# Website Architecture

Project Specification

## Project 2

The specification for Project 2, which tests proficiency with server-side dynamism, including Apache, PHP, HTML forms, and sessions.

**Michael Serritella**

Summer 2010

# Website Architecture

Project Specification | Project 2

## Project Goals

This project tests your proficiency in PHP and Apache, specifically with handling HTML forms and managing sessions. You don't have to design a site's content or organization; that has already been done. Nor do you have to write any of the HTML or PHP which frames the site. You should only have to write the minimum code necessary to exercise the core skills for this project.

You will accept HTML form data in the form of text and uploaded files. You will use PHP sessions to group a series of form submissions into a conceptually unified structure. You will provide a search function which very basically searches the data submitted during the current session. You will use Apache to assist in the creation of intuitive URIs for the search function.

## General Description

Basically, you will be making a crude copy of a popular file dropbox service, which is to say you will be making a competitive alternative. We can call it an Interent Putbox.

The user will submit files and provide some metadata, and you will keep track of all the files the user has submitted in the current session (i.e. until the cookie is cleared). You will validate the inputs of that form. You will provide a page which lists the files submitted during the session, along with their metadata. You will provide a page which takes a search string and searches the metadata of the files submitted during the session. You will provide a page which clears the session but does not necessarily clear the data from disk.

A shell for each of the pages has been provided, written in PHP and using the MiniArc library, the primary component of which is the HTPPS. You do not need to worry about the aesthetics of the site, the validity of the markup, or subtle browser-compatibility issues. You may modify all of the provided code, as long as you still use PHP and Apache and satisfy all of the other requirements of this document. You do not need to use MiniArc (which includes the HTPPS) in any of the code that you write, and you may even deconstruct the code given. However, it should make your life much easier.

The project will be graded using Mozilla Firefox 3.0 or newer, for Windows or Linux. Your submission should not differ significantly when viewed in any of these browsers; however, if it does, tell the instructor, so that the most favorable version may be used.

## Provided Materials

All provided materials are on the course website in a file called "CIS4930_WAM_2010Su_Project2_Provided.zip". This includes:

- The `public_html` folder, which should be the site's document root. This contains scripts and files that may be requested via `http://localhost`.
- An `includes` folder, which is a sibling to `public_html`.
- A `MiniArc` folder, which is a child of `includes`. `MiniArc` contains the MiniArc distribution, along with a derived/customized pretty printer that writes the pages of this site (e.g. site logo, header, footer). This modified pretty printer is in `MiniArc/Specialized`.
- Site-specific PHP includes inside of `includes`, which provide classes for business objects and functions which use the pretty printer to write the site's pages.

Directly within `includes` are perhaps the most interesting components for this project. The most unique thing in there is the `PutboxFileMetadata` class, which is partially defined. It is a guideline which you may want to fully define. The properties of this class correspond to the form data that you must collect, and the methods will help you in your requirements. See the class definition in the `InternetPutbox_FileMetadata.inc` file for explanatory comments.

## Requirements

This section details the components of your project along with their requirements, followed by general requirements which apply to the whole project. Each of the specific sections corresponds to one page of the site.

### Submission form

These requirements apply to the file submission form.

#### Form elements

The form shall collect the following information, including validation:

**File** The file itself, which must have a size less than or equal to 1 MB (1,048,576 bytes). The file's name must be kept, as it must be provided to the user later under the same name. You can assume that no two files will coexist with the same name.

**Title** A title for the file, which may or may not be its filename. It may contain only alphanumeric characters, spaces, and English punctuation found on an American English keyboard. It should have no leading or trailing spaces.

**Author** The file's author. It may only contain alphanumeric characters and spaces.

**Rating** A preference score within [1,5], like a product review rating. Must be stored as an integer.

**Purpose** A choice within {Business, School, Personal}. You can encode this any way you want and represent it however you want; you only need to be able to accurately reproduce the user's choice and ensure that one of these is chosen.

**Timestamp** A timestamp that is generated upon submission, which does not come from user input. The user must not be able to influence this.

You may use whichever form elements you want; you design the form.

The file shall be placed in a directory specific to the current session. After the session is over, the files may remain.

**Pro tip:** See the validation functions in MiniArc. See the PHP **documentation** on file uploads, which is further discussed in Lezione 10.

## File list

This is a list of all files uploaded in the current session. It may present the files in any order. It must reproduce all data stored by the submission form, including links to download the files.

You will probably want to use the HTPPS and the `ToMarkup()` method of the `PutboxFileMetadata` class to generate these displays. You don't need to, and it would be approximately the same amount of work (or maybe even less) to not use the HTPPS. But your solution should probably be reusable in any case, and you may want practice with the HTPPS, since you'll be using it for the next ten or twenty years.

The files must be downloadable under their original filenames, though they may be in whatever folder you want.

## Search

This is a search of the metadata of files uploaded in the current session. The search shall look for matches in the file's title and author fields, and it shall at least support a case-sensitive, full match with the search query (i.e. the easiest case; this is not about search algorithms). This would receive full credit. If you want, you can make it case-insensitive, and you may also search other fields, like "business"/"Business" for the file's purpose.

The search page has two interfaces; they must both lead to the same results page (i.e. the same script must execute). The interfaces are:

1. The search box which is on every page.
2. The URI pattern "/Search/<query>", where "<query>" is the search query. You may assume that this contains no spaces. The "/Search" is relative to the document root of the website, which looks something like `public_html/Search` in the filesystem. There may be an actual `Search` directory, but there should not be files within `Search` that correspond to search terms.

The search results page must render the metadata of any matching files in the same way that they appear in the list of all files.

## Clear

This is a page which clears the current session and starts a new one. There is a link to this page from every page in the site. Once clicking on any of those links, the session will be cleared and the user will be redirected to the previous page, without the user having to do anything else. When the user returns to the page from which he/she clicked the "Clear" link, the page shall look as if the history is cleared; it shall not show any evidence of the prior session.

You are not required to clear the actual files from disk. In fact, files from old sessions should be able to coexist with files from current sessions, as this is realistic (they are usually "garbage collected" imprecisely, perhaps after a conservative time has passed). There is no such requirement for points, but you should think of it this way.

## General requirements

These requirements apply to the project overall.

### Value-added content

For fun or education, any benign content may be added to that which is required - except that which is required to not exist - and it would not incur a penalty.

## Submission

The project must be contained within a folder called "Project2_*[Surname]*", where *[Surname]* is your surname, using only alphabetic characters, without spaces, with the first letter of each word in your surname capitalized within the representation. That folder must be contained within a .zip file, such that the folder is the top-level element in the zip file. The zip file must be named "Project2_*[Surname]*.zip".

In Linux, you can achieve this by navigating to the parent of your project directory and executing the command:

```
zip -r Project2_[Surname].zip Project2_[Surname]
```