Server-Side Scripting Languages as Tools for Collecting and Organizing Data

作者：**Xiaoqing He Dan Cabirac**

Office of Information Technology
University of Maryland
College Park, Maryland 20742

xhe@glue.umd.edu
dcabirac@glue.umd.edu

Abstract

Server-side scripting languages can provide educational institutions with the ability to create web-based applications, which efficiently collect data in an well-organized manner. This presentation covers Cold Fusion Markup Language (CFML) and Hypertext Preprocessor (PHP), two popular scripting languages, which interact with databases. These languages provide an efficient and flexible means of building on-line assessment tools, questionnaires, and surveys. The data collected is automatically inserted into databases, and may also be sent as email. Several projects will be discussed, ranging from a simple web-based form, which sends data to a database, to a complex system, which not only collects data, but also allows authorized users to view the data online and to add comments which will be archived.

Introduction

The development of Internet and WWW presents us with a unique opportunity for on-line distance teaching and learning. This field has developed very rapidly in the past few years because of the many practical uses of the web. Among these practical uses is the ability of web-based fill-in forms to efficiently gather data that can then be inserted into databases for later retrieval. Server-side scripting languages are one of the most effective ways to collect data on-line. They can also be used to query a database to retrieve data and dynamically create web pages. This paper will provide some background into the development of these languages, describe how they function, and present some case studies showing real life uses of this technology at a large American university.

First we need to examine some of the basic features of the web to see how these scripting languages developed. A major feature of the WWW is the potential for developers to create links between text and other media, not only within an individual document but also between documents residing on any computers in the world which have access to the Web.

Web documents are usually written in HTML (Hypertext Markup Language), a simple markup language that describes the logical structure of the text. Very often, HTML is stored in static files on the server's hard disk. Such simple, static hypertext documents can convey a great deal of information, but eventually their limitations become clear. For example, what if the author wishes to provide dynamic information, which could be based on the user inputs, interaction with the database etc.? Dynamic documents, unlike static documents, require the server to generate the documents on the fly. For example, dynamic documents can be generated from databases of all kinds, from video capture systems, and from scientific instruments such as various real-time monitoring systems. Such documents are often transmitted directly to the clients as they are created, without even being stored in the file system. In other cases, they may consist largely of fixed content, with a small amount of dynamic content generated when a page is actually delivered.

One approach to serving a dynamic Web page is through the "Server-side include" technique. The HTML system makes it easy to link documents to each other. However, the current versions of HTML do not allow a single large HTML document to be assembled from several smaller documents through simple reference in the documents unless it physically contains copies of the these smaller documents. But nothing prevents Web servers from providing their own extended version of HTML in which such included sections are permitted, as long as

the server takes care of the task of including these documents and delivers the complete HTML documents to the client. This approach is called the server-side include mechanism. A server side script language extends the server side include interface normally used to insert entire documents and attempts a more natural feel as an "extension" to HTML.

There are many server-side scripting languages available. Here we will concentrate on Cold Fusion Markup Language (CFML) because of its wide usage. We will also introduce the Hypertext Preprocessor
(PHP) language because it is open source and free to everyone. Our case study will be based on CFML.

Cold Fusion (CF) and Hypertext Preprocesser (PHP)

Cold Fusion is a scripting language that is used to write dynamic web pages. It lets you create pages on the fly that differ depending on user input, database lookups, time of day or other criteria one can define. Cold Fusion pages consist of standard HTML tags such as <FONT SIZE="+2"> together with CFML (Cold Fusion Markup Language) tags such as <CFQUERY>, <CFIF> and <CFLOOP>. Cold Fusion was introduced by Allaire in 1996 and is currently in version 4.5.1.

Cold Fusion is a complete Web application server for developing and delivering scaleable online applications which allow one able to do:

- Rapid Development - Intuitive visual tools and an innovative tag-based programming environment make Cold Fusion a highly productive platform for delivering applications.
- Scaleable Deployment - A high performance, multithreaded architecture and advanced features such as just-in-time compiling, load balancing, and fail-over ensure that your applications will scale to handle the most demanding sites.
- Open Integration - Open integration with databases, email, directories, Java, XML, and enterprise systems means you can develop complex Web applications quickly and easily.
- Complete Security - The latest advanced Internet security technologies and clean integration with network and Web server security, give you the services to build secure systems.

A Cold Fusion application is very simply a collection of pages, similar to a static Web site. But unlike the pages in a static Web site, the pages in a Cold Fusion application include the server-side Cold Fusion Markup Language in addition to HTML. CFML gives you the ability to control the behavior of your applications, integrate a wide range of server technologies, and dynamically generate the content that is returned to the Web browser.

The diagram below shows what happens when a Web browser requests a page in a Cold Fusion application.

1. When a user requests a page in a Cold Fusion application by submitting a form or clicking a hyperlink, the user担 Web browser sends an HTTP request to the Web server via the Internet or Intranet.
2. The Web server passes the data submitted by the client and the requested page to the Cold Fusion Application Server either through a server API or CGI. Cold Fusion pages are automatically compiled and cached in memory so processing in is very fast and scaleable even under high loads.
3. Cold Fusion reads the data from the client and processes the CFML used in the page. Based on the CFML the Cold Fusion Application Server executes the application logic and interacts with a wide range of server technologies.
4. Cold Fusion dynamically generates an HTML page and returns it to the Web server.
5. The Web server then passes the page back to the user担 Web browser.

Similar to CF, PHP (recursive acronym for PHP: Hypertext Preprocessor) is another server-side script language to generate dynamic Web page and interact with various other resources on the Web server side. One unique features of PHP is that it is open-source software (freely downloadable from php.net and zend.com) and so enjoys the support of a large group of open-source developers. The code is continuously updated with improvements and language extensions to expand PHP's capabilities.

Cold Fusion Case Study

College: the College of Agriculture and Nature Resources (AGNR) at the University of Maryland.
Product: Cold Fusion 3.5 (will be upgraded to version 4.5.1).
Hardware: Gateway2000, Pentium II, 512K RAM, 2x9 GB hard disk.
Web Server: Microsoft IIS 3.0
OS: Windows NT 4.0
Database: Microsoft Access97

- Dynamic Calendar

AGNR has developed a calendar application in which event data can be dynamically added, viewed and extracted from an on-line calendar system. Users with proper authorization can use the "Event Submission Form" to post events. The calendar is dynamically generated each time a user queries the calendar database, but this is transparent to the user.

The primary purpose of this calendar was to allow users who have little or no technical background to easily post events to our on-line calendar.

The following graph and code are the example of how Cold Fusion (CF) can be used to query and list events from the online calendar event database:

events.cfm

```
<HTML>
<HEAD>

<TITLE>AGNR List of Upcoming Events</TITLE>
…
<CFQUERY NAME="SearchMonth" DATASOURCE="AGNR Calendar">
SELECT Month(StartDate) AS Month, Year(StartDate) AS Year, GenLocation
FROM Events
WHERE Internal = 0
ORDER BY StartDate
</CFQUERY>

<CFQUERY NAME="SearchCategory" DATASOURCE="AGNR Calendar">
SELECT Category
FROM Events
WHERE Internal = 0
ORDER BY Category
</CFQUERY>
…

<INPUT TYPE="button" VALUE="Display" onClick="searchEvents(document.search.month.options
[document.search.month.selectedIndex].value, document.search.category.options[document.search.category.selectedIndex].value,
```

```
document.search.location.options[document.search.location.selectedIndex].value, document.search.facility.options
[document.search.facility.selectedIndex].value)">
…
<CFOUTPUT QUERY="EventList" GROUP="Month">
<H3>#DateFormat(StartDate, "mmmm yyyy")#</H3>
<BLOCKQUOTE>
<CFOUTPUT>
<CFIF EndDate NEQ "">
…
<DT><B>#DateFormat(StartDate, "mmmm d, yyyy")# #thru# #DateFormat(EndDate, "mmmm d, yyyy")#</B>
<DD><B>#Link#</B> - #Description#
#WhatTime#
<DD><I>Location:</I> #Location#
<DD><I>Contact:</I> <A HREF="mailto:#POCEmail#">#POCName#</A> #POCPhone#
</DL>
</CFOUTPUT>
</BLOCKQUOTE>
</CFOUTPUT>
…
</body>
</html>
```

http://www.agnr.umd.edu/agnrcalendar

- Convert Archived Newsletters to Web page directly using CF

The University of Maryland担 Agricultural Extension Service has many old newsletters that contain information valuable to agricultural workers, students, and researchers. This on-line application was developed to convert these old newsletters which were only available in paper format into on-line HTML documents. Workers with no knowledge of HTML, and with no experience in uploading files onto a web server are able to simply type the text of the paper into a fill-in form in order to put these old newsletters on-line. After the worker keys in the newsletter, and submits it, the newsletter is archived in an Access database. When the user retrieves a newsletter, it is dynamically generated from the database. This application demonstrates that using CF you can very easily convert old newsletters to HTML format and put all the newsletters online for easy access. The user can also perform a keyword search of the newsletter database to find a particular article.

The following screenshot and CFML code show how the University of Maryland担 College of Agriculture uses CF to convert old newsletters to HTML format. The application shown also let users perform keyword searches on the archive. (See keyword cow's search results below)

```
<HTML>
<HEAD>
<TITLE>AGNR Newsletters</TITLE>
```

```
<CFQUERY NAME="Newsletter" DATASOURCE="AGNR Newsletters">
SELECT *
FROM NL_Main
WHERE ID = #url.ID#
</CFQUERY>

<CFQUERY NAME="Issue" DATASOURCE="AGNR Newsletters" MAXROWS=1>
SELECT ID, Date, MainID
FROM NL_Issues
WHERE MainID = '#url.ID#'
ORDER BY Released DESC;
</CFQUERY>

</HEAD>
...
```

http://www.agnr.umd.edu/testing/mark/newsletters

- Online News Briefs

Here CF is used to generate on-line news briefs dynamically from an on-line news database. The user is able to ask the web server to search the database and return a subset of the news briefs.

```
<HTML>
<HEAD>

<TITLE>News Release Archive</TITLE>

<CFQUERY NAME="NRList" DATASOURCE="AGNRnews">
SELECT ID, Title, IssueDate
FROM Releases
WHERE IssueDate > (Now()-90)
ORDER BY IssueDate DESC;
</CFQUERY>

<H3>Most Recent AGNR News...</H3>
...
<CFOUTPUT QUERY="NRList">
<TR>
<TD VALIGN="top">#DateFormat(IssueDate)#</TD>
<TD VALIGN="top"><A HREF="NewsRelease.cfm?id=#ID#">#Title#</A></TD>
</CFOUTPUT>
...

</body>
</html>
```

http://www.agnr.umd.edu/AGNRNews/Releases/Index.cfm

The newsrelease CF application server currently uses password protection for the security which in our case study is implemented using the campus-wide LDAP server for authentication and service authorization. More security tools will be considered such as implementing SSL with the Web/CF server.

Summary

In this paper, we introduced two popular server-side scripting languages, Cold Fusion Markup Language (CFML) and Hypertext Preprocessor (PHP) which provide an efficient and flexible means of building dynamic Web documents. Our case studies based on CFML show that powerful server-side scripting language allows us to build on-line applications that can collect information, and generate dynamic Web pages based on user inputs or other constantly changing data.

Acknowledgements

We would like to thank Mark Shute for providing access to his CF codes and Web site. The authors also acknowledge the courtesy of Mark Shute and Koralleen D. Stavish who supplied background information about related to the case study.

Reference

http://www.allaire.com/
http://www.php.net
http://www.zend.com/