Использование пакета Cold Fusion для MS Windows при построении WWW - интерфейсов к базам данных.

5.1Введение

5.2Установка Cold Fusion

5.3Администрирование Cold Fusion

5.4Взаимодействие Cold Fusion с базами данных

5.5Передача параметров в DBML - шаблон

5.6Занесение и модификация данных с использованием тегов DBINSERT и DBUPDATE

5.7Выполнение запросов к базам данных

5.8Использование результатов запроса для динамического создания HTML - документа

5.9Вывод результата выполнения запроса в виде таблицы

5.10Дополнительные замечания по созданию DBML - шаблонов

5.11Использование параметров и переменных в шаблонах
5.11.1Поля формы и параметры URL
5.11.2Переменные окружения CGI
5.11.3Применение тега DBSET для создания переменных
5.11.4HTTP Cookies
5.11.5Использование результатов выполнения запросов

5.12Проверка корректности данных и форматирование вывода
5.12.1Проверка корректности данных в полях формы
5.12.2Функции вывода в DBML

5.13Динамическое изменение содержимого документа
5.13.1Условный оператор (DBIF & DBELSE)
5.13.2Перенаправление на другой URL (DBLOCATION & DBABORT)
5.13.3Включение в шаблон других шаблонов
5.13.4Определение типа данных MIME для содержимого документа

5.14Расширенные возможности
5.14.1Динамическое определение SQL выражения
5.14.2Поддержка транзакций
5.14.3Вложенные области вывода и группирования
5.14.4Использование списочных полей с множественным выбором
5.14.5Дополнительные команды SQL

*При подготовке данной главы отчета использовались материалы " Cold Fusion User Guide " (Руководство пользователя пакета Cold Fusion).*

5.1 Введение

Пакет Cold Fusion фирмы Allaire - это средство для быстрой разработки интерактивных, динамических документов для Web основанное на обработке информации из баз данных, в основе которого лежит следующий набор технологий:

* HTML (Hyper-Text Markup Language)
* CGI (Common Gateway Interface)
* SQL (Structured Query Language)
* ODBC (Open Database Connectivity)

Разработка приложений с использованием Cold Fusion не требует программирования на таких языках как Perl, C/C++, Visual Basic или Delphi. Вместо этого вы создаете приложение, встраивая в обычный (стандартный) HTML файл специальные теги для работы с базами данных.

В данной главе рассматривается Cold Fusion версии 1.5 .

5.2 Установка Cold Fusion

Cold Fusion запускается как CGI приложение на различных Web-серверах под Windows NT и Windows 95 и должен быть совместим с любым сервером поддерживающим CGI.

Cold Fusion тестировался на совместимость со следующими серверами:

* O'Reilly WebSite
* Microsoft Internet Server
* EMWAC HTTPS
* Process Software Purveyor
* Netscape Communications/Commerce Server
* Internet Factory Communications/Commerce Builder
* Spry Safety Web Server
* CSM Alibaba

Для связи с различными СУБД Cold Fusion использует 32-разрядные ODBC - драйвера. Для корректной работы с Cold Fusion ODBC - драйвер должен удовлетворять следующим требованиям:

* Это должен быть 32-разрядный драйвер.
* Он должен поддерживать Уровень 1 ODBC API.
* Должна поддерживаться базовая грамматика SQL.
* Для совместимости с функциями ввода даты/времени, драйвер должен поддерживать соответствующие типы данных.

Для установки и использования Cold Fusion система должна удовлетворять следующим требованиям:

* Операционная система Windows NT 3.51(или выше) или Windows 95.
* Микропроцессор 80386 или выше; рекомендуется Pentium.
* 10Mb свободного дискового пространства.
* 24Mb RAM для Windows NT или 16Mb для Windows 95
* Установленная сетевая поддержка TCP/IP.
* Установленный WWW сервер.

Для установки Cold Fusion нужно запустить программу SETUP.EXE, которая должна находится на инсталляционном диске 1.

Помимо копирования файлов, необходимых для работы Cold Fusion, в процессе установки, в корневой директории с документами Web сервера создается директория с именем CFPRO. Эта директория содержит:

* Тест, для проверки правильности установки системы.
* Мини-учебник, в формате HTML, помогающий освоить азы Cold Fusion.
* Примеры приложений, демонстрирующие различные способы использования Cold Fusion.

Чтобы проверить правильность установки Cold Fusion, нужно открыть документ, URL до которого имеет вид http://myserver/cfpro/getstart.htm, где myserver - имя или IP адрес вашего Web сервера.

5.3 Администрирование Cold Fusion

Для администрирования в Cold Fusion предусмотрен специальный интерфейс администратора. Этот интерфейс позволяет изменять различные параметры настройки Cold Fusion по четырем категориям:

* **Data Sources** - используется для настройки источников данных ODBC, для использования их с Cold Fusion. Чтобы добавить источник данных, нужно нажать на кнопку "Add...", выбрать один из установленных в системе драйверов ODBC и задать определенные для него настройки. Переопределить эти настройки можно воспользовавшись кнопкой "Setup...", предварительно выделив конкретный источник данных. Если нужно определить способы взаимодействия Cold Fusion с источником данных, то нужно нажать на кнопку "Preferences..." и определить такие атрибуты как имя пользователя, пароль, допустимые операции с базой данных и др.
* **Templates** - используется для настройки логических путей до директорий в которых расположены файлы с шаблонами Cold Fusion.
* **Debugging** - используется для настройки отладочных сообщений и сообщений об ошибках.
* **Mail** - используется для настройки параметров SMTP mail и позволяет просматривать журнал сообщений и ошибок.

5.4 Взаимодействие Cold Fusion с базами данных

Cold Fusion позволяет динамически генерировать HTML документы основанные на запросах пользователя. Эти запросы передаются в Cold Fusion CGI - скрипт (DBML.EXE), который пересылает данные в Cold Fusion Engine обрабатывающий эти данные в соответствии с заданным шаблоном, выполняя необходимые запросы и генерируя HTML документ, который отправляется пользователю.

Основой динамического создания документов являются специальные теги, входящие в язык разметки DBML, ориентированные на работу с базами данных. Почти все основные возможности Cold Fusion сосредоточены в четырех тегах:

* DBQUERY - выполнение SQL - запроса к базе данных;
* DBINSERT & DBUPDATE - создание и модификация записей в базе данных;
* DBOUTPUT - отображение результата запроса, допускающее его произвольное размещение среди HTML - тегов.

Шаблон, на основе которого генерируется HTML - документ, представляет собой комбинацию тегов HTML и DBML:

* HTML - теги используются для форматирования как постоянной части документа, так и результатов запросов. Например, можно определить полужирный шрифт для каждого поля и разделительные линии между записями.
* DBML - теги используются для формирования запроса к базе данных, а также определяют где и как будут отображены результаты запросов.

На рисунке 5-1 показывается, как Cold Fusion обрабатывает запрос, полученный от клиента:

Рисунок 5-1

1. Когда пользователь нажимает кнопку типа "Submit" в форме или выбирает гипертекстную ссылку в документе, Web - браузер отправляет запрос на Web - сервер.
2. Web - сервер, если в запросе указан DBML - шаблон, запускает процесс Cold Fusion, отправляя ему данные полученные от клиента.
3. Cold Fusion принимает данные полученные от клиента обрабатывает теги DBML в шаблоне, включая подготовку запроса к базе данных и форматирование, которое будет использоваться в результирующем документе.
4. Cold Fusion взаимодействует с базой данных используя ODBC.
5. Cold Fusion динамически генерирует HTML - документ содержащий результат выполнения запросов к базе данных и возвращает его Web - серверу. Cold Fusion может также динамически генерировать почтовое сообщение и отправлять его через почтовый SMTP - сервер.
6. Web - сервер возвращает сгенерированный HTML - документ Web - клиенту.

5.5 Передача параметров в DBML-шаблон

Существует несколько способов передачи параметров между шаблонами. Можно передавать параметры непосредственно в URL, использовать для этого форму либо cookie.

Если параметры передаются через URL, то они добавляются к адресу вызываемого шаблона через символ "&" (амперсант) в виде *параметр = значение*. Например, гипертекстовая ссылка, приведенная ниже, отправляет параметр с именем 'user\_id' и значением 5 в шаблон 'example.dbm':

<A HREF="cgi-shl/dbml.exe?Template=example.dbm&user\_id=5">

При передаче параметров через форму используются поля формы, которые должны иметь имена, совпадающие с именами параметров, которые требуется передать. Ниже приведен пример передачи параметра, из предыдущего примера используя форму:

<FORM ACTION="cgi-shl/dbml.exe?Template=example.dbm">
<INPUT TYPE="HIDDEN" NAME="user\_id" VALUE="5">
<INPUT TYPE="SUBMIT" VALUE="Enter">
</FORM>

Заметим, что при обращении к CGI - программе DBML.EXE должен быть определен стандартный параметр Template, указывающий на конкретный шаблон.

Переменные, занесенные в cookie браузера и переменные окружения CGI доступны в любом шаблоне. Способы занесения информации в cookie описаны в [п.5.11](file:///C%3A%5CD%3A%5C%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%5C%D1%81%D1%82%D0%B0%D1%82%D1%8C%D0%B8_%D0%B8_%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA%D0%B8%5C%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B8_WWW_%D0%B4%D0%BB%D1%8F_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0_%D0%BA_%D0%91%D0%94%5C#5_11).

5.6 Занесение и модификация данных с использованием тегов DBINSERT и DBUPDATE

При использовании тегов DBINSERT и DBUPDATE для занесения или модификации данных, параметры должны быть переданы в шаблон обязательно из формы, используя метод POST.

Для создания новой записи в базе данных используется тег DBINSERT, а для модификации существующей записи используется тег DBUPDATE. При использовании этих тегов необходимо определить атрибуты DATASOURCE и TABLENAME. DATASOURCE это название источника данных ODBC, содержащего редактируемую таблицу, а TABLENAME - имя этой таблицы.

Например, если источник данных ODBC называется 'Person DB', а таблица, в которой требуется создать запись - 'Person', то тег DBINSERT в шаблоне будет иметь следующий вид:

<DBINSERT DATASOURCE="Person DB" TABLENAME="Person">

Параметры, переданные в шаблон должны совпадать с именами полей таблицы, в которой создается (модифицируется) запись. В том случае, если не все передаваемые параметры должны участвовать в этой процедуре, используется атрибут FORMFIELDS, в котором через запятую перечисляются имена полей таблицы, для которых должны существовать одноименные параметры.

Особо следует отметить, что для того, чтобы использовать тег DBUPDATE для модификации записи, в таблице должен быть определен первичный ключ, а для всех полей включенных в первичный ключ из обрабатываемой HTML-формы обязательно должны быть переданы соответствующие параметры.

Теги DBINSERT и DBUPDATE могут иметь также еще два необязательных атрибута:

* TABLEOWNER - для тех источников данных которые поддерживают права собственности на таблицы (например, SQL Server, Oracle и др.), этот атрибут можно использовать, чтобы указать собственника таблицы.
* TABLEQUALIFIER - для различных источников данных этот атрибут может иметь разный смысл. Так, для SQL Server и Oracle - это имя базы данных, в которой содержится таблица, а для Intersolv dBase - это директория в которой расположены DBF файлы.

Пример
Пусть определена HTML - форма для ввода данных:

<HTML>
<HEAD>
<TITLE>Пример ввода данных для создания записи</TITLE>
</HEAD>
<BODY>
<FORM ACTION="/cgi-shl/dbml.exe?Template=example.dbm"
METHOD="POST">
ФИО : <INPUT TYPE="Text" NAME="FullName">
Телефон : <INPUT TYPE="Text" NAME="Phone">
Дата рождения : <INPUT TYPE="Text" NAME="Birthday">
</FORM>
</BODY>
</HTML>

Следующий шаблон, example.dbm, которому будут переданы данные из формы создает запись в таблице и выдает подтверждающее сообщение:

<DBINSERT DATASOURCE="Person DB" TableName="Persons"
FORMFIELDS="FullName,Phone,Birthday">
<HTML>
<HEAD><TITLE>Подтверждение</TITLE></HEAD>
<BODY>
<H1>Запись создана!</H1>
</BODY>
</HTML>

5.7 Выполнение запросов к базам данных

Для выполнения запросов к базе данных используется тег DBQUERY. Этот тег имеет следующий синтаксис:

<DBQUERY NAME="имя запроса"
DATASOURCE="имя источника данных odbc"
SQL="sql выражение" TIMEOUT=n MAXROWS=n DEBUG>

Атрибут NAME определяет имя запроса, которое используется далее для отображения результата выполнения запроса. Имя запроса должно начинаться с буквы и может содержать буквы и цифры (пробелов быть не должно).

Атрибут DATASOURCE задает имя источника данных ODBC, который должен быть создан с помощью интерфейса администратора Cold Fusion.

Ключевым атрибутом тега DBQUERY, является атрибут SQL, который собственно и определяет запрос к базе данных на языке SQL (для улучшения читабельности, допускается расположение значения атрибута SQL на нескольких строках).

Создавая SQL запрос, следует помнить, что конкретная база данных может иметь свои особенности в синтаксисе SQL, использование которых ограничивается этой базой данных. Чтобы проверить, является ли конкретное SQL выражение совместимым с ODBC и независимым от конкретной базы данных, лучше всего использовать Microsoft Query, входящий в состав Microsoft Office. Для этого нужно в меню Microsoft Query выбрать "Файл/Выполнить SQL", в появившемся окне диалога ввести предложение SQL, выбрать источник данных ODBC, нажав на кнопку "Источники...", после чего нажать на кнопку "Выполнить". Этот продукт можно также использовать и для создания SQL - выражений, используя для этого визуальные средства создания запросов. Получить SQL - выражение созданного таким образом запроса можно нажав на кнопку "SQL" в панели инструментов.

Атрибут MAXROWS является необязательным и определяет максимальное количество записей, которые могут быть возвращены в результате выполнения запроса.

Атрибут TIMEOUT также является необязательным и определяет максимальное количество миллисекунд для выполнения запроса, до выдачи сообщения об ошибке. Заметим, что этот атрибут поддерживается только некоторыми ODBC - драйверами (например, драйвером для MS SQL Server 6.0).

Атрибут DEBUG используется для отладки запросов. При наличии этого атрибута пользователю отправляется дополнительная информация о выполнении этого запроса, такая как текст выполненного SQL - запроса, число возвращенных записей и др.

Приведем пример запроса с именем 'AllPersons', который возвращает все записи таблицы 'Persons' из базы данных, с которой связан источник данных ODBC с именем 'Person DB':

<DBQUERY NAME="AllPersons" DATASOURCE="Person DB"
SQL="select \* from Persons">

Для динамической настройки SQL - выражения можно использовать параметры, переданные в шаблон. Это могут быть параметры, переданные из формы, URL, а также переменные CGI. Параметры, используемые внутри SQL - выражения, должны быть обрамлены символом "#" (например #Name#). При обработке запроса Cold Fusion ищет параметр с таким именем среди параметров, полученных из формы, в URL или среди переменных CGI. При нахождении подходящего параметра его значение подставляется вместо соответствующей ссылки на параметр.

**Примеры использования параметров в SQL -выражении**

Пример 1
Предположим что обрабатывается URL

"/cgi-shl/dbml.exe?Template=prs.dbm&Id=22",

а атрибут SQL в DBQUERY имеет вид

SQL="select \* from Persons where Id = #Id#",

тогда в базу данных будет передано следующее SQL - выражение:

select \* from Persons where Id = 22 .

Пример 2
Предположим, что в шаблон передан параметр FirstLetters и нужно найти в таблице Persons записи, в которых первые буквы в поле FullName совпадают со значением этого параметра. Значение атрибута SQL в этом случае будет следующим:

SQL="select \* from Persons
where FullName like '#FirstLetters#%'"

Следует обратить внимание на то, что маска, состоящая из параметра и символа '%', в отличие от предыдущего примера, обрамлена одинарными кавычками. Это связано с тем, что поле Id из примера 1 имеет числовой тип, а поле FullName - текстовый тип (синтаксис SQL требует, чтобы текстовые значения всегда были обрамлены одинарными кавычками).

Для того чтобы задать маску, в примере использовался символ '%', который в SQL - запросах соответствует произвольной последовательности символов. Также для определения маски может использоваться символ '\_' (подчерк), соответствующий одному произвольному символу.

5.8 Использование результатов запроса для динамического создания HTML - документа

Для вывода данных возвращаемых в результате выполнения запроса определенного в DBQUERY применяется тег DBOUTPUT. Внутри этого тега, связанного с конкретным запросом, может находиться обычный текст, теги HTML, ссылки на поля определенные в запросе. При обработке шаблона, содержимое тега DBOUTPUT отправляется клиенту для каждой записи, возвращаемой в результате выполнения запроса, с подстановкой соответствующих значений параметров и полей.

Тег DBOUTPUT имеет следующий синтаксис:

<DBOUTPUT QUERY="имя запроса" MAXROWS=n>
Текст, теги HTML,
ссылки на поля и параметры (т.е. #Name#)
</DBOUTPUT>

Атрибут QUERY применяется для указания имени запроса DBQUERY, результат выполнения которого будет использоваться, а атрибут MAXROWS определяет максимальное количество записей этого запроса, которые будут переданы для вывода в тег DBOUTPUT.

Пример
Для вывода результата выполнения запроса с именем 'AllPersons', отображая имя персоны и телефон, и разделяя записи горизонтальной линией, может использоваться следующая конструкция:

<DBOUTPUT QUERY="AllPersons" MAXROWS=50>
<HR>
#FullName# (Телефон: #Phone# ) <BR>
</DBOUTPUT>

Результат обработки этого тега будет иметь вид:

<HR>
Иванов Иван Иванович (Телефон: 222-22-22 ) <BR>
<HR>
Петров Петр Петрович (Телефон: 444-44-44 ) <BR>

5.9 Вывод результата выполнения запроса в виде таблицы

Теги DBTABLE и DBCOL всегда употребляются вместе для отображения результата выполнения запроса в виде таблицы.

Атрибуты тега DBTABLE:

|  |
| --- |
|    |
|  QUERY  | - имя DBQUERY, для которого нужно отобразить данные; |
|  MAXROWS  | - максимальное количество записей, которое может быть отображено в таблице; |
|  COLSPACING  | - количество пробелов, которые будут вставлены между колонками (по умолчанию 2); |
|  HEADERLINES  | - количество строк, которые будут отведены для заголовка (по умолчанию 2); |
|  HTMLTABLE  | - при наличие этого тега результат запроса будет отображен в виде HTML - таблицы, в противном случае будет использован тег HTML <PRE>. |
|  BORDER  | - используетя только вместе с атрибутом HTMLTABLE для отображения рамки в таблице. |

Атрибуты тега DBTABLE:

|  |
| --- |
|    |
|  HEADER  | - текст, который будет выводиться как заголовок колонки; |
|  WIDTH  | - ширина колонки в символах (по умолчанию 20); |
|  ALIGN  | -выравнивание содержимого колонки (LEFT, RIGHT и CENTER); |
|  TEXT  | -заключенный в кавычки текст, определяющий содержимое колонки, в котором могут находиться те же теги, ссылки на параметры и др., что и в теге DBOUTPUT. |

Приведем пример использования тегов DBTABLE и DBCOL:

<DBTABLE QUERY "AllPersons" MAXROWS=20>
<DBCOL HEADER="Фамилия Имя Отчество" WIDTH="30" TEXT="#FullName#">
<DBCOL HEADER="Телефон" WIDTH="10" TEXT="#Phone#">
<DBCOL HEADER="Дата рождения" WIDTH="9" TEXT="# DateFormat(Birthday)#">
</DBTABLE>

5.10 Дополнительные замечания по созданию DBML - шаблонов

* В шаблонах DBML, для комментариев, используется три тире (<!---), в отличие от двух в HTML (<!--). Эта специальная форма синтаксиса для комментариев позволяет Cold Fusion игнорировать теги и текст, содержащиеся внутри этого комментария.
* Ссылки на другие файлы (графические, HTML и CGI программы), содержащиеся в DBML шаблоне, должны использовать полный путь, начиная с корневого каталога сервера.
* Так как символ '#' является специальным символом в Cold Fusion, то чтобы включить его в область вывода, определяемую тегом DBOUTPUT, нужно в шаблоне использовать два символа '#' вместо одного. Это же правило относится и к двойной кавычке, если этот символ нужно вставить, например, в атрибут SQL (см. следующий абзац).
* Cold Fusion не поддерживает имена полей, содержащих пробелы, внутри тега DBOUTPUT. Если в имени поля все же встречаются пробелы, то при определении SQL выражения в теге DBQUERY, для каждого такого поля следует задать псевдоним. Например,

SQL="select ""Full Name"" as FullName from Persons"

Псевдонимы бывает, также, полезно применять для удобства, в случаях, если имя поля велико.

5.11 Использование параметров и переменных в шаблонах

5.11.1 Поля формы и параметры URL

Если в шаблон, с помощью полей формы или в URL, были переданы параметры, то внутри любого тега DBML к этим параметрам можно обращаться, используя следующий синтаксис: #Form.Name#, #URL.Name#. На самом деле, префиксы 'Form.' и 'URL.' могут опускаться, если заранее известно, что не может быть параметров других типов с такими же именами. Это правило относится ко всем типам параметров и переменных.

5.11.2 Переменные окружения CGI

Каждый сеанс связи вызывающий CGI - программу имеет конкретные переменные окружения. Доступ к ним из шаблона осуществляется, также как и к другим параметрам, только используется префикс 'CGI.', например #CGI.REMOTE\_ADDR#.

5.11.3 Применение тега DBSET для создания переменных

С помощью тега DBSET можно создавать переменные непосредственно в самом шаблоне и использовать их. Приведем пример, в котором создается переменная #UserId# и ей присваивается значение 10.

<DBSET #UserId#=10>

В правой части операции присваивания в DBSET, может находиться как число, текст (заключенный в кавычки), так и любые параметры доступные в шаблоне, например #CGI.SCRIPT\_NAME#. Обращаясь к этим переменным, следует использовать префикс Variable, например #Variable.UserId#.

5.11.4 HTTP Cookies

Cookies - это механизм, позволяющий приложениям о стороны сервера сохранять и использовать параметры на стороне клиента. Этот механизм поддерживается всеми версиями Netscape Navigator, MS Internet Explorer начиная с версии 2.0, и будет поддерживаться остальными Web - браузерами в ближайшем будущем.

Для сохранения параметров в Cookies. Используется тег DBCOOKIE, имеющий следующий синтаксис:

<DBCOOKIE NAME="Имя\_параметра" VALUE="Значение параметра" EXPIRES="Срок действия" SECURE>

В атрибутах NAME и VALUE определяются имя и значение параметра соответственно. Атрибут EXPIRES определяет, когда закончится срок действия этого параметра. Этот атрибут может быть задан как дата т.е. '10/09/97', количество дней (т.е. 10, 100), NOW (удаляет параметр) или NEVER. Наличие необязательного атрибута SECURE запрещает отправлять параметр браузеру, если тот не поддерживает стандарт SSL.

К параметрам, хранимым в cookies можно обращаться внутри любого тега DBML, добавляя префикс 'Cookies.', например:

<DBOUTPUT>
#Cookies.User\_Id#
</DBOUTPUT> .

5.11.5 Использование результатов выполнения запросов

После выполнения запроса, результат его выполнения может быть использован в качестве динамического параметра для спецификации другого запроса. Например, если создан запрос с именем 'FindUser', который возвращает идентификатор записи, расположенный в поле 'USER\_ID', то можно использовать этот идентификатор в другом запросе, используя имя запроса как префикс к имени поля, разделяя их точкой (т.е. #FindUser.UserId#).

Каждый запрос, описанный тегом DBQUERY, после выполнения имеет два специальных атрибута, RecordCount и CurrentRow, содержащих информацию о количестве возвращенных в результате выполнения запроса записей и о текущей записи, обрабатываемой тегом DBOUTPUT, соответственно. Используются эти атрибуты так же как и поля запроса (#FindUser.RecordCount#).

5.12 Проверка корректности данных и форматирование вывода

5.12.1 Проверка корректности данных в полях формы

В Cold Fusion предусмотрен механизм проверки корректности заполнения полей формы. Этот механизм основан на добавлении в форму дополнительных полей типа HIDDEN (скрытые), с именем, составленным из имени поля, значение которого нужно проверить и одного из допустимых в Cold Fusion суффиксов, задающих контекст проверки.

В следующей таблице приведен список всех суффиксов используемых для проверки:

|  |  |  |
| --- | --- | --- |
| **Суффикс**  | **Значение атрибута VALUE**  | **Описание**  |
| \_required  | Текст сообщения об ошибке | Проверяется произведен ли ввод в поле формы.  |
| \_integer  | Текст сообщения об ошибке | Проверяется, является ли значение, введенное пользователем, целым числом.  |
| \_float | Текст сообщения об ошибке | Проверяется, является ли значение, введенное пользователем, числом.  |
| \_range | MIN = Минимальное\_Значение MAX = Максимальное\_Значение  | Проверяется, находится ли введенное число в определенных границах.  |
| \_date | Текст сообщения об ошибке | Проверяется, находится ли введенная пользователем дата в одном из следующих форматов: DD/MM/YY, DD/MM/YYYY, DD/MM (используется текущий год). В качестве разделителя можно, также, использовать тире (т.е. DD-MM-YY).  |

Пример
Ниже приведен фрагмент описания формы, состоящий из двух текстовых полей: обязательное для заполнения поле FullName и поле типа дата Birthday, и для каждого из этих полей описано поле типа HIDDEN задающее контекст проверки.

Фамилия Имя Отчество : <INPUT TYPE="TEXT" NAME="FullName">
<INPUT TYPE="HIDDEN" NAME="FullName\_required"
VALUE="Фамилия Имя Отчество должны быть заданы!">
Дата рождения : <INPUT TYPE="TEXT" NAME="Birthday">
<INPUT TYPE="HIDDEN" NAME="Birtday\_date"
VALUE="Дата рождения должна быть в формате DD/MM/YYYY (например, 20.08.1968)">

5.12.2 Функции вывода в DBML

Для отображения, данные в нужном формате, в Cold Fusion предусмотрены специальные функции. При использовании такой функции, примененной к конкретному параметру, она заключается в символ '#', например, #DateFormat(Form.LastUpdate)#.

***Список основных функций***

|  |  |  |
| --- | --- | --- |
| **Функция**  | **Описание**  | **Пример**  |
| DateFormat  | Отображает поле базы данных типа 'дата/время' или 'дата' в формате DD/MM/YY.  | 12/01/96  |
| TimeFormat  | Отображает поле базы данных типа 'дата/время' в формате HH:MI AM/PM..  | 10:22 AM  |
| NumberFormat  | Отображает числовые значения как целые числа, разделяя разряды запятой.  | 10,256 |
| DecimalFormat  | То же, что и NumberFormat, плюс отображаются два знака после десятичной точки.  | 10,256.3  |
| DollarFormat  | То же, что и DecimalFormat, плюс добавляется символ $ и вместо знака минус перед отрицательным значением оно помещается в скобки.  | $10,256.73  |
| YesNoFormat  | Отображает данные логического типа как Yes или No. Все ненулевые значения интерпретируются как Yes, нуль - как No.  | Yes |
| ParagraphFormat  | Применяется при отображении данных введенных в поле TEXTAREA. Преобразует символ перевода строки в пробел, два перевода строки подряд - в тег параграфа HTML (<P>).  |   |
| HTMLCodeFormat  | Удаляет символ перевода строки и пропускает все специальные символы (>, <, ", &), применяя к тексту тег преформатирования HTML (<PRE>).  |   |
| HTMLEditFormat  | То же, что и HTMLCodeFormat, только без добавления тега <PRE>.  |   |

Кроме вышеперечисленных есть еще несколько функций обеспечивающих дополнительные возможности манипулирования параметрами в шаблоне:

* ParameterExists

Эта функция проверяет, доступен ли в шаблоне параметр с заданным именем, возвращая Yes или No. Например, чтобы проверить был ли отправлен из формы параметр UserId, используется следующее выражение:

<DBIF #ParameterExists(Form.UserId)# is Yes>

* PreserveSingleQuotes

Эта функция обычно используется в SQL выражениях для устранения из значений параметров одиночных кавычек, которые являются специальным символом в SQL. Приведем пример использования этой функции:

SELECT \* FROM Persons WHERE
FullName Like '#PreserveSingleQuotes(Form.FullName)#%'

* URLEncodedFormat

Функция заменяет пробелы на символ "+" и все не латинские символы и цифры - шестнадцатеричным эквивалентом, что позволяет использовать результат в строке URL.

* IncrementValue и DecrementValue

Эти функции возвращают переданный им параметр, прибавив или отняв от него единицу соответственно. Например, чтобы увеличить параметр 'OrderCount' можно воспользоваться следующим выражением:

<DBSET #OrderCount# = #IncrementValue(OrderCount)#

* ValueList и QuotedValueList

Используя в качестве аргумента имя поля конкретного запроса, эти функции возвращают разделенный запятыми список значений этого поля для каждой записи, возвращенной в результате выполнения запроса.

Например, если запрос возвращает четыре записи, то результат функции ValueList будет иметь вид 11,22,33,44, а результат функции QuotedValueList, примененной к этим же данным, будет возвращать '11','22','33','44'.

Эти функции могут применятся для использования результата одного запроса в операции IN последующего запроса, например:

<DBQUERY NAME="Customers" ...определение запроса...>
<DBQUERY NAME="CustomerOrders" DATASOURCE="EXAMPLE"
SQL="SELECT \* FROM Orders WHERE Customer\_ID
IN ( #ValueList(Customer.CustomerID)# )">

5.13 Динамическое изменение содержимого документа

5.13.1 Условный оператор (DBIF & DBELSE)

Основным средством динамического определения содержимого документа являются теги DBIF и DBELSE, позволяющие производить проверку некоторого условия и в зависимости от этого формировать результат.

Тег DBIF имеет следующий синтаксис (тег DBELSE может не использоваться):

<DBIF *значение* оператор *значение*>
теги HTML и DBML
<DBELSE>
теги HTML и DBML
</DBIF>

В качестве элемента тега DBIF - "*значение*", могут использоваться любой параметр или переменная Cold Fusion (например, #Form.Name#, #CGI.User\_Agent#), числовое значение, произвольная последовательность символов (заключенная в кавычки).

Элемент тега DBIF - "*оператор*" ограничивается следующим списком:

* **is** - сравнивает два значения, с учетом регистра, и возвращает значение True (истина), если эти значения совпадают;
* **is not** - оператор обратный оператору **is**;
* **Contains** - проверяет, содержится ли значение находящееся слева от оператора в значении справа и возвращает True, если да;
* **does not contain** - оператор обратный оператору **contains**;
* **great then** - проверяет, что значение слева от оператора больше чем справа и возвращает True, если да;
* **less then** - проверяет, что значение слева от оператора меньше чем справа и возвращает True, если да;
* **greater then or equal to** - оператор обратный оператору **less then**;
* **less then or equal to** - оператор обратный оператору **great then**;

Пример
<DBIF #PersonSearch.RecordCount()# is 0>
<P>Лиц, удовлетворяющих заданным критериям поиска,
в базе данных не обнаружено!
<DBELSE>
<DBOUTPUT QUERY="PersonSearch">
<HR>
Фамилия Имя Отчество: #FullName# <BR>
<DBIF #Phone# is "">
Телефон: #Phone# <BR>
<DBIF>
</DBOUTPUT>
</DBIF>

5.13.2 Перенаправление на другой URL (DBLOCATION & DBABORT)

Для перенаправления пользователя на другой URL предназначен тег DBLOCATION. Этот тег обычно применяется, если в шаблоне выполняется один или более запросов, а затем нужно сразу же перейти к другому документу, либо URL, на который нужно отправит пользователя, зависит от параметра. Приведем пример использования тега, иллюстрирующий его синтаксис:

<DBIF #NewPassword# is not #PasswordConfirmation#>
<DBLOCATION URL="/login/invalidpassword.htm">
</DBIF>

В качестве значения атрибута URL тега DBLOCATION можно использовать параметры и переменные, допустимые в шаблоне. Например, <DBLOCATION URL=#Page#>.

Для прерывания обработки шаблона в Cold Fusion используется тег DBABORT.

<P>Этот текст будет отправлен клиенту
<DBABORT>
<P>Этот текст не будет отправлен клиенту

Обычно этот тег используется при неправильной аутентификации.

5.13.3 Включение в шаблон других шаблонов

По мере усложнения приложений, разрабатываемых с использованием Cold Fusion, появляется необходимость упростить используемые шаблоны. Одним из способов решения этой проблемы, предлагаемых в Cold Fusion, является выделение из шаблонов часто используемых блоков, таких как запросы и области вывода, и их многократное использование в других шаблонах. Для включения часто используемых шаблонов в другой шаблон, используется тег DBINCLUDE.

Тег DBINCLUDE может быть расположен в любом месте шаблона, кроме как в тегах DBQUERY, DBOUTPUT и DBTABLE. Тег DBINCLUDE имеет атрибут TEMPLATE который задает путь до файла с шаблоном. Этот шаблон будет обработан Cold Fusion как часть основного шаблона (то есть, в нем могут использоваться запросы, уже запущенные в основном шаблоне, а также ссылки на параметры формы, URL и CGI).

Приведем пример включения в шаблон шаблона с именем 'test.dbm':

<DBINCLUDE TEMPLATE="test.dbm">

5.13.4 Определение типа данных MIME для содержимого документа

Cold Fusion позволяет задавать тип MIME для данных которые будут отправлены пользователю из текущего шаблона (по умолчанию используется text/html). Для этого используется тег DBCONTENT, имеющий единственный атрибут TYPE, который, собственно, и задает тип данных. Например, чтобы отправить клиенту VRML - документ может использоваться следующий шаблон:

<DBCONTENT TYPE="x-world/x-vrml">
<DBQUERY NAME="GetCyberRoom"
SQL="SELECT VRML\_Script FROM CyberRooms WHERE
RoomNumber=#URL.RoomNumber#">
<DBOUTPUT QUERY="GetCyberRoom">
#VRML\_Script#
</DBOUTPUT>

Заметим, что Cold Fusion не будет отправлять клиенту текст, расположенный до тега DBOUTPUT.

5.14 Расширенные возможности

5.14.1 Динамическое определение SQL выражения

В некоторых сложных приложениях может потребоваться, в зависимости от значений параметров, определять не только содержание запроса, но и его структуру. В Cold Fusion предусмотрен тег DBSQL, который может употребляться внутри тега DBQUERY, доопределяя SQL - выражение, в зависимости от значений параметров. Тег DBSQL имеет единственный атрибут SQL, значение которого будет добавляться к основному SQL - выражению.

Пример
<DBQUERY NAME="SiteSearch" DATASOURCE="Sites Database"
SQL="SELECT \* FROM SITES WHERE SiteType = #SiteType# ">
<DBIF #Form.City# is not "">
<DBSQL SQL=" AND City = '#Form.City#' ">
</DBIF>
<DBIF #Form.SortOrder# is not "">
<DBSQL SQL=" ORDER BY #Form.SortOrder# ">
</DBIF>
</DBQUERY>

5.14.2 Поддержка транзакций

Для объединения нескольких запросов в одну транзакцию может быть использован тег DBTRANSACTION. Все запросы, содержащиеся внутри этого тега, будут интерпретироваться как одна транзакция. То есть все изменения сделанные в базе данных, либо будут одновременно сохранены, либо не будет сохранено ни одно из них.

Приведем пример, в котором денежная сумма переводится с одного банковского счета на другой:

<DBTRANSACTION>
<DBQUERY NAME="WithdrawCash" DATASOURCE="Bank Accounts"
SQL = "UPDATE Accounts SET Balance = Balance - #Amount#
WHERE Account\_ID = #AccountFrom# ">
<DBQUERY NAME="DepositCash" DATASOURCE="Bank Accounts"
SQL = "UPDATE Accounts SET Balance = Balance + #Amount#
WHERE Account\_ID = #AccountTo# ">
</DBTRANSACTION>

Заметим, что не все драйверы ODBC поддерживают транзакции. Например, драйверы для Oracle, SQL Server и Access поддерживают транзакции, а драйверы для FoxPro, dBase и Paradox - нет.

5.14.3 Вложенные области вывода и группирования

Теги DBOUTPUT могут вкладываться друг в друга, с целью сгруппировать области вывода. Группирование достигается с помощью использования атрибута GROUP в теге DBOUTPUT, который содержит другой тег DBOUTPUT. Этот атрибут определяет поле по которому будет производится группирование. Внешний тег DBOUTPUT обычно используется для вывода заголовка группы , а внутренний - для вывода записей содержащихся в группе.

Приведем пример вывода сотрудников организации сгруппированных по отделам:

<DBQUERY NAME="ListEmployees" DataSource="Employees"
SQL="SELECT \* FROM Emp ORDER BY Department">
<DBOUTPUT QUERY="ListEmployees" GROUP="Department">
<P> <H2>#ListEmployees.Department#</H2>
<UL>
<DBOUTPUT>
<LI> #FullName# ##
</DBOUTPUT>
</UL>
</DBOUTPUT>

Обратим внимание на то, если поле используется для группирования, то результат запроса должен быть отсортирован по этому полю. Если используется многоуровневое группирование (ограничений на количество уровней вложенности тегов DBOUTPUT нет), то соответственно в SQL - выражении должна быть задана многоуровневая сортировка (например, ORDER BY Country, Region).

5.14.4 Использование списочных полей с множественным выбором

Если HTML - форма содержит поле типа SELECT с множественным выбором, либо поля с одинаковыми именами (например, поля типа checkbox), то данные будут переданы в шаблон в виде, разделенных запятыми, списка значений. Такая форма представления наиболее удобна для использования в операторе IN языка SQL.

Пример
Предположим, что в форме содержится поле:

<SELECT NAME="SelectedPersons" MULTIPLE SIZE="3">
<OPTION VALUE="1">Иванов И.И.
<OPTION VALUE="2">Петров П.П.
<OPTION VALUE="3" SELECTED>Сидоров С.С.
</SELECT>

Этот параметр, переданный в шаблон, может быть использован в следующем SQL - выражении:

SQL="SELECT \* FROM Persons
WHERE Person\_ID IN ( #SelectedPersons# )

Следует обратить внимание на то, чтобы параметр, который используется в операторе IN, был не пуст. Для этого можно пользуясь стандартными средствами Cold Fusion описать поле как требуемое, либо использовать поле типа HIDDEN с тем же именем и с заведомо неверным значением. Например, предыдущий пример можно дополнить следующим полем:

<INPUT TYPE="HIDDEN" NAME="SelectedPersons" VALUE="-1">

5.14.5 Дополнительные команды SQL

В теге DBOUTPUT, кроме выражения SELECT языка SQL, допускается использование и любых других, допустимых для конкретного источника данных, SQL - команд, включая:

* INSERT - добавление записи в таблицу.
* UPDATE - модификация записи в таблице.
* DELETE - удаление записи из таблицы.

Использование непосредственно команд SQL вместо тегов DBINSERT и DBUPDATE в некоторых случаях может обеспечить большую гибкость и эффективность. Например, при модификации или создании новой записи, появляется возможность использовать все параметры и переменные доступные в шаблоне в том числе и результаты определенных в шаблоне запросов.