CGT 353: Principles of Interactive and Dynamic Media

Flash and Data Integration

Introduction

- The internet was designed for data transfer.
- There are several formats for transfering data
    - JSON
    - Name = value pairs
    - Simple Text
    - Proprietary
    - XML
- Flash can transfer data in a number of these methods

Terminology

- **Data source**: Any place flash can load external data from
- **Data transfer**: The act of transfering the data to and from source('loading', 'sending')
- **External data:** Data that is stored in some form outside of the SWF file, and loaded into the SWF file when needed.

- **URL-encoded variables:** The URL-encoded format provides a way to represent several variables (pairs of variable names and values) in a single string of text. Individual variables are written in the format `name=value`. Each variable (that is, each name-value pair) is separated by ampersand characters, like this: `variable1=value1&variable2=value2`. In this way, an indefinite number of variables can be sent as a single message. ('url-string')

- **Request method:** When a program such as Flash Player or a web browser sends a message (called an HTTP request) to a web server, any data being sent can be embedded in the request in one of two ways; these are the two *request methods* GET and POST.
    - On the server end, the program receiving the request will need to look in the appropriate portion of the request to find the data, so the request method used to send data from ActionScript should match the request method used to read that data on the server.
    -

Data Sources

Text Files:

- Flash can load text (.txt) files containing data formatted using the URL string format

Server Side Scripts:

- Placed on ASP, CFML, CGI, or JSP pages and executed by a server
- Can return data in a number of formats including XML and URL string formats

XML Files:

- Usually a text file with XML formatted data

XML Socket:

- Applications that run on a server and connect several simultaneous users to one another
- Flash can send or receive information via the socket using the XML format

GET vs POST:

- Two ways to transfer data from Flash to server-side scripts: GET or POST
- Also used with HTML forms
- Using GET you're simply concatenating variable name/ values into the URL

http://www.purdue.edu/login.asp?firstname=Kellen&lastname=Maicher

- This method is not very secure and also has a 1024-character limitation
- Using POST, the data is contained in the header of the HTTP request, so you can't see it transferred
- POST also doesn't have a character limit

AS 2.0 Data Methods

- **Movieclip methods** - `getURL(), loadVariables(), loadVariablesNum(), loadMovie(), and loadMovieNum().`
- **LoadVars methods** - `load(), send(), and sendAndLoad()`
- **XML methods** - XML are `XML.send()`, `XML.load()`, and `XML.sendAndLoad()`.

Details

- **getURL()** - returns any information to a browser window, not to Flash Player.
- **loadVariables()** - loads variables into a specified timeline or level in Flash Player.
- **loadMovie()** method loads a SWF file into a specified level or movie clip in Flash Player.
- **Send:** Sends data to a source
- **Load**: Loads data from a source
- **sendAndLoad**: Sends data, and loads the output from a source

Parameters

- *URL* is the file in which the remote variables reside.
- *Location* is the level or target in the SWF file that receives the variables. (The getURL() function does not take this parameter.)
- *Variables* sets the HTTP method, either GET (appends the variables to the end of the URL) or POST (sends the variables in a separate HTTP header), by which the variables are sent. When this parameter is omitted, Flash Player defaults to GET, but no variables are sent.

Load Vars Class

- Use the LoadVars class when working with data in the URL string format
- Enables you to load variable from a text file or to and from a server-side script
- Note: You cannot directly write to a text file from Flash
    - (How could you instead?)

Creating a LoadVars Object:

var KellenInfo:LoadVars = newLoadVars();

Loading variables from a URL into a LoadVars Object:

KellenInfo.load(http://www.purdue.edu/stupidtext.text);

So, if you had a LoadVars object named KellenInfo and loaded the following string from a text file:

Firstname=Kellen&lastname=Maicher&age=30

You could reference those names using:

KellenInfo.firstname

KellenInfo.lastname

KellenInfo.age

or

var userAge: Number = KellenInfo.age

- If you want to send variables in a LoadVars object to a server side script for processing, use the send() method.

myLoadVarsObject.send("http://www.purdue.edu/process.asp");

- Use the **sendAndLoad** method to get a response

myLoadVarsObject.sendAndLoad("http://www.purdue.edu/process.asp, receivingLoadVarsObject);

Properties of the LoadVars Class:

1. **contentType –** gives you the mime type specified in the HTTP header
2. **loaded –** returns a true or false value depending on whether or not the data has finished loading into the object

- The only event available to the LoadVars class is the **onLoad** method
- Used to call a function when data has finished loading into an object
- To load variables into an object then call a function when the loading is complete:

    - Define a function
    - Create a new LoadVars object
    - Specify the function to be called when loading complete
    - Invoke the load()method of the LoadVars object

```
function myFunction():Void{
    trace("Data is loaded");
}
var container:LoadVars = new LoadVars(); //creates the LoadVars object
container.onLoad = myFunction;  //calls the function(myFunction) when data is loaded
container.load("http://www.purdue.edu/myfile.asp");
```

Actionscript 3.0 Data Methods

Overview

- Use URLLoader to load data
- Typecast data to specified type
    - Use URLVariables class for URLString data
- Handle data as needed

Example – External Scripts

```
var variables:URLVariables = new URLVariables("name=Franklin");
var request:URLRequest = new URLRequest();
request.url = "http://www.[yourdomain].com/greeting.cfm";
request.method = URLRequestMethod.POST;
request.data = variables;
var loader:URLLoader = new URLLoader();
loader.dataFormat = URLLoaderDataFormat.VARIABLES;
loader.addEventListener(Event.COMPLETE, completeHandler);
try
{
    loader.load(request);
}
catch (error:Error)
{
    trace("Unable to load URL");
}

function completeHandler(event:Event):void
{
    trace(event.target.data.welcomeMessage);
}
```

Shared Objects

Shared Objects:

- Shared objects save data to a client machine
- With traditional Web applications, this is done via **cookies**
- With Flash, we use **shared objects**
- Use the **SharedObject** class

Creating Shared Objects:

- Process for creating a new shared object and opening an existing shared object is the same
- getLocal() opens a local shared object of the specified name if it exists

var lsoPreferences:SharedObject = SharedObject.getLocal("userpreferences");

- File extension for shared object files is **.lso,** but don't include that in the parameter
- Flash either finds the existing file or creates a new one

Setting Values in Shared Objects:

var lsoPreferences:SharedObject = SharedObject.getLocal("userpreferences");

lsopreferences.data.backgroundColor = "red";

lsoPreferences.data.name = "A Reader";

- The **data** property is an object in itself which you can assign additional properties
- Properties assigned to the SharedObject will not be saved, only the properties of the data object

Saving the Shared Object to the Client

- Flash automatically tries to save the shared object when the SharedObject instance is deleted
  - When player closes
  - When movie is closed or replaced in the player
  - When object is deleted with the delete statement
  - When the object is garbage-collected after it falls out of scope
- Don't rely on automatic saves however, use the flush (method)

AS 3.0 Shared Object Example:

```
var so:SharedObject = SharedObject.getLocal("userHighScore");
 so.data.highScore = new Number(1234567890);
 so.flush();
```

Other Data Sources

Actionscript 3.0's flash.net package also contains classes for other types of remote communication:

- **FileReference** class for uploading and downloading files from a server
- **Socket and XMLSocket classes** for communicating directly with remote computers over socket connections

- **NetConnection and NetStream classes**, used for communicating with Flash-specific server resources (such as Flash Media Server and Flash Remoting servers) as well as for loading video files.

Last note:

AMFPHP: Flash remoting server for PHP. Great when handling complex data transfer between Flash and PHP