



Developing WWW Applications Using Cold Fusion™

By: Rachel Campbell and Linda Cingel
February 12, 1998

What is a WWW Application?

- n It is a tool for performing a task, used through a World Wide Web browser
- n It should be distinguished from:
 - Static informational web pages
 - Desktop applications, which must be installed
- n Example: Acronym Application

Why WWW Applications-- and When?

- n Eliminates the need for installation support
- n Easily provides access to a distributed user base
- n Cross-platform delivery with fewer cross-platform programming problems
- n Works well with *small applications* that require *little client-side processing*.

Why Use Cold Fusion?

- n Easy to learn--even for “non-programmers”
- n Supports quick, easy “rapid application development” of WWW applications

BUT...

- n You should Know When to Say NO!
 - Large applications become unmanageable, sometimes unusable, and difficult to maintain.
 - Many applications *should* still be desktop applications!

Goals For This Presentation:

- n Provide general understanding of Cold Fusion's architecture, development process and capabilities
- n Provide a specific tutorial for developing a basic Cold Fusion application
- n Provide additional resources and direction for moving on into more complex development

Agenda

- n Cold Fusion Application Architecture
- n What You Need to Get Started
- n Getting Started With Cold Fusion
- n Getting Down To Basics
- n Tips and Techniques
- n Advanced Features
- n Additional Resources

Cold Fusion Application Architecture



Cold Fusion Application Components

n Server Applications:

- HTTP Server
- Cold Fusion Server Application

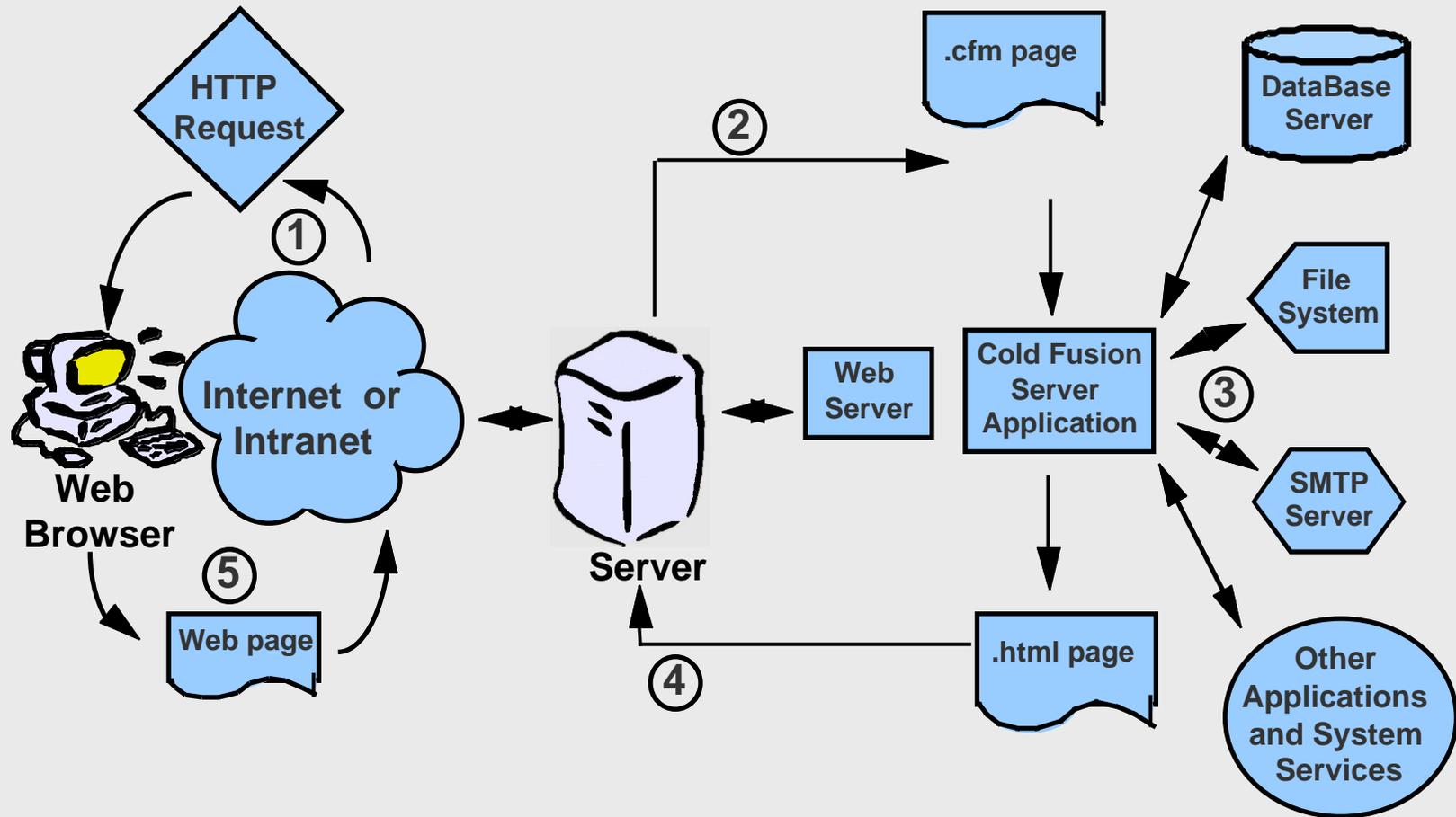
n Your Code:

- .cfm Application Pages (Example 1)

n Data:

- ODBC Data Source(s) and Database(s)
- (Optional) Other sources of data (File System, SMTP Server, COM objects, etc.)

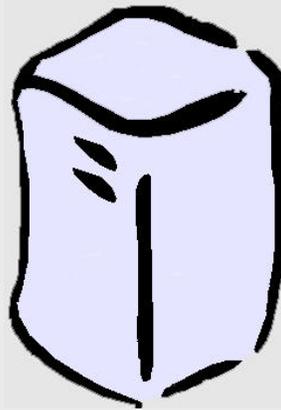
How they Work Together



What You Need to Get Started



Developer Software

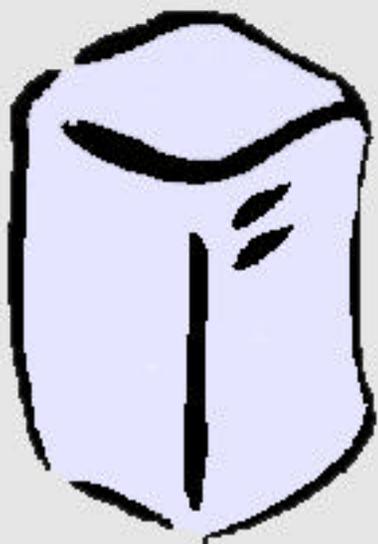


Server
Software



Client Software

Server Software:



- n Microsoft NT Server/Solaris
- n Cold Fusion Application
- n HTTP Server (e.g., MS Information Server)
- n RDBMS (e.g., MS SQL Server)

Developer Software:



- n Cold Fusion Studio, or other HTML editor
- n Database Client software (e.g., SQL Enterprise Manager, or at least MS Query)
- n Note: Some Developer software is included w/server software.

Client Software:

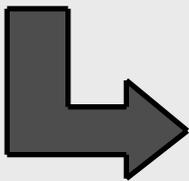


- n WWW Browser (e.g., Netscape Navigator, MS Internet Explorer)
- n Recommend Testing on all Supported Browsers
- n Note: Cold Fusion works with any browser... it simply delivers standard HTML.

Getting Started With Cold Fusion

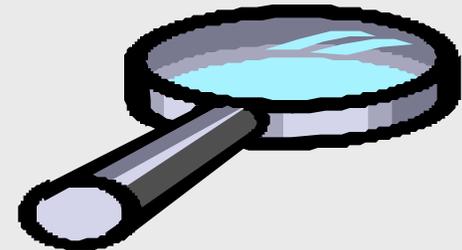


Step 1: Design

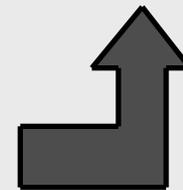


Step 2:

Build Application Pages



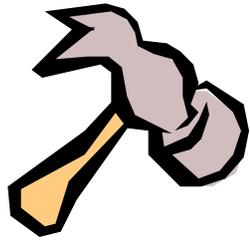
Step 3: Debug





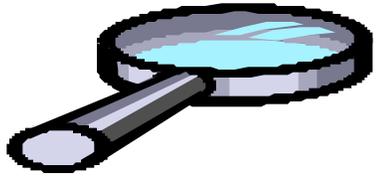
Step 1: Design

- n Employ a Rapid Application Development mindset:
- n Plan the Information Design As Well As you can...
 - Task Analysis (Example 2)
 - Naming Conventions (Example 3)
- n Then Quickly Move On, Relying on an Iterative Process



Step 2: Build Application Pages

- n Elements of a Cold Fusion Application Page
 - HTML, Cold Fusion, JavaScript, and more...
(Example 4)
 - Remember, Cold Fusion can be used with most other WWW technologies
- n Design the Interface
 - Use CF Studio, or a WYSIWYG Editor
- n Make it Live
 - Add Cold Fusion to the Page



Step 3: Debug

- n Database: Is Your Query Correct?
- n CF Syntax/Logic: Is Your Cold Fusion Syntax Correct?
- n Other Technologies: What About JavaScript?

Getting Down to Basics:

- n Selecting/Displaying Information
- n Updating Information
- n Creating Dynamic Pages
- n Using CF Functions and Expressions
- n Creating Reports

Selecting/Displaying Information

- n Assume infrastructure is complete (i.e., software, database, ODBC data source, permissions all in place and correct)
- n Create a basic Cold Fusion template page:
 - Define a Query
 - Define Results Display

Define a Query

n Using a <CFQUERY> tag

Basic Syntax:

```
<CFQUERY NAME="query_name" DATASOURCE="ds_name"  
  USERNAME="username" PASSWORD="password">  
  SQL statements  
</CFQUERY>
```

n Using stored procedures

Basic Syntax:

```
<CFQUERY NAME="query_name" DATASOURCE="ds_name">  
  { call procedure-  
    name([parameter][,parameter]...) }  
</CFQUERY>
```

Define how the Query's results will be displayed:

n <CFOUTPUT> tag provides the most flexible way to display query results

n Basic Syntax:

```
<CFOUTPUT QUERY="queryname" MAXROWS=n >  
    Literal text, HTML tags, and  
    dynamic field references (e.g.  
    #FieldName#)  
</CFOUTPUT>
```

n (Demo 1)

n (FYI) Other CF display features include CFTABLE, CFCONTENT, and CFFORM (See Documentation)

Updating Information

- n Inserts, Updates, and Deletes require two .cfm template pages:
- n A <FORM> page accepts user input as to the changes required
- n An Action page performs the requested action.

Create a <FORM> to accept user input

- n Use basic HTML or a dynamic <FORM> for users to provide information for the action

- n **Basic Syntax:**

```
<FORM ACTION="ActionPage.cfm" METHOD="post">
```

```
    Data Entry: <INPUT TYPE="text" NAME="DataEntry">
```

```
    <INPUT TYPE="Submit" VALUE="Enter Information">
```

```
</FORM>
```

- n (Demo 2, 3)

The <FORM> Action calls the Action page

- n An Action Page is a .cfm template containing a QUERY to handle the user input.
- n Cold Fusion provides two ways to handle database updates:
 - Simplified queries using <CFINSERT> and <CFUPDATE>
 - User-defined queries using <CFQUERY>

Updating Using <CFINSERT> and <CFUPDATE>

n These Tags allow for simple queries, without qualifier statements

n <CFINSERT> allows for simple insert statements

Basic Syntax:

```
<CFINSERT DATASOURCE="ds_name"  
  TABLENAME="tbl_name" TABLEOWNER="owner"  
  FORMFIELDS="form_field1, formfield2, ...">
```

n <CFUPDATE> allows for simple update statements

Basic Syntax:

```
<CFUPDATE DATASOURCE="ds_name"  
  TABLENAME="tbl_name" TABLEOWNER="owner"  
  FORMFIELDS="form_field1, formfield2, ...">
```

n (Demo 4)

Updating using <CFQUERY>

n <CFQUERY> allows you to create more complex queries (e.g., updating several tables, etc.)

n Basic Syntax:

```
<CFQUERY NAME="query_name" DATASOURCE="ds_name"  
  USERNAME="username" PASSWORD="password">
```

```
  SQL statements
```

```
</CFQUERY>
```

n (Demo 5)

Creating Dynamic Pages

- n Cold Fusion provides several tags to control page flow
- n These options allow you to provide the user with different pages based on user selection or the processing of query results.
- n Use these options to make the web site more intelligent and “application-like”.

Redirecting Application Page Requests (CFLOCATION)

- n You can redirect a page request to another page or to another URL using the CFLOCATION tag.
- n Use to define an application page that performs one or more CFQUERYs and then moves on to another page.
- n Also useful if you want the URL to which the user is directed to depend upon a dynamic parameter.
- n Basic Syntax:

```
<CFLOCATION URL="filename.cfm">
```
- n (Demo 6)

Including Application Page Files (CFINCLUDE)

- n You can include one or more template files in a given .cfm template.
- n Use for creating reusable code, to be included in a variety of pages.
- n **Basic Syntax:**

```
<CFINCLUDE TEMPLATE="TemplateName">
```
- n (Demo 7)

Conditional Processing (CFIF, CFELSEIF, and CFELSE)

- n Use simple conditional statements to customize the behavior of your application.
- n You can also create complex conditional statements, combining multiple conditional statements with boolean operators.

- n **Basic Syntax:**

```
<CFIF value operator value>
```

```
    HTML and CFML tags
```

```
<CFELSE>
```

```
    HTML and CFML tags
```

```
</CFIF>
```

- n (Demo 8)

Looping (CFLOOP)

- n Looping lets you repeatedly display a set of instructions or some output depending on a particular set of conditions.
- n `<CFLOOP>` allows for four different types of loops:
 - Index loops (also called FOR loops)
 - Conditional loops (also called WHILE loops)
 - Looping over a query
 - Looping over a list
- n **Basic Syntax:**

```
<CFLOOP INDEX="LoopCount" FROM="1" TO="100">  
    The value is <CFOUTPUT>#LoopCount#</CFOUTPUT>.  
    <CFIF LoopCount IS 7> <CFBREAK> </CFIF>  
</CFLOOP>
```
- n It is especially useful for looping outside of `CFOUTPUT`.

Using CF Functions

- n Cold Fusion provides a wide range of functions similar to those you would expect in a programming language.
- n Functions can be used to process user input before updating the database, or to manage the display of data.
- n Refer to the Cold Fusion documentation for a complete description.
- n (Example 5)

Creating Reports

- n Cold Fusion provides support for Crystal Reports:
 - Crystal Reports comes bundled with Cold Fusion v.3.
 - The Cold Fusion `<CFREPORT>` tag allows you to specify a CR definition. The report is then generated to HTML and displayed in the browser.
 - Alternatively, you can simply define the query and output for display only using Cold Fusion.
- n Example 6

Tips and Techniques

- n Using <CFTRANSACTION> and Stored Procedures
- n Unit Testing
- n Using Variables
- n When to Use JavaScript?

Using <CFTRANSACTION> and Stored Procedures

- n Problem: What if a problem happens during a database save?
- n Solution: Use Stored Procedures or a <CFTRANSACTION> tag.
 - Keep your common database queries in stored procedures. The application in general will be easier to maintain, and you can “back out” if the procedure fails.
 - Alternatively, whenever you need to perform more than one action, put them within a <CFTRANSACTION> tag.

Unit Testing

- n Problem: In debugging, it is sometimes difficult to tell if the problem lies in your SQL or your Cold Fusion logic.
- n Solution: Unit Test using RDBMS software.
 - Make sure that all your SQL is kept in stored procedures.
 - Test each stored procedure using database management software before testing it within Cold Fusion.
 - Note: This approach allows for various types of people to work on a single application (e.g., an interface designer and a programmer/database expert).

Using Variables

There are five types of variables:

- Form Elements Accessible from the <FORM>'s Action page only (the page *after* the page in which the form element is defined).
- Global Variables Accessible “application-wide”, for a user-defined period of time
<CFCOOKIE>
- Local Variables Accessible only during the lifetime of the page in which the variable is defined
<CFSET>
- URL Variables Passed through the URL, and accessible during the lifetime of the page which is called
- Query columns Accessible only during the lifetime of the page from which the query is executed

When do you use each kind of variable?

n Form Elements:

- Use input fields for user input
- Hidden fields are useful for storing primary key information

n Global Variables (<CFCOOKIE>):

- Use for “App-wide” information that won’t change (e.g., user id)
- Be aware of limitations on Cookies and conflicts with other apps

n Local Variables (<CFSET>):

- Use to process information in a given page

n URL Parameters

- Use to pass local variables to be used in the next page
- Pass information “back” to an earlier form (e.g., an edit page expecting a primary key id).

When To Use JavaScript ?

- n Problem: JavaScript and Cold Fusion both provide capabilities for field validation and dynamic HTML. When should you use which?
- n Solution: Try to perform “client-oriented” activities on the client (JavaScript) and data- or “server-oriented” activities on the server (Cold Fusion).
- n JavaScript is especially suited for maintaining Frames and interface elements (e.g., buttons, etc.)
 - Perform most field validation using Cold Fusion.

Advanced Features

- n Java
- n Email Support
- n File Access
- n Custom Tags
- n Support for Standards (COM, LDAP)

Additional Resources

n Cold Fusion Web Site:

- <http://www.allaire.com>

n Developers Forum (from CF Web Site)

n Cold Fusion Users Guide

- <http://aaadev.gsfc.nasa.gov/cfdocs/user/index.htm>

n Good Books

- The Cold Fusion Web Database Construction Kit; Ben Forta (Editor), et al
- Cold Fusion 3.0 : Application Development Toolkit (Internet Professional Series); John Desborough