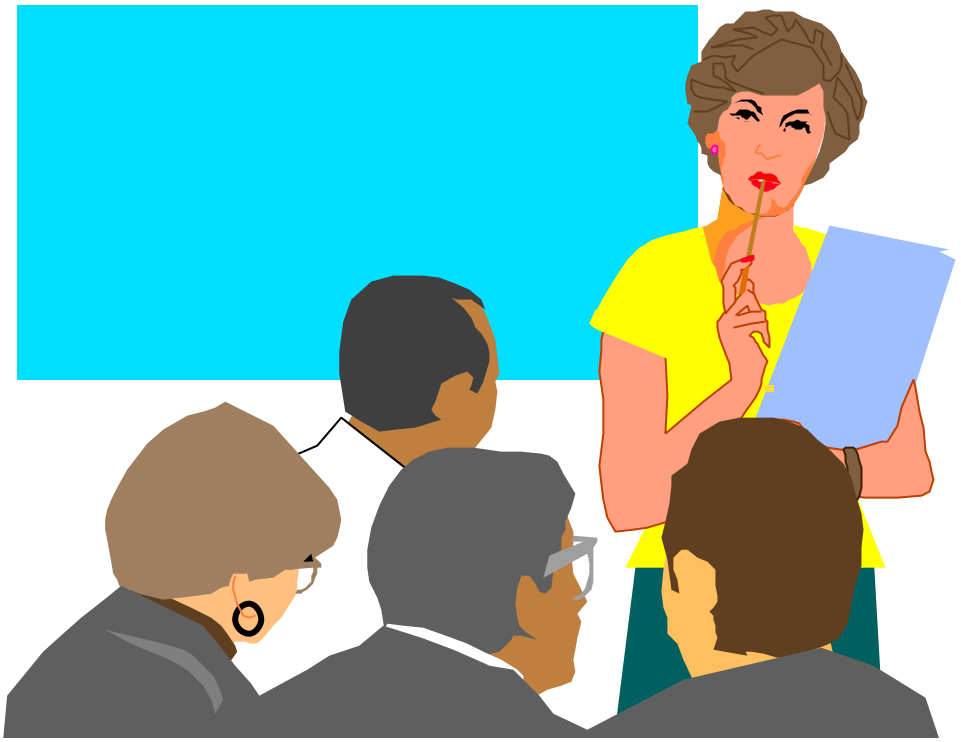


Cold Fusion 速成课程

培训教材

北京朝阳区外大街 22 号泛利大厦 612 100020

010-65880915/16/17/18 免费热线：8008100708

http://www.coldfusion.com.cn *http://www.webleader.com.cn*

Cold Fusion 速成课程

Unit 1：课程总揽

全部课程为期三天。

学员要求：

- 熟悉 Web 技术
- 理解 Web 服务器特性
- 熟悉 HTML 及句法
- 熟悉 SQL 语言，包括 insert,select,update,delete 等

课程内容为：

- 建立 Cold Fusion 开发环境
- 使用 Cold Fusion Studio
- 使用 Cold Fusion tags 发布动态信息
- 常用源代码重复利用
- 使用 Cold Fusion 建立表单
- 建立检索界面
- 建立用户菜单界面
- 接受用户信息，对数据库更新
- 加入 JavaScript 在客户端检验数据有效性
- 界面安全性

课程结束后，您将能够：

- 建立、管理 Cold Fusion 开发环境
- 使用 CFML 存储、输出数据
- 定义 ODBCDriver，与数据库连接
- 建立查询语句，动态发布、添加、更改信息
- 加入 JavaScript，在客户端验证数据有效性
- 在应用中使用本地、Cookie 及其他变量
- 创建交互性的数据库 Web 应用

Unit 2：Cold Fusion 简介

静态页面和动态页面

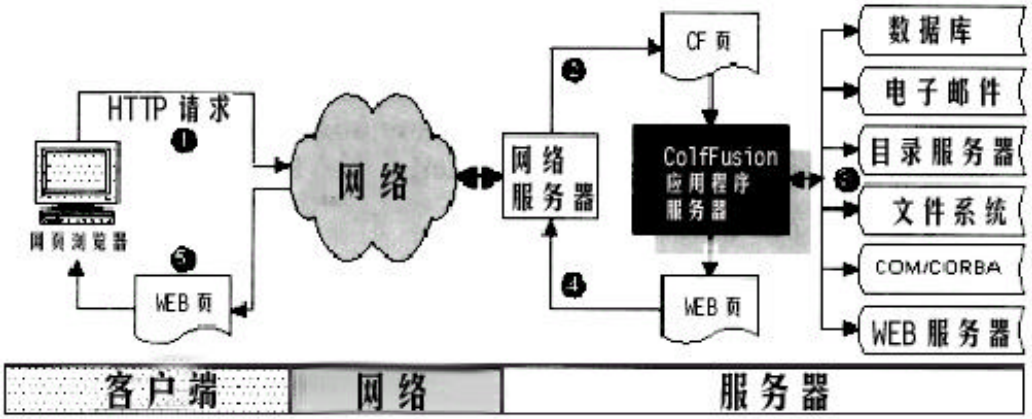
在最初的互联网上,网页是静止的，所谓静止就是指 Web 服务器只是简单地把存储的文本文件和图形文件传给用户，只有编辑者使用文字处理器和图形编辑器对他们进行修改,它们才会发生改变。

随着 Netscape 推出 JavaScript 和 Sun 推出 Java，这两大语言技术让网页浏览者可以见到页面有了一些变化，例如，一个 JavaScript 程序可以执行一个计算器的功能，用来

存储和计算数值，并在屏幕上显示结果 。Java 的功能更加强大，因为它具有丰富的环境和快速、表现力强的语言。

然而服务器端仍旧是没有动态变化，服务器端提供的程序和文本文件和图形文件一样没有变化。但是不久，服务器可以通过 Common Gateway Interface（CGI）、Active Server Pages（ASP）、ISAPI 应用程序等技术给浏览器传送用户定制的内容，Coldfusion 也是其中的一种动态发布技术，并且功能强大，使用简单。如果你是一个即将在网上建立动态 WEB 应用和交互 WEB 站点的专业开发者，Allaire 公司的 Cold Fusion 应是你可以选择的一个很好的开发工具。它提供集浏览器、服务器和数据库技术于一体的强大的 WEB 应用。

利用 Coldfusion 发布动态信息的原理如下：



1. 用户点击提交键或超级链,用户的 Web 浏览器通过 Internet 或 Intranet 发送 HTTP 请求到 Web 服务器。
2. Web 服务器通过服务器的应用程序接口将用户请求和相应的页交给 Cold Fusion 服务器。
3. Cold Fusion 从用户端读取用户数据并处理其中的 CFML（Cold Fusion 制标语言），基于 CFML，服务器可通过 Cold Fusion 应用程序接口或通过 COM/DCOM.建立与数据库服务器、文件系统、SMTP 系统等的联系。
4. Cold Fusion 动态地建立一个 Web 页，并将此页返回给 Web 服务器。
5. Web 服务器返回标准的 HTML 页给用户浏览器。

Cold Fusion 特点：

这里列出的特点是结合用户在实际应用中所关心的几个方面提出的，

1. 快速开发性：利用 Cold Fusion 强大的可视化工具、直接的编程环境和便于管理的应用服务器，可以快速、简单地建立复杂的应用程序。
2. 开放的综合性：Cold Fusion 提供数据库、邮件服务器、文件目录、XML、COM、CORBA 等服务器系统的支持。
3. 安全性： 对服务器的开发和管理进行访问控制，利用高级安全特性避免运行时的其他应用。使用 SSL 对远程文件，数据传输进行加密。
4. 高效率： 程序员可指定在多次调用数据库时，数据库保持打开状态多长时间。缓存经常使用的查询结果，在保证数据一致性的同时减少数据库通信。

5．出错处理：当 Cold Fusion 不能找到模板文件中指定的变量时，他提供详细的错误信息和原因。
6．输入合法性检查：在输入表格中用一隐含域实现输入合法性检查。一些标准的合法性检查是数据类型、范围等。
7．可扩展性：利用 CFML、C\C++、COM、CORBA、JavaScript、VBScript 建立自己的部件和标识。通过<CFX_XXX>调入页中。
8．多种数据源：本身带有 Oracle 和 Sybase 的连接驱动，可以和任何 ODBC 兼容的数据库连接，通过 OLE-DB 与 Exchange、 Lotus Notes 等连接。
9．全文索引：利用内嵌的 Verity Search 97 全文索引非结构化的数据、标准文本及桌面文件。
10．源代码重用：可将经常使用的代码保存为 CFML 模块或 CFX，日后重复应用于其他应用和界面中。
11．Email：数据库输出不仅能送至 Web 浏览器，还能作为 email 发给客户。
12．Cookies：提供一种途径，可在客户端存储信息以做将来检索用。
13．与协议的结合性：支持多种网络协议，如 FTP、HTTP、MAIL、POP、LDAP。

这里介绍的 Cold Fusion 特性，有的没有经过自己亲自使用理解会不深刻，如果在学完了这三天的课程，经过了亲自动手的练习，有了一端时间的开发经验后再反过来看这些特性，体会会更加深刻的。

运行 Cold Fusion 应用软件应具备以下部件：

- Cold Fusion Server ：一个高性能、可升级、开放平台，用来传输 Web 应用，从简单的数据驱动页到网络中的全套电子商业解决方案。
- Cold Fusion Administrator ：一个完全的应用软件管理控制平台。
- Cold Fusion application pages ：以 cfm 为后缀的文件，内容为 HTML 和 CFML 的混合编码。
- ODBC data sources and other data sources ：ODBC 数据源或其他数据源。Cold Fusion 支持使用 32 位 ODBC 驱动或 OLE DB。
- Cold Fusion Extensions ： 一个开放的，基于扩展制标语言的构架，通过组成服务器新的成份来扩展 Cold Fusion。使用 COM，CORBA，C/C++，VB Script，JavaScript or CFML 来连通企业系统。

设置 Cold Fusion 开发环境

ColdFusion 中带有 Administrator，在其中可以对虚拟目录、安全设置、数据源、日志文件、文件索引等进行配置。以安装时写入的密码进入 Administrator。

对于刚开始使用 Cold Fusion 来进行简单应用的用户，Administrator 中有两项是必须配置好的，一个是 Mapping（设置虚拟路径），一个是 ODBC（设置 ODBC 数据源）。下面讲解一下在 Administrator 中常用到的一些选项：

在 Server 中：

- ColdFusion Server 的基本配置，一般情况下不必修改
- 配置 Administrator 和 ColdFusion Studio 的基本和高级安全项。
- 授权和配置 ColdFusion 的 application，session，和 client 变量

- 设置虚拟路径
- ColdFusion 版本信息

在 Data Sources 中：

- 配置内置的数据库驱动程序（Oracle and Sybase）
- 配置 ODBC 数据源
- Verifying a ColdFusion data source

在 Extensions 中：

- 配置 CFX tags、用户自己使用 C++ 编写的 tags
- 配置 Java applets

在 Logging 中：

- 指定一个目录存放 ColdFusion 的日志文件
- 设置邮件日志文件选项
- 查看 ColdFusion 的日志文件

在 Automated tasks 中：

- 生成各种定期运行的任务

在 miscellaneous 中：

- 配置管理员的邮件服务器，以便 ColdFusion 服务器能自动发送信息给管理员
- 设置程序运行时的环境变量
- 设置工作路径、初始化路径和超时等

CFML 语言

CFML 语言是 Cold Fusion 特有的一种语言，也是 Cold Fusion 重要的组成部分。它是一种服务器端的脚本语言。它由 CFML Tags 、CFML Functions 和 WDDX JavaScript Objects 组成。使用 CFML tags 和数据库发生联系、处理数据并显示结果，CFML tags 十分简单，语法类似 HTML 的元素语法。CFML 提供大量有用的函数，如动态赋值函数 、队列函数、串函数、查询函数等。当执行 WEB 分布式交换时使用 JavaScript Objects。WDDX 是 WEB 动态数据转换的缩写，它本身是一种扩展语言。它可将大量数据，无论是数字还是结构体、记录集转为文本块，并将其放入 WDDX 格式中，不受限制地从一个地方送到另外一个地方。

CFML 语言特点：

1. 处理过程封装。用<CFxxx> </CFxxx>来封装一个 CF 处理过程。
2. 变量处理：在动态页中处理变量。变量可在页与页之间传递，可以建立浏览器的 Cookie 变量，管理服务器、应用程序、会话层和用户层数据。
3. 页流向控制：可通过布尔操作(AND, OR, NOT, etc.)和 CFIF CFELSE CFELSEIF、CFSWITCH CFCASE CFDEFAULTCASE 和 CFLOOP 等 Tag 来对页的流向进行控制。
4. 可扩展性：通过 CFX_xxx,可以将自行开发的应用程序引入应用中来。

练习题：

可以上网观看几个 ColdFusion 的实际网站，对 ColdFusion 有一个大致的了解：
www.coldfusion.com.cn

Unit 3：初试身手

安装 ColdFusion

到目前为止 Cold Fusion 支持 Microsoft Windows 和 Sun Solaris 两种应用环境，在这里我们着重介绍 ColdFusion 在 Windows 环境中的安装过程。

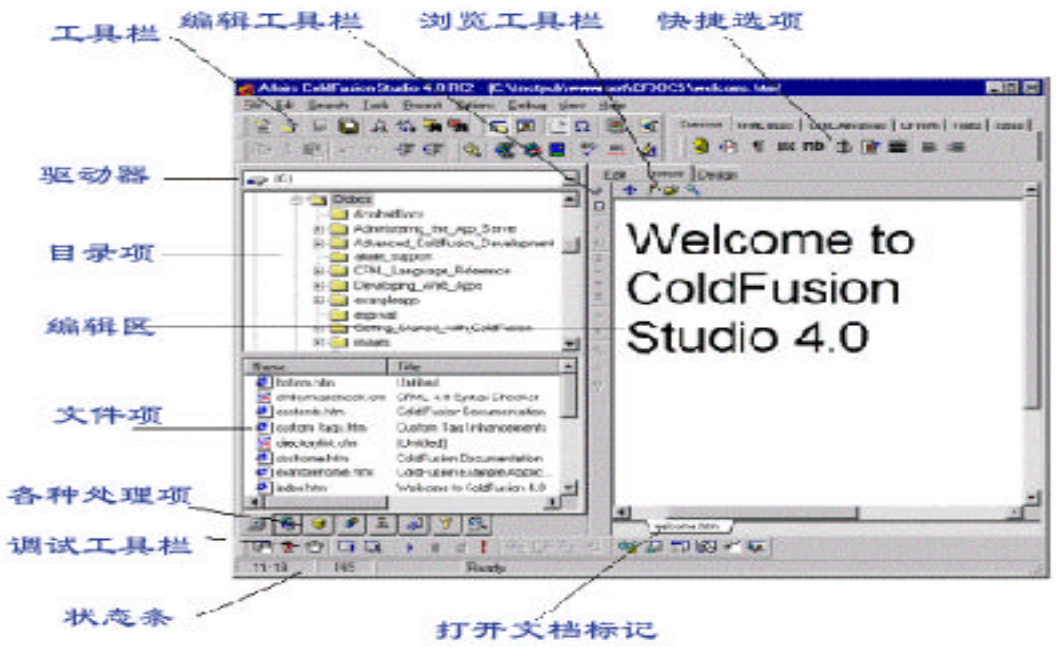
在安装 ColdFusion 之前要先确定本机的 Web Server 已经安装并且正在运行。

1. 运行 ColdFusion 光盘上的 setup.exe
2. 阅读 ColdFusion 许可证协定
3. 填入用户姓名，公司名，和序号。
4. 选择安装目录，可使用缺省目录(c:\CFUSION)。
5. 选择 Web 服务器，如 NT IIS 或 Personal Web Server
6. 选择 ColdFusion HTML 的路径，缺省为系统的主路径，如想使用别的路径，可改变其值。
7. 选择安装内容。
8. 系统要求填入 ColdFusion Server 的密码，该密码以后配置时使用。
9. 系统要求填入 ColdFusion Studio 的密码。
10. 显示 Programe Folders。
11. 显示当前配置。
12. 开始安装。

使用 ColdFusion Studio

ColdFusion Studio 是 ColdFusion 的一个集成化的高效开发环境，带有可视化的编程和数据库工具。几乎所有工作区中的部件都是可以拖动的。

下图是 ColdFusion Studio 的工作区：



使用 ColdFusion Studio 可以：

- 可以看到源码
- 在 CFQUERY 快速建立 SQL 语句
- 从快节选项中快速选择 HTML 和 CFML tag
- 可用内部或外部的浏览器浏览页面，在设计模式中可以编辑页中内容。
- 为应用建立工程，使文件易于维护和上载。
- 使用扩展查询和替换实现全部改变。
- 可以将文件的一部分存储为可重复利用的部分。
- 可以查询全部 ColdFusion 函数、常数、表达式、变量。
- 调试应用代码。
- 可察看整个站点结构。
- 可以进行 HTML 和 CFML 代码正确性验证。
- 可验证链接的正确性。
- 可以进行团队开发。

在各种处理项中有：

- 文件 – 左边的上面显示框中显示本地或网络的目录结构，下面的显示框显示目录中的文件
- 远端 – 可以从远端增加 FTP 和 RDS 服务器
- 数据库 – 配置 ColdFusion servers，查看 data sources
- 工程 – 建立工程来存储和管理应用文件
- 站点查看 – 显示当前文档的物理结构和连接情况
- 摘录 – 存储可重复利用的代码
- 帮助 – 查看 ColdFusion 文档和其他在线资源
- Tag 检查 --上面显示框中显示 Tag 树，下面的显示框显示 Tag 检查

总之，ColdFusion Studio 是进行 ColdFusion 的最有力工具，它的许多性能和优点需要你在实际使用中不断认识和探索。

开发流程

CFM 文件可以由纯 CFML 代码组成，可以是 HTML 和 CFML 混编代码，所以整个开发流程可以全部在 ColdFusion Studio 进行，也可以在第三方的工具如 Frontpage、Dreamweaver 等上开发 HTML 网页界面，再在 ColdFusion Studio 中加入 CFML 代码，组成完整的 CFM 文件。对于开发后的 CFM 文件可以在 ColdFusion Studio 中进行调试。所以整个开发流程可分为界面开发、代码混编、后期调试。

在进行开发之前，要明确以下几点：

- 明确开发 CFM 文件要解决的问题
- 进行应用程序的总体设计.
- 决定应用中有哪些输入数据和输出数据.
- 决定应用程序要求用户级还是源码级的安全措施.
- 设计应用程序总体结构

使用 ColdFusion 变量

ColdFusion 支持多种类型的变量，下面以图表形式介绍这几种变量：

变量名称	变量描述
查询变量	当运行一个查询时，可以使用它的结果作为动态参数。例如，建立一个查询叫 LookupUser ，用来查出已知用户名的 ID ，这个 ID 或许会在另一个查询、一个 CFOUTPUT 输出中用到，ID 就是一个查询变量
本地变量	使用 CFSET and CFPARAM 这两个 tag 建立的变量。例如 <CFSET A=5> 建立变量 A 赋值为 5.。这种变量只有在包含他们的页面建立后才有效。
URL 变量	这个变量跟在一个 URL 后面，形式是 variablename=value。可用在页与页之间传递参数值。
表格变量	表格被用做是传递变量的常用方法。当用户在表格中输入一个数据，一个带有表格中此项名称的参数就被传递到了表格的处理页，在处理页可以直接通过 From..名称来调用数据。
Client 变量	Client 变量用来存放客户端长期有效的数据信息,可以把 Client 变量存储在系统注册表里,一个数据库里或 cookie 变量里.
Server 变量	定义的 Server 变量,所有的应用程序都可以调用,并长期有效,直到 ColdFusion 服务器 shut down.
Session 变量	用于存储短期有效的单个用户要访问的信息和一系列请求信息,不象 client 变量，session 变量存储在服务器的内存里,并且可以设定有效期.
Application 变量	Application 变量定义 这些变量存储在服务器内存中,可以设置有效时间. Application 定义一些单独和特殊使用的变量,存放在 Application.cfm 文件中.
HTTP Cookies	HTTP Cookie 存储在浏览器端，在页面被调用时有效。你可以使用 CFCOOKIE tag 来建立 Cookie 变量。
CGI 环境变量	在页面中有提交请求或调用其他页面时环境变量十分有用，它包含有服务器和客户端的许多信息。

创建本地变量

下面着重介绍如何创建本地变量，其他变量在以后用到时再详细介绍。本地变量在建立他们的页或包含在此页中的其他页中有效。有两个 Tag 都可以用来建立本地变量 CFSET 和 CFPARAM，他们稍有不同：

名称	语法	描述
CFSET	<CFSET 变量名称 = 变量值,参数或表达式>	CFSET 定义变量如果变量存在，则重新被赋新值。 静态变量:<CFSET FirstName="Jack"> 动态变量: <CFSET UserDescription="#UserName#"> 定义数组:<CFSET myarray=ArrayNew(1)> <CFSET myarray[1]="January"> <CFSET myarray[2]="February"> 表达式: <CFSET TotalValue=2 * (4 + 5)>
CFPARAM	<CFPARAM NAME="变量名称" (DEFAULT="默认值")>	● 只使用 NAME 属性可以用来检测变量是否存在，如果不存在 ColdFusion 服务器停止此页的执行. ● 使用 NAME 和 DEFAULT 属性,先检测变量是否存在,如果存在,继续执行,原值不变；如果不存

		在，建立变量并赋予默认值"

把#变量名称#放在一对<CFOUTPUT></CFOUTPUT>中可以输出此变量。

例如：在<CFSET FirstName="Jack">之后写入：

```
<CFOUTPUT>
    Your FirstName is # FirstName #.
</CFOUTPUT>
```

结果是：Your FirstName is Jack.

使用 ColdFusion 函数

ColdFusion 函数大致可以分为以下几大类：数组函数、日期和时间函数、结果函数、显示和初始化函数、动态赋值函数、列表函数、结构函数、数学函数、字符串函数、系统函数、查询函数等。他们使用简单，功能强大。

下面举例说明几个简单函数的使用：

- Trim(string)，去掉 string 中前后的空格。

```
<CFSET FirstName="   Jack   ">之后写入：
<CFOUTPUT>
    Your FirstName is "#trim("#form.myText#")#"。
</CFOUTPUT>
```

- Encrypt(string, key)，用 key 对 string 进行加密。
  Decrypt(encrypted_string, key)，用 key 对 encrypted_string 进行解密。

```
<CFSET string = “ this is a test of encrypt ”>
<CFSET key = “abc”>
<CFSET encrypted = encrypt(string, key)>
<CFSET decrypted = decrypt(encrypted, key)>
<CFOUTPUT>
     The string: #string# <BR>
     The key: #key#<BR>
     Encrypted: #encrypted#<BR>
     Decrypted: #decrypted#<BR>
</CFOUTPUT>
```

- IsDefined("variable_name")，用它可以检测一个变量是否存在。

```
<CFIF Not IsDefined("FirstName ")>
<CFOUTPUT>the variable is not exist!!!</CFOUTPUT>
</CFIF>
```

ColdFusion 函数种类繁多，在此不再一一说明。在实际应用中，你会发现许多有用的函数，和一些函数的巧妙的使用方法。仔细研究一下这些函数，你会有很大的收获。

练习题：

自己动手安装 ColdFusion 和 Studio，配置 Administrator。创建第一个 ColdFusion 例程：用 CFSET、CFPARAM 定义几个变量，并输出。在 Documentation 中看以下 ColdFusion 都有哪些函数。

Unit 4：发布数据库内容

ColdFusion 支持多种类型的数据库，但不同的版本支持的数据库类型有所不同，专业版支持所有 ODBC 数据源和 OLE DB 数据源(尽限 Windows 环境中)、企业版支持所有 ODBC 数据源和 OLE DB 数据源(尽限 Windows 环境中)，并带有两个本地数据库驱动，分别为 Sybase System 11 和 Oracle 7.3 、8.0（尽限 Solaris 和 Windows NT 环境中)。你可以在 Administrator 中的 Data Sources 页中配置数据源，但这些数据源的驱动必须已经安装在你的机器中了。

ColdFusion 中数据库的使用十分简单方便，在配置好数据库以后，查询、输出等都可以通过几个定制标记、标准的 SQL 查询语句和几条简单语句处理完成。

设置 ODBC 数据源

你可以在你的机器中安装 Microsoft Data Access Components (MDAC 2.0)，它会在你的机器中安装几种 ODBC 驱动程序：Microsoft SQL Server、Microsoft Access and FoxPro databases、Borland dBase-compliant databases、Microsoft Excel worksheet data ranges 和 Delimited text files 。

下面介绍配置 ODBC 数据源的步骤：

1. 进入 Administrator 中的 Data Sources 页
2. 在 data source name 中输入一个名字
3. 在 ODBC driver 中选取一个适当的数据库类型
4. 双击 Add 按钮打开建立 ODBC 数据源页
5. 输入数据源文件目录和名称和其他一些信息，双击 CF Settings 按钮进入可选设置页，在此页中可以设置 Login 信息，限制 SQL 的操作。
6. 双击 Create 按钮，新的数据源被加入到系统中了，并显示在 Data Sources 页中，双击数据源名字可以在进入数据源页
7. 检查数据源连接是否正确，打开 Verify Data Source 页，选择要检查的数据源名字，双击。

查询数据库

在所有数据库的使用中，查询数据库是基础。下面介绍如何在 CFQUERY 标记中使用结构化查询语句（SQL）查询数据库。

下面是一个简单的数据库查询的例子，要查询的数据库的 ODBC 数据源名叫 CompanyDB，建立一个查询名叫"EmployeeList"，从 Employee 表中返回所有的记录：

```
<CFQUERY NAME="EmployeeList" DATASOURCE="CompanyDB">
    SELECT *
    FROM Employees
</CFQUERY>
```

在这里查询名称十分重要，它用来标识此次查询，在以后的结果输出时或引用查询结果时要用到它。

CFQUERY 的标准语法是：

```
<CFQUERY NAME="query_name"// (必须)定义查询名称
```

```
    DATASOURCE="ds_name"//(必须)ODBC 数据源名称
    DBTYPE="type"//数据驱动的类型
    DBSERVER="dbms"//针对本地数据库驱动，指定它的数据库服务器的名称，一旦指
                    定，DATASOURCE 指定的数据源无效。
    DBNAME="database name"//数据库名称（只对 Sybase System 11 驱动有效），一
                        旦指定，DATASOURCE 指定的数据源无效。
    USERNAME="username"//数据源要求的用户名称
    PASSWORD="password"//数据源要求的用户密码
    MAXROWS="number"//最多返回多少行记录
    BLOCKFACTOR="blocksize"//此参数用于 ORACLE 本地数据库驱动和 ODBC 驱动，
                        指定每次从服务器上读取的最大记录数
    TIMEOUT="milliseconds"//指定多少个毫秒后表示执行超时。对于大部分 ODBC 驱
                        动无效， SQL Server 6.x 或以上版本的驱动支持
    CACHEDAFTER="date" //指定一个日期值，ColdFusion 在此日期后使用查询结果
                        缓存，要在 Administrator 中开启查询缓存项
   CACHEDWITHIN="timespan" //指定时间间隔，在时间间隔内如果原始查询结果无
                        效则使用缓存的查询结果
    PROVIDER="COMProvider" // COM 提供者 (只限 OLE-DB )
    PROVIDERDSN="datasource" // COM 提供的数据源名字(只限 OLE-DB )
    DEBUG="Yes/No"//使用查询调试>
    SQL statements
</CFQUERY>
```

输出结果

在输出查询结果前，可以先判断查询是否有结果。例如对于上面的那个查询的例子，可以用下面语句判断查询结果是否为空：<CFIF # EmployeeList. RecordCount# EQ 0>其中 EmployeeList 为查询的名称。

在输出结果时常用到的定制标记有 CFOUTPUT、CFTABLE、CFCOL 等。CFOUTPUT 的输出功能强大，它可以用来输出查询结果，也可以用来输出其他操作的结果。它的标准语法是：

```
<CFOUTPUT QUERY="query_name"//（可选）查询名称
          MAXROWS="max_rows_output"//（可选）指定输出的最大列数
          STARTROW="start_row"//（可选）从哪一行开始输出
          GROUP="parameter"//（可选）按 GROUP 指定的参数成组的输出结果>
</CFOUTPUT>
```

针对 CFQUERY 查询和 CFOUTPUT 输出结果，我们给出一个完整的例子：

我们定义一个 CFM 文件：employeelist.cfm 用来查询数据库和输出结果

```
<!--- Query to select customers --->
<CFQUERY NAME="EmployeeList" DATASOURCE="CompanyDB">
    SELECT *
    FROM Employees
</CFQUERY>
<HTML>
<HEAD>
```

```
    <TITLE>Employee List</TITLE>
</HEAD>
<BODY>
<H2>Employee List</H2>
<!--- Output section --->
<CFOUTPUT QUERY="EmployeeList">
    <HR>
    # FirstName# # LastName#
    (Phone: #PhoneNumber#) <BR> //其中 FirstName、
</CFOUTPUT>
</BODY>
</HTML>
```

你可以在一个应用程序中调用这个文件

```
<A HREF="路径名称/employeelist.cfm">Employee List</A>
```

ColdFusionServer 执行了查询后，返回结果，送回到客户端看到的是标准的 HTML 语言编写的结果：

```
<HR>
Deborah Jones (Phone: 612-227-1019) <BR>
<HR>
John Smith (Phone: 507-452-7224) <BR>
<HR>
Frank Wilson (Phone: 612-831-9555) <BR>
```

使用表格输出结果

因为 CFOUTPUT 标志中可以包含标准的 HTML 代码，所以可以使用 HTML 表格输出查询结果，但 ColdFusion 中也提供自定义的标志 CFTABLE 和 CFCOL 来输出结果，下面我们分别介绍一下这两种输出方式。

- CFML 表格

CFTABLE 定义要输出的查询名称，最大行数等参数；CFCOL 定义每一列的标题和宽度。两个标记同时使用定义整个表单。还以上面的查询为例，使用 CFTABLE 和 CFCOL 来输出 EmployeeList 查询的结果表单。表单由三列组成，不超过 10 行，分别列出 CompanyDB 数据库 Employees 表单中的 FirstName、LastName 和 PhoneNumber。

```
<CFTABLE QUERY=" EmployeeList " MAXROWS=10>
    <CFCOL HEADER=" FirstName " WIDTH=25
        TEXT="<I># FirstName #</I>">
    <CFCOL HEADER=" LastName " WIDTH=15
        TEXT="# LastName #">
    <CFCOL HEADER=" PhoneNumber " WIDTH=15 ALIGN=RIGHT
        TEXT="# PhoneNumber #">
</CFTABLE>
```

- HTML 表单

尽管使用 CFTABLE 和 CFCOL 来输出查询的结果表单十分简单易用，但是它毕竟

没有使用 HTML 编写表单那样灵活，有的编程人员喜欢使用 HTML 编写具有自己风格的表单，所幸的是 CFOUTPUT 标志中可以包含标准的 HTML 代码，这就可以使用 CFOUTPUT 和 HTML 代码编写 HTML 表单。还以上面的查询为例，使用 CFOUTPUT 和 HTML 代码编写 HTML 表单。

```
<table border="0" cellpadding="0" cellspacing="0" cellspaceing="0">
<tr><td colspan="6" width="400"><font size="2"><strong>雇员信息浏览：
</strong></font></td></tr>
<tr><td colspan="6"
width="400">      </td></tr>
<CFOUTPUT MAXROWS =10 StartRow = 1 query = EmployeeList >
<tr>
 <td width="100"><font size="2" face="文鼎 CS 长美黑 "><strong> 名
 称:</strong></font></td><td width="50"># FirstName #</a></td>
 <td width="100"><font size="2" face="文鼎 CS 长美黑 "><strong> 姓
 氏:</strong></font></td><td width="80"># LastName #</td>
 <td width="100"><font size="2" face="文鼎 CS 长美黑 "><strong> 电
 话:</strong></font></td><td width="80"># PhoneNumber #</td>
</tr>
</CFOUTPUT>
</table>
```

如果学员有兴趣可以在练习中分别用 CFTags 和 HTML 表单输出查询结果，看看两者在输出形式上有何不同。

练习题：

使用 Access 或 SQL Server 建立一个小型的数据库，配置 ODBC，做简单查询，并输出结果。

Unit 5：模版重用和结构控制

ColdFusion 在结构上具有很大的灵活性，它可以对应用文件的流向进行控制，针对不同的条件调用不同的文件，它也支持源代码重用，具有很强的可扩展性。可将经常使用的代码保存为 CFML 模块或 CFX，日后重复应用于其他应用和界面中。也可以利用 CFML、C\C++、COM、CORBA、JavaScript、VBScript 建立自己的部件和标识。通过<CFX_XXX>调入页中。

常用源程序重用

使用 CFINCLUDE 可以完成常用源程序重用的任务。

你可以利用 CFINCLUDE 做许多有用的事：

- 如果有一段代码是你在多个应用中重复使用的，每次书写此段代码十分浪费时间，你可以将此段代码放到单独的页中，在每一个要用到它的应用程序中调用。
- 你可以建立一个头文件"header.cfm" 和尾文件"footer.cfm"，在文件中包括头尾的格式和风格，在每一个应用文件中调用这两个文件使你的所有文件看起来形式风格统一。
- 如果你编写了一个极其复杂的 CFIF/CFELSE 语句，使阅读你程序的人觉得你的程序结构复杂、难于阅读，你可以将一些处理放到新的一页中，并给它起一个一望便知的名字，这样可使你的程序清楚、易读。

CFINCLUDE 的语法是：

```
<CFINCLUDE TEMPLATE="FileName">  //文件名要包括相对的路径名
```

例如：

```
<CFINCLUDE TEMPLATE="header.cfm">
... Page contents
<CFINCLUDE TEMPLATE="footer.cfm">
```

或是：

```
<CFINCLUDE TEMPLATE="../index.cfm">
```

使用 Custom tags

由于 ColdFusion 具有很强的扩展性，用户可以自己编写 Tag 在应用程序中使用，也有不少 ColdFusion 的爱好者和技术人员自己编写一些十分有用的 Tag，供大家使用，在 http://www.allaire.com/taggallery中你就可以找到不少有用的 Tag。所有用 CF_打头的 Tag 都是 CFML 用户用 CFML 编写的 tags，那些以 CFX_ 打头的 Tag 是用 C/C++编写的 ColdFusion 扩展。

下面先讲如何加入一个 CFX_打头的 Tag：

- 首先 C/C++编写的程序必须先编译，生成 DLL 为后缀的动态链接库后才可以用做 Custom tags。
- 打开 ColdFusion Administrator 中的 CFX Tags 页，所有的已注册的用户 tags 都列在那里。
- 填入新的 CFX _xxx 名称，双击 Add 键，进入这个新的 Tag 页。
- 如果在前一页没有填入名称，在 tag name 后填写 CFX_ xxx。
- 在 Server library (DLL) 后填写引用的 DLL 和路径和名称。
- 输入执行 tag 的程序名，这个程序必须和 DLL 中存在。
- 选中 Keep library loaded ，防止以后要重新装入。
- 在 Description 中填写这个 CFX Tags 的功能。
- 双击 Add 来存储这个新的 tag.

对于 CF_打头的 Tag 使用起来十分简便，可以直接调用，但 CF_后面跟随的名字要和 CFML 文件名称相同，例如： 使用 CF_MyTag 来定义 MyTag.cfm 这个文件。

也可以使用 CFMODULE 来调用用户自定义的 tag 。CFMODULE 可以处理可能发生的用户自定义的 tag 名称冲突、给自定义的 tag 传送参数。

它的语法如下：

```
<CFMODULE TEMPLATE="template" //描述 CFM 文件的相对路径
    NAME="tag_name"           // tag 的名称
    ATTRIBUTE="value"         //传递的参数
    ATTRIBUTE="value"
    ...>
```

使用条件处理

Coldfusion 有两个 Tag 可以进行条件处理：CFSWITCH 和 CFIF。

CFSWITCH 条件处理

和 CFSWITCH 紧密相连的还有 CFDEFAULTCASE 和 CFCASE。CFSWITCH 根据一个表达式的

值，判断它是否符合 CFCASE 规定的某一个值，然后匹配结果；CFDEFAULTCASE，处理 CFCASE 的值都不符合的情况。它们的语法：

```
<CFSWITCH EXPRESSION="expression">
  <CFCASE VALUE="value" DELIMITERS="delimiters"> HTML and CFML tags</CFCASE>
  additional <CFCASE></CFCASE> tags
  <CFDEFAULTCASE> HTML and CFML tags</CFDEFAULTCASE>
</CFSWITCH>
```

例子：有一个记录员工情况的数据库 cfsnippets，库中表格 Employees 存放有 FirstName、LastName、Department 等员工所在哪个部门的情况，列出员工情况：

```
<CFQUERY NAME="GetEmployees" DATASOURCE="cfsnippets">
SELECT    Emp_ID, FirstName, LastName, EMail, Phone, Department
FROM     Employees
</CFQUERY>
<CFOUTPUT query="GetEmployees">
<CFSWITCH EXPRESSION=#Trim(Department)#>
   <CFCASE VALUE="Sales">
   #FirstName# #LastName# 在<B>销售部门</B><BR><BR></CFCASE>
   <CFCASE VALUE="Accounting">
   #FirstName# #LastName# 在<B>财务部门</B><BR><BR></CFCASE>
   <CFCASE VALUE="Administration">
   #FirstName# #LastName# 在<B>管理部门</B><BR><BR></CFCASE>
   <CFDEFAULTCASE>#FirstName# #LastName#不在上述三个部门<BR>
   </CFDEFAULTCASE>
</CFSWITCH>
</CFOUTPUT>
```

例如：有一个员工叫：zhangping 在管理部；有一个员工叫：chentao 不属于以上各个部门，输出的结果是：

```
zhangping 在管理部门
chentao 不在上述三个部门
```

CFIF 条件处理

和 CFIF 有关的 Tag 还有：CFELSE 和 CFELSEIF。它们的语法是：

```
<CFIF condition1>
    Display this text only if condition1 is true.
<CFELSEIF condition2>
    Display this text only if condition1 is false and condition2 is true.
<CFELSEIF condition3>
     Display this text only if condition1 and condition2 are false and
condition3 is true.
<CFELSE>
    Display this if condition1, condition2, and condition3 are false.
</CFIF>
```

条件中可以带有布尔表达式 AND、OR、NOT.

使用 CFIF 可以判断查询的情况：有一个数据库查询叫 CustomerSearch

```
<CFIF #CustomerSearch.RecordCount# IS 0>
```

```
<P>对不起，数据库中没有消费者的信息 </P>
<CFELSE>
<!--- 列出查询到的消费者名字 --->
    <CFOUTPUT Query=" CustomerSearch ">
    #FirstName# #LastName# <BR>
    </CFOUTPUT>
</CFIF>
```

文件重新定向

你可以使用 CFLOCATION 进行文件重新定向，文件的重新定向十分有用。你可以依据不同的条件使文件调用不同的“下一页”。

CFLOCATION 的语法十分简单：

```
<CFLOCATION URL="#Pagename#">
```

只要单纯地写下要调用的“下一页”的 URL。

例如，你可以使用 CFIF 去判断用户是否通过了用户验证，如果用户的密码或用户名不对，CFLOCATION 将文件定向到其他页去：

```
<CFIF #NewPassword# IS NOT `#PasswordConfirmation#'>
    <CFLOCATION URL="invalidpassword.cfm">
</CFIF>
```

本节介绍了两个 Tag：CFLOCATION 和 CFINCLUDE，它们都可以调用一个 CFM 文件，但它们的功能和用途不一样。

CFINCLUDE	用于嵌入 CFM 文件	可用于模版重用
CFLOCATION	用于打开 ColdFusion 文件或是 HTML 文件	可用于文件重新定向

练习题：

自己定义一个 CFM 页，在另外一页中调用它。

Unit 6：使用 Cold Fusion 建表

表单是用来搜集用户提交的信息的，HTML 中有表单的具体规定，ColdFusion 中也规定了相应的标志。表单提交的信息要通过 CGI（公共网关接口）传递给脚本程序，CGI 规定了 WEB 服务器如何向脚本程序发送信息；在收到脚本程序返回的信息又应该如何处理等内容。然而，ColdFusion 取代了用 C/C++或 Perl 编写 CGI 的必要；但 ColdFusion 也能利用 CGI 与 WEB 服务器或 WEB 服务器的 API 连接，此外要强调的是 ColdFusion 不是 CGI 执行程序，而是一个多线程的系统服务，更加稳定。

HTML 表单

HTML 表单的主要功能是搜集用户端要提交到服务器的信息。

HTML 使用一对<Form></Form> 作为表单的开始和结束，表单中可以有<input>、<select>和<textarea>三种标签，分别用来定义表中不同类型的内容。

Form 具有三个属性：Action、Method 和 Enctype。Action 通知服务器，提交的内容由哪个文件来处理；Method 指定在服务器和 CGI 之间通信的方法，有 POST 和 GET 两种；Enctype 指定所发文件的类型。例如：<Form Action=”xx.c” Method=”POST”>。

在一份<Form>中,你所希望用户输入的绝大部分信息都放在<input>标签中。在<input>标签中有多个不同的类型：text ,password , hidden , checkbox, radio, submit, reset。text 创建文本框，让访问者输入文本信息；password 用来输入密码，用户的输入在浏览器上用*代替；hidden 用来存放用户不可见的信息；checkbox 类似 ON/OFF 开关；radio 实现多选一；submit 用来提交表单；reset 可以重新生成完整的表单。

<select>使访问者可以从一个下拉菜单或滚动菜单中选取某一项。<select></select>括住菜单体，中间的内容由<Option>标签设定。例如：

```
<Form>
<select name=”month”>
<option value=”1”>Jan
<option value=”2”>Feb
.............
<option value=”12”>Dec
</select>
</Form>
```

若你想让访问者输入较大量的内容，可以通过使用<textarea>创建文本框，可以自行定义文本框的大小和宽度。例如：

```
< textarea name=”comments” cols=10 rows=40>
this is test!!
< /textarea >
```

掌握了以上所讲的内容，你就可以建立 HTML 表单了。

ColdFusion 表单

ColdFusion 也提供一套与之对应的表单标志：

HTML 标记	ColdFusion 标记
<Form>	CFForm
<input>	CFInput
<select>	CFSelect
<textarea>	

CFForm 可以用来建立表格，但它的功能比 HTML 的 Form 要强。它所包括的标签有：

- CFINPUT – 建立表单输入项 (类型有：radio button, text box, or checkbox) 并可以判断输入是否合乎限制条件。
- CFSELECT – 建立下拉式菜单
- CFSLIDER – 建立滑动条
- CFTEXTINPUT – 建立文本输入框
- CFTREE – 建立树型结构控制
- CFGRID – 建立栅格控制来显示表格数据
- CFAPPLET – 在表格中插入已经注册的 Java applet。Applets 的注册在 ColdFusion 的 Administrator 中。

CFFORM 的语法是：

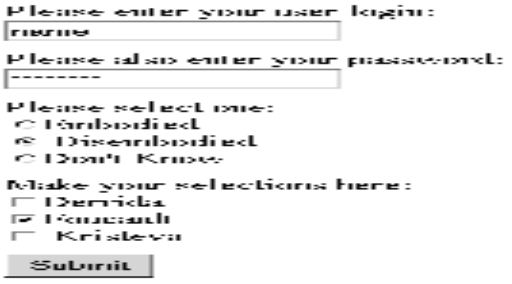
```
<CFFORM  NAME="name"  //FORM 的名称
         ACTION="form_action" //调用的“处理页”路径和名称
```

```
            ENABLECAB="Yes/No" //是否下载微软的 CAB 文件
            ONSUBMIT="javascript" //在表格提交前调用的 JavaScript
            TARGET="window_name"//表格输出时所在窗口或窗口中某一帧的名称
            ENCTYPE="type"//MIME 类型>
...
</CFFORM>
```

下面着重介绍使用 CFINPUT 建立输入项。通过一个例子来说明 CFINPUT 建立的各种类型的输入项：

```
<CFFORM NAME="Form1" ACTION="submit.cfm" METHOD="Post">
//建立表单，提交的数据由 submit.cfm 来处理
<TABLE CELLPADDING=5 border=0>
    <TR><TD>Please enter your user login:<BR>
    <CFINPUT TYPE="text" NAME="loginID" VALUE="name"></TD></TR>
//建立文本框
    <TR><TD>Please also enter your password:<BR>
    <CFINPUT TYPE="password" NAME="pwd" VALUE="password"></TD></TR>
//建立密码输入框
    <TR><TD>Please select one:<BR>
    <CFINPUT TYPE="radio" NAME="radio1" VALUE="select1">Embodied<BR>
    <CFINPUT TYPE="radio" NAME="radio1" CHECKED="yes" VALUE="select2">
    Disembodied<BR>
    <CFINPUT TYPE="radio" NAME="radio1" VALUE="select3">Don't
    Know</TD></TR>
//建立多选一的选择项
    <TR><TD>Make your selections here:<BR>
    <CFINPUT TYPE="checkbox" NAME="checkbox1" VALUE="one">Derrida<BR>
    <CFINPUT TYPE="checkbox" NAME="checkbox1" CHECKED="yes"
    VALUE="two">Foucault<BR>
    <CFINPUT TYPE="checkbox" NAME="checkbox1" VALUE="three">
    Kristeva</TD></TR>
//建立 ON/OFF 选项
    <TR><TD><INPUT TYPE="Submit" VALUE="Submit"></TD></TR>
//建立提交按钮
</TABLE>
</CFFORM>
```

建立的表单如下：



传递表单变量

使用表格传递变量可以有多种方法，下面介绍如何通过 input 的 Hidden 类型传递变量和在 Action 页中如何引用变量。

通过例子讲解如何通过 input 的 Hidden 类型传递变量：

```
<FORM ACTION="example.cfm" METHOD="Post">
        <INPUT TYPE="Hidden"  NAME="Customer_ID"  VALUE="24">
// VALUE 值也可以是变量 VALUE="#Customer_ID#",在处理页中,可以通过        引
  用 Form.Customer_ID 变量来得到 Customer_ID 的值。
        <INPUT TYPE="Submit" VALUE="Enter">
</FORM>
```

这样在表单提交后，变量 Customer_ID 的值 24 也传递到了服务器端，在 example.cfm 处理页中可以直接引用 Form.Customer_ID 这个变量名称得到传来的值。

Form 表单中的各种类型的提交值都是这样传递的,在 Action 页中直接引用 Form.NAME 这个变量名称得到传来的值。

以往用 CGI 处理提交内容，提交内容存储在 CGI 环境变量中，程序人员要自行从 CGI 环境变量中提取出提交的内容，这样处理十分麻烦。在 ColdFusion 中，你不必自行处理提交的变量，只要直接引用他们就可以了，在 ColdFusion 的 Form 中，所有的 Tag 都有 NAME 属性，这就可以在处理页中通过直接引用 Form. NAME 来得到提交的相应的数据。

练习题：

用 HTMLTag 建立表单,再用 ColdFusion 建立同样内容的表单，观看这两个表单有没有不同。

Unit 7：建立检索界面

动态使用 Select 控制

使用 Select 编写下拉选菜单，其供选择的内容可以事先知道自己填写，例如，员工信息填写表中性别一项分为男、女，可以用 Select 编写下拉选菜单，其中只有两项可供选择。有时，Select 编写的下拉选菜单的内容是你事先并不知道的，这时候要动态地使用 Select 控制。

先讲一个例子，例子中 Select 的内容是从数据库中读出，动态生成的：

```
<CFQUERY NAME="ParkNames" DATASOURCE="ParkDB">
    SELECT ParkName_ID, ParkName      FROM Parks
</CFQUERY>
<FORM ACTION="example.cfm" METHOD="Post">
    <SELECT NAME="ParkName_ID">
    <CFOUTPUT QUERY="ParkNames">
        <OPTION VALUE="#ParkName_ID#">#ParkName#
    </CFOUTPUT>
    </SELECT>
    <INPUT TYPE="submit" VALUE="Submit">
</FORM>
```

在此页生成之前，用户不知道有哪些 ParkName 可供选择，这些 ParkName 是动态产生的。

创建动态查询

动态查询是指事先没有完整的 SQL 语句，而是动态地决定执行哪一条查询条件。动态查询要借助 ColdFusion 的 CFIF/CFELSEIF/CFELSE tag 来控制 SQL 的结构。大致的结构如下：

```
<CFQUERY NAME="queryname" DATASOURCE="datasourcename">
   ...Base SQL statement
<CFIF value operator value >...additional SQL</CFIF>
</CFQUERY>
```

下面给出一个具体的例子：

```
<CFQUERY NAME="GetParkList" DATASOURCE="CF 4.0 Examples">
      SELECT * FROM Parks WHERE 0=0
<CFIF #ParkName# is not ""> AND ParkName LIKE '%#ParkName#%'
</CFIF>
<CFIF #ParkType# is not "AllTypes"> AND ParkType = '#ParkType#'
</CFIF>
<CFIF #Region# is not "AllRegions"> AND Region = '#Region#'
</CFIF>
<CFIF #State# is not ""> AND State = '#State#'
</CFIF>
</CFQUERY>
```

WHERE 0=0 是为了在四条 IF 语句都不符合时，SQL 语句不至于出现语法错误。

测试多重条件

ColdFusion 可以支持 SELECT 的多重选择(如同 HTML input 类型中 SELECT 带有 MULTIPLE 属性) 。被选择的多个值并排排列，使用“ , ”来分割。例如有一个多重选择是：red, green, 和 blue。用户选择了 red 和 green ，在表单提交时'red', 'green'被传送到“ 下一页 ”。

例如，假如你希望用户从一个 SELECT 中进行多重选择，可供多重选择的内容是从数据库中读出的。

选择一个以上的公司:

```
<SELECT Name="SelectOrgs" MULTIPLE>
    <OPTION VALUE="5">Mobil Corporation
    <OPTION VALUE="19">ShapeWare, Inc.
    <OPTION VALUE="13">BankBoston
</SELECT>
```

如果用户选择了 Shapeware 和 BankBoston 公司,这个 SelectedOrgs 得值将会是 19,13

如果这个选择的参数用于一个 SQL 语句:

```
SELECT * FROM Organizations
WHERE Organization_ID IN (#SelectedOrgs#)
```

那么这条语句被送到数据库时是：

```
SELECT * FROM Organizations
WHERE Organization_ID IN (19,13)
```

练习题：

利用第四天建立的数据库，将前一天建立的表单中的 Select 项改为动态 Select。

Unit 8：建立用户菜单界面

使用 CFTREE 可以把数据库的内容以树状结构列出，可以在树中只列出唯一标识记录的信息，每一条信息都有超级链接，链接到记录的详细信息页。

传递 URL 参数

有时候，你希望建立一个参数，多个文件都可以使用它，这就涉及到参数如何传递，有多种方法可以传递参数：使用 URL，使用 form，或使用浏览器 cookies 变量和客户端变量。下面主要介绍如何使用 URL 传递参数。

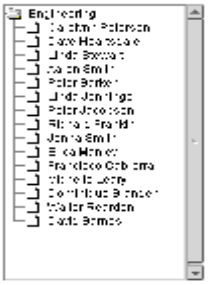
利用 URL 传递参数是在 URL 后面加上？在加上要传递的参数：<A HREF="example.cfm?user_id=5>，在此例中把 user_id=5 传递到 example.cfm 文件中，在 example.cfm 中你可以使用 URL.参数名称来引用它。例如：<CFOUTPUT> Your user ID is #URL.user_id# </CFOUTPUT>。使用 URL 传递参数十分直观，在浏览器的地址栏中，你可以看到 URL 和它后面跟着的参数。

建立用户菜单界面

使用 CFTREE，你可以将数据库查询的结果使用树状结构表示。CFTREEITEM 是用来列出树中内容：

```
//先查询数据库
<CFQUERY NAME="Engineering" DATASOURCE="cfsnippets">
    SELECT FirstName + ' ' + LastName AS FullName
    FROM EMPLOYEES
</CFQUERY>
//将查询结果以树状结构列出
<CFFORM NAME="form1" ACTION="submit.cfm"
    METHOD="Post">
<CFTREE NAME="tree1" REQUIRED="yes" HSCROLL="no" vSCROLL="yes">
   <CFTREEITEM VALUE=FullName QUERY="Engineering" QUERYASROOT="yes"
     IMG="folder,document">
</CFTREE>
</CFFORM>
```

结果如下：



树中每一个分支可以是一个超级连接，将 URL 和 CFTREE 相结合：

```
<CFFORM ACTION="submit.cfm">
    <CFTREE NAME="oak" HIGHLIGHTHREF="yes" HEIGHT="100"
        WIDTH="200" HSPACE="100" VSPACE="6" HSCROLL="no"
       VSCROLL="no" BORDER="no" DELIMITER="?">
        <CFTREEITEM VALUE="Important Links">
        <CFTREEITEM VALUE="Allaire Home" PARENT="Important Links"
            IMG="document" HREF="http://www.allaire.com">
        <CFTREEITEM VALUE="Allaire Forums" PARENT="Important Links"
            IMG="document" HREF="http://forums.allaire.com">
     </CFTREE>
</CFFORM>
```

结果如下：

这样可以建立一个数据库菜单界面，使用 CFTREE 可以建立各种结构的“树”，方式灵活，形式各异，CFTREE 的具体使用在此不做过多的解释，想要知道更多的信息，可以查看再线帮助。

练习题：

利用 CFTREE 将第四天建立的数据库输出。

Unit 9：数据添加

本节主要讲解如何将数据添加到数据库中。

你可以使用 CFFORM 标记或使用标准的 HTML 表格标记建立表单，收集用户要插入到数据库的信息。当表格提交后，表格信息传递到处理页，由处理页执行插入操作，插入操作可由 CFINSERT 或 CFQUERY 加上 SQL 语句来实现，最后处理页要给用户发回确认信息。所以一般的数据添加都有两个 CFM 文件：

- 建立添加表格
- 建立添加处理页

建立数据添加表单

当使用 HTML 建立添加表单时，要定义 ACTION 和 METHOD 两个属性，通常 METHOD 为 POST，ACTION 可以指向一个以.CFM 为后缀的 ColdFusion 文件。例如：<FORM ACTION="insdata.cfm" METHOD="Post">。

依照要添入到数据库中的各项建立表单项，表单项的名称属性值要和数据库中的数据项名称相同。例如，有一个存贮雇员信息的数据库，ODBC 数据源的名称是 Employee DB，其中有一个表叫 Employees，存储雇员的姓氏（FirstName）、名字（LastName）和电话号码（Phone）三项信息，下面建立添加雇员的表单：

```
<FORM ACTION="insdata.cfm" METHOD="Post">
    <!-- Data entry fields -->
<PRE>
    First Name: <INPUT TYPE="text" NAME="FirstName">
    Last Name: <INPUT TYPE="text" NAME="LastName">
    Phone: <INPUT TYPE="text" NAME="Phone">
    <INPUT TYPE="Submit" VALUE="Enter Information">
</PRE>
</FORM>
```

When the Submit button is clicked, the form action is carried out, and all inputs (including hidden inputs) are made available to the next page.

当提交键按下后，insdata.cfm 文件运行，输入的内容在 insdata.cfm 中全部有效，可以使用。

建立添加 Action 页

ColdFusion 有两个 Tag 可以用来更新数据库，CFINERT 和 CFQUERY。它们一个简单易用，一个功能较强。下面分别介绍这两个 Tag。

使用 CFINSERT 建立更新处理页。

先讲解 CFINSERT 的语法：

```
<CFINSERT DATASOURCE="ds_name" //数据源名称
    DBTYPE="type"//数据源类型，可以是 ODBC、Oracle73、Oracle80、Sybase11
    DBSERVER="dbms"//
    DBNAME="database name"//Sybase 11 驱动器专用
    TABLENAME="tbl_name"//要添加的数据库中表格名称
    TABLEOWNER="owner"
    TABLEQUALIFIER="tbl_qualifier"
    USERNAME="username"
    PASSWORD="password"
    PROVIDER="COMProvider" //OLE_DB 专用
    PROVIDERDSN="datasource" // OLE_DB 专用
    FORMFIELDS="formfield1, formfield2, ...">
```

在多数情况下，用到的 CFINSERT 属性很少。

下面的例子是添加雇员信息的处理的文件 insdata.cfm：

```
<!--- Inserts the data from the the HTML Form --->
```

```
<CFINSERT DATASOURCE="Employee DB" TABLENAME="Employees">
<HTML>
<HEAD><TITLE>Input Form</TITLE></HEAD>
<BODY>
<CENTER><H2>Thank You!</H2></CENTER>
<HR>
<P>此雇员的信息已正确添加到了数据库中 !!! </P>
<HR>
</BODY>
</HTML>
```

使用 CFQUERY 建立添加处理页。

要处理复杂的添加处理可以在 CFQUERY 标记中使用 SQL 的插入语句，SQL 的插入语句使用十分灵活，它的语法：

```
INSERT INTO tablename (columnnames) VALUES (values)
```

还以登记雇员信息为例：

```
<CFQUERY NAME="AddEmployee" DATASOURCE="Employee DB">
    INSERT INTO Employees (FirstName, LastName, Phone)
    VALUES ('#Form.FirstName#', '#Form.LastName#', '#Form.Phone#')
</CFQUERY>
```

练习题：

根据第四天的数据库建立一个添加界面和处理页。

Unit 10：数据库记录更新

要更新数据库里的记录通常要建立两个步骤：

- 建立更新表格：使用 CFFORM 标记或 HTML 表格标记来建立更新表格，提交是调用更新处理页。
- 建立更新处理页：使用 CFUPDATE 或 CFQUERY 包含有 SQL UPDATE 语句来处理记录的更新，并为用户返回更新完成的提示信息。

记录更新页面

要更新记录首先要选择哪条记录要更新，所以更新页面和插入页面有所不同，更新页面要先列出数据库中记录的唯一标识记录的信息，由用户选择要更新的记录，再列出此记录的全部信息供用户更新。所以在更新页面中首先要查询数据库，找到要更新的哪条记录，再在<CFOUTPUT>中列出记录的内容。下面举例说明：

使用 http://web_root/updateform.cfm?employeeid=2 ，调用更新数据库中 employeeid=2 的记录，生成记录更新页面。

```
<CFQUERY NAME="EmployeeRecord" DATASOURCE="Employee DB">
        SELECT *  FROM Employees
        WHERE Employee_ID = #URL.EmployeeID#
</CFQUERY>
```

```
<HTML>
<HEAD><TITLE>Update Form</TITLE></HEAD>
<BODY>
<CFOUTPUT QUERY="EmployeeRecord">
<!--- Input form --->
<FORM ACTION="EmployeeUpdate.cfm" METHOD="Post">
<!--- 列出记录的内容 --->
<INPUT TYPE="Hidden" NAME="Employee_ID"
    VALUE="#Employee_ID#">
<PRE>
    FirstName: <INPUT TYPE="Text" NAME="FirstName" VALUE="#FirstName#">
//在输入框中列出此条记录的 FirstName，因为 VALUE="#FirstName#"，这样可以完成表单值预制。
    LastName: <INPUT TYPE="Text" NAME="LastName" VALUE="#LastName#">
    Phone: <INPUT TYPE="Text" NAME="Phone" VALUE="#Phone#"><BR>
<INPUT TYPE="Submit" VALUE="Update Information">
</PRE>
</FORM>
</CFOUTPUT>
</BODY>
</HTML>
```

如果此条记录的 FirstName 是 zhang，LastName 是 ping，Phone 是 62301111，那么记录更新页面就象下图一样，表单的值是预制的：

建立更新数据库的 Action 页

ColdFusion 有两个 Tag 可以用来更新数据库，CFUPDATE 和 CFQUERY。下面分别介绍这两个 Tag。

使用 CFUPDATE 建立更新处理页。

使用 CFUPDATE 处理更新十分简单，

CFUPDATE tag 的语法规则：

```
<CFUPDATE  DATASOURCE="ds_name"  //数据源名称
             DBTYPE="type"
             DBSERVER="dbms"
             DBNAME="database name"
             TABLENAME="table_name"  //数据源中的表格名称
             TABLEOWNER="name"
             TABLEQUALIFIER="qualifier"
```

```
                USERNAME="username"
                PASSWORD="password"
                PROVIDER="COMProvider"
                PROVIDERDSN="datasource"
                FORMFIELDS="field_names"  //列出要更新的数据项，如果没有填写
                                          此项则更新全部数据项>
```

下面的 EmployeeUpdate.cfm 例子是对上面记录更新页面的处理，使用 CFUPDATE，并给出处理完成的提示信息：

```
<CFUPDATE DATASOURCE="Employee DB" TABLENAME="Employees">
<HTML>
<HEAD><TITLE>Reply</TITLE></HEAD>
<BODY>
<H2>谢谢!</H2>
<HR>
<P>你数据库中的记录已经正确更新!</P>
<HR>
</BODY>
</HTML>
```

使用 CFQUERY 建立更新处理页。

使用 CFUPDATE 处理更新虽然简单，但它不能完成复杂的操作，你可以使用 CFQUERY 结合 SQL 语句来实现复杂的更新处理。标准的 SQL 更新语句的语法是：

```
UPDATE tablename SET columnname = value WHERE condition
```

我们的例子还是处理更新页面中的雇员更新：

```
<CFQUERY NAME="UpdateEmployee" DATASOURCE="Employee DB">
       UPDATE Employees
        SET Firstname='#Form.Firstname#', // Form.Firstname 是从上面更新页
                                          面中由   用户填入，提交后传送来
                                          的值
            LastName='#Form.LastName#',
            Phone='#Form.Phone#'
    WHERE Employee_ID=#Employee_ID#
</CFQUERY>
```

客户端/服务器端数据验证

当你要求用户输入数据时，总希望用户的输入能满足你的要求，如输入的名字中不要带有阿拉伯数字；当你想要把用户的输入添加到数据库中时，也要求数据符合数据库中具体项的类型设定，例如 String 类型的项中不能填入 Integer。所以，下面我们来介绍一下十分有用的数据验证技术，数据验证可以在客户端进行，也可以在服务器端进行，两者各有利弊，下面使用表格对比两者的区别：

验证类型	描述
客户端数据验证	在 CFFORM 表格中，你可以在 CFINPUT，CFGRID，CFSLIDER，CFTEXTINPUT、CFTREE 的 ONVALIDATE 的属性中添加一段 JavaScript 程序来判断输入的数据是否符合

	要求。这种判断实在提交之前进行的所以叫做客户端数据验证
服务器端数据验证	在 CFFORM 表格中,In a CFFORM，你可以在 CFINPUT，CFTEXTINPUT 中的 VALIDATE 属性中规定输入数据的类型可以是 date、eurodate 、 time 、 float 、 integer 、 telephone 、 zipcode 、 creditcard 、 social_security_number。这种判断是和 CFM 文件一起在服务器端进行的。

下面我们以 CFFORM 来说明和认证用关的几个属性的使用：

```
<CFINPUT TYPE ="input_type"
    NAME ="name"
    VALUE ="initial_value"
    REQUIRED ="Yes/No"
    RANGE ="min_value，max_value" //当要求输入为数字时，使用此属性规定数字
                                    的范围
    VALIDATE ="data_type" //规定输入数据的类型
    ONVALIDATE ="javascript_function" //定义数据判断的 javascript 函数名称，
                一旦定义了 ONVALIDATE，VALIDATE 定义的数据类型判断将失效
    MESSAGE ="validation_msg" //一旦输入类型不对，出示错误信息
    ONERROR ="text" //输入类型不对时调用的 javascript 函数名称
    SIZE ="integer" //规定除 Radio 和 Checkbox 类型以外其他输入的大小，例如
                允许输入五个字符
    MAXLENGTH ="integer"//当输入类型是 TEXT 时，允许输入的最多字符数
    CHECKED ="Yes/No">
```

总之 CFINPUT 比 HTML 的 Input 功能要强大许多，可以对输入进行多种限制，而不用编写 JavaScript，又可以引入 JavaScript 进行更复杂的判断。

使用 Javascript 判断输入的合法性

CFINPUT、CFGRID、CFSLIDER 、CFTEXTINPUT、CFTREE 带有 ONVALIDATE 属性，可以添加 JavaScript。一旦输入不符合要求时，可以通过 ONERROR 属性添加进行相应处理的 JavaScript，CFGRID 、CFINPUT、CFSELECT、CFSLIDER、CFTEXTINPUT、CFTREE 带有 ONERROR 属性。

下面举一个例子，使用 JavaScript 判断用户是否输入电子邮件地址和输入的电子邮件地址是否符合规范，指在邮件地址中要有“@”字符：

```
<HTML>
<HEAD><TITLE>JavaScript Validation</TITLE>
<SCRIPT><!--
function testbox(form) {
    Ctrl = form.inputbox1;
    if (Ctrl.value == "" || Ctrl.value.indexOf ('@', 0) == -1) {
    return (false);
    } else
```

```
        return (true);
} //-->
</SCRIPT>
</HEAD>
<BODY><H2>JavaScript validation test</H2>
<P>Please enter your email address:</P>
<CFFORM NAME="UpdateForm" ACTION="update.cfm" >
    <CFINPUT TYPE="text" NAME="inputbox1" REQUIRED="YES"
    ONVALIDATE="testbox" MESSAGE="请正确输入电子邮件地址!!"
    SIZE="10" MAXLENGTH="10">
    <INPUT TYPE="Submit" VALUE=" Update... ">
</CFFORM>
</BODY>
</HTML>
```

练习题：

根据第四天的数据库，建立一个更新界面和处理页，并加上输入验证，使用客户端和服务器两种分别验证。

Unit 11：安全的界面

安全性十分重要，因为将 WEB 服务器放在 Internet 上意味着任何人都可能访问这个服务器。敏感的信息必须受到保护以防止无权访问它们的人查看，同时对有权限或被允许的人开放。所以我们要编写安全的界面，对进入界面的用户进行必要的认证。ColdFusion 提供了多种安全机制，可以利用它们编写安全的界面。

使用应用级变量

在讲应用级变量前，先讲一下 Cold Fusion 的 WEB 应用程序的框架。Cold Fusion 的应用是由多个应用程序组成的，在 Cold Fusion 的应用框架中有一个文件叫做：Application.cfm，在此文件中可以定义：

- 应用的名称
- Application 和 session 变量
- 定制错误页 Custom error pages
- 数据源

等等。当任何一个 Cold Fusion 的应用程序被调用时，Cold Fusion 都先要查找 Application.cfm 这个文件，一旦这个文件被找到，先执行这个文件。正因为 Cold Fusion 的这特性,所以可以利用这个文件来进行用户认证。就象进入 Cold Fusion 的 Administrator 一样，在输入的密码口令正确后才进入。

应用级变量是永久性变量，存储在服务器上的，可以指定失效日期和时间间隔。应用级变量只有使用 Application.MyVariable 引用后才被击活。它对个别指定的应用和所有访问 ColdFusion 服务器的应用有效。

使用 Cookies 变量

cookies 是客户机上存储的一个数据项，它由服务器软件产生并作为 HTTP 响应的一部分发送给浏览器，然后浏览器将其保存在一个本地文件中，服务器 cookies 可以存储在客户机上，用户也可以配置其浏览器不接受 cookies。当浏览器产生一个 HTTP 请求时，便会检查存储在客户机上的 cookies。Cookies 允许 WEB 应用程序在客户机上存储信息，并在后来得到该信息，这点十分有用。

CFCOOKIE 可以用来定义 cookies 变量，包括存在期限和安全问题。语法如下：

```
<CFCOOKIE  NAME="cookie_name" //COOKIE 名称
           VALUE="text"// COOKIE 值
           EXPIRES="period" //存在期限
           SECURE="Yes/No" // 指出变量发送时是否加密，如果浏览器不支
                              持安全套接层(SSL)，cookie 将不传送
           PATH="urls" // Cookie 有效的领域的 URL
           DOMAIN=".domain" //指定 Cookie 有效的领域，例如.allaire.com>
```

如果你没有给 Cookies 变量设定存在期限，那么这个 Cookies 变量只有在浏览器打开时存在，浏览器关闭时它就不存在了。CFCOOKIE 定义的 Cookies 变量在浏览器关闭后不存储到 cookies.txt 中。

下面的例子建立一个变量叫 Cookie.User_ID 值是 2344，存在期限 100 天

```
<CFCOOKIE NAME="User_ID" VALUE="2344" EXPIRES="100">
```

要删除 Cookies 变量，将 EXPIRES 设为 now，这个 Cookies 变量将在用户关闭浏览器时删除。

在用户认证中，我们可以使用 Cookies 来存放用户名称和密码，规定密码使用的有效日期。并将这些 Cookies 定义在 Application.cfm 文件中，对用户进行认证。

使用 Session 变量

如果要为一次访问或一系列请求建立属于它自己的变量，使用 Session 变量，例如你可以使用 Session 变量存储用户网上购物车中的商品。

Session 变量开始于用户的第一个连接，结束于最后一个连接。每个用户的 Session 变量与用户本身是一一对应的，Session 不在网上传递，Session 变量和应用级变量一样，存储在服务器内存中的，每个用户有一个 ClientID，服务器通过 ClientID 来识别用户并对应其 Session 变量，由于网络的无界性，服务器无法知道用户何时离开，所以 Session 变量可以设有超时。

例子：在线商店的 Application.cfm 文件中，定义购物车

```
<CFAPPLICATION          NAME="CF40_STORE"          SESSIONMANAGEMENT="Yes"
                    SESSIONTIMEOUT="#CreateTimeSpan(0,1,30,0)#">
<!--- 建立手推车的 Session 变量 --->
<CFPARAM NAME="Session.StoreItems" DEFAULT="">
<CFPARAM NAME="Session.StoreQuantities" DEFAULT="">
```

注册信息可以放在 Cookie 变量中,也可以放在 Session 变量中,使用 Cookie 变量时,Cookie 值会在网络中来回传送，可能会引起安全方面的问题，使用 Session 变量就没有这种问题；但 Session 变量只在用户访问同一网站时有效，如果一段时间内不访问该网站 Session 变量自动清除。因为有此特性,所以使用 Session 变量存储在线商场的购物变量比使用 Cookie 变量更加安全有效；在处理 Login 变量时，Session 变量和 Cookie 变量各有利弊。

练习题：

自己建立一个 Login 页。