# http://www.ahphosting.net/images/ColdFusion-9.jpgIntroduction to Adobe ColdFusion

Adobe ColdFusion is a rapid web application development platform. The programming language is called ColdFusion Markup Language (CFML) which uses a tag syntax that closely resembles HTML and also allows for embedded scripts that can be written in a JavaScript-like language known as CFScript. It provides convenient methods for creating data-driven websites. A typical ColdFusion project is a combination of web-based and server-side technologies (HTML, CSS, JavaScript, ColdFusion, SQL, etc.):

**Server Side**

**Database**

**ColdFusion**

**Webserver**

**FTP Site**

**Webservice**

**Client Side**

**Web Browser**

**HTML**

**CSS**

**JavaScript**

#### ColdFusion Markup Language

ColdFusion Markup Language (CFML) is a tag-based language, like HTML, that uses special tags and functions. With CFML, you can enhance standard HTML files with database commands, conditional operators, high-level formatting functions, and other elements to rapidly produce web applications that are easy to maintain. However, CFML is not limited to enhancing HTML. For example, you can use CFML to create web services for use by other applications.

**The ColdFusion scripting environment**

The ColdFusion scripting environment provides an efficient development model for Internet applications. At the heart of the ColdFusion scripting environment is the ColdFusion Markup Language (CFML), a tag-based programming language that encapsulates many of the low-level details of web programming in high-level tags and functions.

# ColdFusion Workshop

A basic understanding of how a webpage works is required before beginning a ColdFusion project. This workshop will guide you through the steps needed to collect data from a user and submit that data to a CFML page for processing, display data from a database table, authenticate a LDAP user, send an email, and use JavaScript within a project. You will also learn how to organize your code using the Model/View/Controller (MVC) architecture. All data-driven projects intended for users should be coded within this architecture.

## Lesson 1 – Hello World

Subjects covered:

* Creating a ColdFusion/HTML source code file
* Setting a ColdFusion variable
  + Variable scopes
* Output the variable within HTML code
  + The hash (#) sign within the cfOutput tag

Steps:

1. Open an FTP connection
   1. Create a directory within your directory for this lesson
2. Create a file
   1. Open the file and add the base HTML tags that make up a webpage (html, head, title, and body).
   2. Add a cfSet tag at the top of the file.
      1. <cfSet local.text = “Hello World” />
         1. <cfSet {scope}.{name} = {text or other CF variable} />
   3. Add cfOutput opening and closing tags within the HTML body tags.
      1. <cfOutput>#local.text#</cfOutput>
         1. Hash tags (#) are required to indicate that CF is outputting a variable.
3. Save the file and open it in your browser.

Extra Credit:

* Modify your file so that the local.text variable is “World” but “Hello World” still gets output to the browser.

Notes:

## Lesson 2 – Form Submission

Subjects covered:

* HTML forms
* Outputing a form input in ColdFusion
* Conditional ColdFusion statements (cfIf)
* ColdFusion error messages

Steps:

1. Create a new file
   1. Open the file and add the base HTML tags that make up a webpage (html, head, title, and body).
   2. Add a HTML form element within the body tag.
      1. The method should be post
      2. The action should point to the current page
   3. Add a text input box and a submit button.
2. Save the file and test it in your browser.
3. To correct the error, use the cfIf tag to check for the existence of a variable
   1. Add a cfIf tag to check for the existence of form.text
      1. “form” is the variable scope assigned to HTML objects defined within the submitting HTML form element.
      2. <cfIf isDefined(“{scope}.{name}”)>  
         </cfIf>
      3. Output the variable within the cfIf statement using the cfOutput tag.
4. Save the file and test it in your browser.

Extra Credit:

* Change the form so that it submits to a new page that outputs the submitted form elements.

Notes:

## Lesson 3 – Sending Email

Subjects covered:

* Multiple field HTML form submission
* Validating user input
  + JavaScript
  + ColdFusion
* Sending email through CFML (cfMail tag)
  + HTML mail
  + Text Mail
  + cfMailPart
  + Attachments
* File Uploads
* CFML Custom Tags

Steps:

1. Modify the previous form to include three text boxes (to, from and subject) and a text box (email body).
2. Using cfIf statements, validate the user input on the ColdFusion page that the form submits to.
3. Change the submitted to ColdFusion page from the previous lesson so that it uses the cfMail to send an email.
   * Use the ColdFusion documentation to look up the cfMail tag <http://help.adobe.com/en_US/ColdFusion/9.0/CFMLRef/index.html>
   * Send a plain text email
   * Send a HTML email with some HTML formatting
   * Using cfMailPart, send an email that contains both plain text and HTML
4. Change the form submission page to use JavaScript to validate the user input
   * This uses a CFML custom tag and custom attributes within HTML form elements. See example.

Extra Credit:

* Change the project so that a user can upload a file and send the file as an email attachment.
  + Add a file upload HTML element to the form.
  + Use the ColdFusion documentation to learn about sending an attachment through the cfMail tag.

Notes:

## Lesson 4 – The MVC Framework

Subjects covered:

1. Model/View/Controller concepts
2. Database connections
3. Querying and displaying data
   * cfQuery, cfOutput

Steps:

1. Copy the mvc directory from /snyderb/training/ to your user directory
   1. This is a blank copy of the framework
   2. The view directory should contain any of the files that contain HTML and are what the user sees.
   3. The model directory should contain all of the query logic.
   4. The controller directory should contain all of the functional logic
2. Review the sb\_template and sb\_settings files in the root project directory.
3. Create files in the model and view directories of the same name.
4. Add the query logic to the file in the model directory.
   1. The table schema and name are “gvadc” and “training\_user”
   2. The cfQuery tag needs the following attributes
      1. datasource=”DS\_APPCFGVADC”
      2. name=”{descriptive query name prefixed with qry}”
   3. Write a SQL query within the cfQuery opening and closing tags that returns the following columns:
      1. firstName
      2. lastName
      3. email
      4. entryDate
5. Add HTML code to the file in the view directory to display the results of the query in a table:
   1. Use the cfOutput tag with the query attribute set to the name of the query you defined in the model page
      1. This will loop through all rows returned by the query.

Extra Credit:

* Use the ColdFusion documentation to figure out how to format a date value.

Notes:

## Lesson 5 – Two Stage Authentication (LDAP)

Subjects covered:

1. LDAP Authentication
   * cf\_ldap\_auth
2. Verifying authenticated users against a database table
   * cfQuery, cfOutput
3. Clearing variables via the structClear command

Steps:

1. Helpful Hint - Redirects are accomplished by setting the variable sb.useRedirect to a page in the project.
2. Add an index.cfm file to your MVC projects view directory.
   1. Add a form that accepts a username & password.
3. Add a page in your controller that accepts the form post and validates the user credentials.
   1. Use the cf\_ldap\_auth CFML custom tag which takes two attributes “username” and “password”.
      1. To authenticate student LDAP accounts use the cf\_ldap\_authst CMFL custom tag.
   2. If the cf\_ldap\_auth tag successfully authenticates the user, a variable named “ldap\_auth” will be set to “1”.
      1. If authentication is successful, set a variable in the session scope named “session.user.username” then redirect the user to the database query page from the previous lesson.
      2. If authentication is not successful, redirect the user back to the login page and display an error message.
   3. Add a query to validate an authenticated username against the gvadc.training\_user table.
      1. If the user exists in the table, set a variable in the session scope named “session.user.username” then redirect the user to the database query page from the previous lesson.
      2. If the user does not exist in the table, redirect the user back to the login page and display an error message.
4. Add a page in your controller that logs out a user when called.
   1. Call the structClear command on the session variable either with a cfSet tag or a cfScript tag.

Extra Credit:

* Create a CSS file and add some style to your project.

Notes:

## Lesson 6 – Using CAS for Authentication

Subjects covered:

1. CAS (single-sign on) Authentication
   * cf\_single\_sign\_on CFML custom tag
   * CAS also performs a single sign-out function as well.

Steps:

1. Modify the index.cfm file and add the single sign on CFML custom tag at the top of the file (<cf\_single\_sign\_on>).
   1. Note that the rest of the file is no longer used and can be cleared out.
2. Add a page in your view that will be displayed when a user is logged out.
3. Change the page in your controller that logs a user off to call the cf\_single\_sign\_on CFML custom tag.
   1. <cf\_single\_sign\_on action=”logout” page=”{the page you created in step 2}”>
4. If your project uses two stage authentication, you will need to modify your login post page.
   1. Remove any cf\_ldap\_auth calls
   2. Add a conditional statement that checks for the existence of a variable set after the page is initially called.
   3. Ensure that the proper redirects are at the bottom of the page.
5. Modify the sb\_settings.cfm file in your projects root directory.
   1. sb.defaultSessionPath = “#sb.func.url({name of the page in your controller that handles the login post})#”
   2. sb.noSessionActions = “index,{name of the page to display once a user logs out}”

Notes:

## Lesson 7 – Concepts for larger projects and time for questions

Subjects covered:

1. MVC framework “circuits”
2. Code organization and reuse
3. ColdFusion Components (CFCs) and AJAX
4. Project look and feel – HTML, CSS, and JavaScript frameworks.

Notes: