shows how to build map mashups:

markMap.html

```
<!DOCTYPE HTML>
<html>
 <body>
  Click the button to mark your position:
  <button onclick="getLocation( )">Mark It!</button>
  <p id="mapholder"></p>

  <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
  <script>
   var  mapholder = document.getElementById( "mapholder" );

   function  getLocation( ) {
    if ( navigator.geolocation )
     navigator.geolocation.getCurrentPosition( showPosition, showError );
    else
     mapholder.innerHTML = "Geolocation is not supported by this browser.";
   }

   function  showPosition( position ) {
    lat = position.coords.latitude;
    lon = position.coords.longitude;
    latlon = new google.maps.LatLng( lat, lon );
    mapholder.style.height ='400px';
    mapholder.style.width  ='700px';

    var  myOptions = {
     center: latlon,
     zoom: 14,
     mapTypeId: google.maps.MapTypeId.ROADMAP,
     mapTypeControl: false,
```

```
     navigationControlOptions: {
      style: google.maps.NavigationControlStyle.SMALL
     }
    };

    var  map = new google.maps.Map( document.getElementById( "mapholder" ), myOptions
);
    var  marker = new google.maps.Marker( {
     position: latlon,
     map: map,
     title: "You are here!" }
    );
   }

   function  showError( error ) {
    switch( error.code ) {
     case error.PERMISSION_DENIED:
      mapholder.innerHTML = "User is denied the request for Geolocation.";
      break;
     case error.POSITION_UNAVAILABLE:
      mapholder.innerHTML = "Location information is unavailable.";
      break;
     case error.TIMEOUT:
      mapholder.innerHTML = "The request to get user location timed out.";
      break;
     case error.UNKNOWN_ERROR:
      mapholder.innerHTML = "An unknown error occurred.";
      break;
    }
   }
  </script>
 </body>
</html>
```

The following program shows how to draw a direction:

direction.html

```
<!DOCTYPE HTML>
<html>
 <head>
  <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false">
  </script>
  <script>
   var  directionDisplay;
   var  directionsService = new google.maps.DirectionsService( );
   var  map;

   function  initialize( ) {
    directionsDisplay = new google.maps.DirectionsRenderer( );
    var  GF = new google.maps.LatLng( 47.92525699999999, -97.032855 );
    var  mapOptions = {
     zoom: 14,
     mapTypeId: google.maps.MapTypeId.ROADMAP,
     center: GF
    }
    map = new google.maps.Map( document.getElementById('map_canvas'), mapOptions);
    directionsDisplay.setMap( map );
   }

   function calcRoute( ) {
    var  start = document.getElementById('start').value;
    var  end   = document.getElementById('end').value;
    var  request = {
     origin: start,
     destination: end,
     travelMode: google.maps.DirectionsTravelMode.WALKING
    };
```

```
    directionsService.route( request, function( response, status ) {
     if ( status == google.maps.DirectionsStatus.OK )
      directionsDisplay.setDirections( response );
     } );
   }
  </script>
 </head>

 <body onload="initialize( )">
  <div>
   Start:
   <select id="start" onChange="calcRoute( );">
    <option value="streibel hall, und, nd">Streibel Hall</option>
    <option value="ray richards golf course, grand forks, nd">Ray Richards</option>
    <option value="ralph engelstad arena,nd">The Ralph</option>
    <option value="university park, grand forks, nd">University Park</option>
    <option value="cabela, east grand forks, mn">Cabela</option>
    <option value="city hall, grand forks, nd">City Hall</option>
    <option value="columbia mall, grand forks, nd">Columbia Mall</option>
   </select>

   End:
   <select id="end" onChange="calcRoute( );">
    <option value="ray richards golf course, grand forks, nd">Ray Richards</option>
    <option value="ralph engelstad arena,nd">The Ralph</option>
    <option value="university park, grand forks, nd">University Park</option>
    <option value="streibel hall, und, nd">Streibel Hall</option>
    <option value="cabela, east grand forks, mn">Cabela's</option>
    <option value="city hall, grand forks, nd">City Hall</option>
    <option value="columbia mall, grand forks, nd">Columbia Mall</option>
   </select>
  </div>
  <div id="map_canvas" style="width:96%;height:400px"></div>
 </body>
</html>
```

Requirements

This is an open exercise, so the requirements are loosened. This system consists of two themes: (i) location-based services and (ii) AJAX. The proposed location-based service includes the following features:

- **[Location-based service (LBS): 50% total]**
  The LBS proposed by you should provide services to the Grand Forks area, so the instructor can actually test it. The requirements of your LBS are as follows:

  - (*Ideas: 05%*) Design your own LBS. The emphases are (i) creativeness and (ii) usefulness. Location-based services are popular and useful. Some examples of LBS are finding the nearby gas stations or ethnic restaurants, comparing product prices from local stores, location-aware advertising, locating people or events, finding the best bus schedules, to name a few. Explain your LBS ideas in the report. Try not to be too ambitious at first and make sure they are doable.

  - (*Implementation: 45%*) LBSs are part of electronic or mobile commerce, so their implementation are similar. The requirements include:

    - Enter geographical data into a database before starting testing.

        - Handle location-based queries.
        - Generate results based on users' locations and stored geographical data.
        - Give recommendations including map mashups such as markers and/or directions.

- **[Asynchronous JavaScript and XML (AJAX): 35%]**
  AJAX is to make the web applications more interactive and user-friendly, and maybe speedy.

    - The emphases are (i) number of AJAX applications, (ii) interactiveness, and (iii) user-friendliness.
    - The more AJAX applications are applied, the higher score you will receive. The following rankings are usually used by the instructor:

        - Excellent if 6 or more different AJAX applications are used,
        - Very good if 5 AJAX different applications are used,
        - Good if 4 AJAX different applications are used,
        - Fair if 3 AJAX different applications are used,
        - Not good if 2 AJAX different applications are used, and
        - Poor if no or 1 AJAX application is used.

    - The AJAX applications applied have to make sense and be used effectively because some students simply included AJAX in their exercises without advantages in the past. Otherwise, they will not be counted.

- **(Instructor's requirements: 15% total)**
  Other than the above system requirements, the instructor has the following requirements:

    - (*User-friendliness: 10%*) User-friendliness will be heavily considered when grading. In the past, some exercises were awkward, which made the grading or browsing difficult. For example, it is considered not user-friendly if the system repeatedly asks users to enter their names/IDs/passwords.

    - (*Plagiarism-proof: 05%*) It is for the instructor to find any plagiarism. Each interface includes a button "Display source," which is to list ALL the source code for implementing the functions of THIS interface. Only one password is for all interfaces and all exercises. The system will be highly suspected if fail to implement this button.

Evaluations

The following features will be considered when grading:

- **Specification:**
    - The instructor (or your assumed client) has given the exercise specification as detailedly as possible. If you are confused about the specification, you should ask in advance. Study the specification very carefully. No excuses for misunderstanding or missing parts of the specification after grading.

- This is an open exercise, so the specification is not detailed. Design your own LBS and AJAX applications.

- **Grading:**
  - This exercise will not be graded if the submission methods are not met. Students take full responsibility if the website is not working.
  - The location information provided by the HTML5 Geolocation may not be as precise as the one by GPS or may not be available. In addition, *we will not do the actual road testing and desktop computers will be used during the demonstration*. Your LBS should have a backup plan for this; for example, mock-up locations can be entered and used.
  - This is an open exercise. Therefore, the grading of this exercise is subjective and is based on comparison. The instructor will try to grade the exercises as fairly as possible.
  - The grading is mainly based on the testing results. Other factors such as performance, programming styles, algorithms, and data structures will be only considered minimally. Exercises of undergraduate and graduate students are graded/compared separately.
  - Before submitting the exercise, test it comprehensively. Absolutely no extra points will be given after grading.
  - The total weight of exercises is 36% of final grade and all three exercises have the same weight, 12% each.
  - The instructor may check your system from different locations at Grand Forks and by using different ISPs (Internet Service Providers). Make sure your system works correctly for different locations and ISPs.
  - Firefox 51 or above browser will be used to grade exercises. Note that Internet Explorer, Chrome, and Firefox are not compatible. That is your exercises may work on the IE or Chrome but not Firefox.
  - The systems have to be active and the source-code printouts and reports will be kept until the end of this semester. They will be re-checked for plagiarism from time to time. The instructor will inform you the exercise evaluations by emails after grading.

- **Databases:**
  - A database has to be used and try to perform the tasks by using SQL as much as possible because SQL, a non-procedural language, can save you a great deal of programming effort.
  - The SQL DDL commands such as "`create table`" have to be submitted, where SQL is Structured Query Language and DDL is Data Definition Language.
  - From the source code submitted, the database design and programs will be examined. Poor database design or uses will result in a lower grade.
  - **(-05%)** if the database design is NOT optimal.
  - **(-05%)** if the SQL `create` commands of database implementation are NOT submitted.
  - There are many advantages of using databases. If database is not used, the problems caused by not-using-transaction must be considered. For example, if two customers are enrolled at the same time, an ID may be assigned to different customers if databases are not used.

- **Comments:**

- Make the exercise work first. Do not include irrelevant features, such as user passwords, in the beginning.
- One way to build a complex web system from scratch is to design the user interfaces first and then implement the system button by button. By doing this way, it could simplify the construction. The recommended construction steps are

  i. Database design (E-R modeling or normalization),
  ii. Database implementation (SQL),
  iii. Interface building (HTML and CSS), and
  iv. Button-by-button implementation (PHP).

- Many times, simplicity is the same as user-friendliness.
- Security concerns are mainly ignored here. For example, a customer may be able to check others' account information if protection is not enforced. Small or medium -size businesses do not usually implement their own secure payment schemes. Instead they use a third-party payment system such as PayPal or purchase software from company like Cayan and integrate it with their websites.
- The function of automatically sending emails is important for e-commerce systems, but will not be used here since companies complained our students sending out numerous mails because of faulty programs.

Slide 11.10: Server-side PHP scripts
Slide 12.2: AJAX (Asynchronous JavaScript and XML)
Home