Week 4

Web Application Architecture

We have been loading static data files (using GDownloadUrl()) into our JavaScript programs. This is obviously limited, as real world projects query large databases to select just the data that user needs at the moment.

In order to select just certain data to download to the client, our JavaScript programs must run small applications on the web server that will sift through the data to get just the data the user needs. Our JavaScript programs are not allowed to do this, we must rely on small applications (servlets) to do it for us.

Garden variety webservers (e.g. Apache) will not allow web pages like ours to run applications on them. These webservers will just serve up static data files, as they have been doing in response to our XMLHttpRequests we have been sending via GDownloadUrl().

In order to allow our web page to run an application (like a database management system) for us, the web server must have an application server installed on it. A popular open source app server is Tomcat. The small application that Tomcat runs for us is called a servlet, and it can be written in Java, php, or Python, etc. Again, these are required to selectively load data into our web page.

So a web server that is able to run a small servlet for us and call our web application back with the data we requested needs the following software:
operating system (e.g. **L**inux)
web/application server (e.g. **A**pache with Tomcat)
database management system (e.g. **M**ySQL)
interpreter for a servlet (e.g. **P**hp)
This list of software is called the "LAMP Stack", and it is already installed and running on ctislab.fhda.edu. Though there are other examples of technologies that will do the job (for example, Microsoft provides the .NET software), LAMP is the open source solution that most webservers use today.

These servlets can get somewhat complicated, and they tend to perform fairly standard services like searching through inventories or billing customers. So there are packages available that make it easy for server side programmers to create these servlets. These packages are said to be in the middle tier, and they are used to create middleware. Here are some examples:
ColdFusion, J2EE, BEA.

When designing your own application architecture, the client/server dichotomy is a useful one. For each feature of your application that you code, you can choose where to run it, on the client or on the server? If you choose to put more of your code and data on the client, you are said to have a "fat client", whereas if you choose to code more on the server, you are said to have a "thin client."
Here are some examples of specific decisions you will have to make: Where to do error checking? How much data to store on the client? In general, the more code you run on the browser, the better the user experience but the harder it is to develop. The more data you put on the server, the more reliable the data is because you don't have concurrency problems. The following diagram attempts to show this spectrum:

browser <-------HTTP-------> webserver<-->appserver<-----------DB--------->

Where to put the code?
responsiveness ease of dev

Where to put the data?
concurrency problems

Relational Databases

As discussed in the previous section, often you will need to get data from a database on the fly, getting only what you need. Only servlets can access databases, however, and they are small applications with a URL. For example, there is a database on ctislab containing all the countries, cities, languages of the world, together with a lot more information. The database is called world.sql, is downloadable for free since it is used as a sample database for the MySQL database management system. We cannot query this database directly from our JavaScript program, though, so we must call a servlet to query it for us. There is already a servlet on ctislab also, called worldquery.php, which takes a query in SQL as a parameter. (SQL is the standard language used for specifying what data you want to pull out of a relational database.)

So how do we run this servlet (and receive the response) from our JavaScript code? In order to run a servlet, you need to compose a query and send it as a parameter to the servlet in an XMLHttpRequest object (via GDownloadUrl()), and get your response in either XML or JSON. So now when we call GDownloadUrl(), we need to send it three parameters:
1) the URL of the worldquery.php servlet
2) the name of our callback function
3) a string that specifies our search criteria in SQL

Here is an example call to worldquery.php:

```
GDownloadUrl("http://ctislab.fhda.edu/ajax/worldquery.php",   // this is the URL
                    makeAnswerBox,                                  // this is the callback function
 "q=SELECT Continent FROM world.Country WHERE Name = 'France';");  // the query in SQL
```

Although understanding relational databases and SQL is not required for this course, I'd like to explain a little bit about them since you will be including queries in your programs. I also highly recommend that you study them further if necessary, since almost all software applications these days make extensive use of a database.

The database world.sql consists of three tables:
world.City, world.Country, world.Countrylanguage
These tables are connected by a column that is common to each of at least two tables for example, countrycode is a column in all three tables. This way, you can look up a country in world.Country, copy its country code, then look up that country code in world.CountryLanguages to find the languages spoken in that country. The reason that this type of database is called "relational" is because multiple tables *relate* to each other through a shared column.

I copied all the data from these three tables into text files so that you can view it without running MySql. Please see the files:
worldCity.txt, worldCountry.txt, worldCountrylanguage.txt in the following sections of this Module.

In order to form a query, you must know the names of the columns in these files. The example query:

```
q=SELECT Continent FROM world.Country WHERE Name = 'France';
```

was written only with the knowledge that the table called "world.Country" has a column called "Continent" and a column called "Name." I also needed to know that there was a row in the world.Country table where the Name column contained "France".

worldCity.txt

Instructions: Contains the data from the world.City table in the world.sql database.

worldCity.txt

worldCountry.txt

Instructions: Contains the data from the world.Country table in the world.sql database.

worldCountry.txt

worldCountryLanguage.txt

Instructions: Contains the data from the world.CountryLanguage table in the world.sql database.

worldCountryLanguage.txt

Running a servlet from JavaScript

In the following program, gLabTest.html, we call the worldquery.php servlet and send it a query in SQL. Our servlet gives the response in XML, and this program just dumps it to the document window.
As an exercise, you can modify the QUERY constant to ask other questions of the world.sql database.
However, not all queries are allowed by our servlet: they must be read only, not contain *, and not require a loop. The response will always be just a simple list in XML format.

gLabTest.html

Instructions: This program will only run from the ctislab webserver. In order to play with it, please copy it up to that server before loading it into your browser.

gLabTest-1.html