

# УПРАВЛЕНИЕ СЕССИЯМИ В COLD FUSION,

# ИЛИ

# ЗДРАВСТВУЙТЕ,

# Я – ВАША

# ТЕТЯ



**АЛЕКСАНДР МЕЖЕНКОВ**

Рассмотрим ситуацию. Покупатель в интернет-магазине заполнил регистрационную форму, побродил по страничкам с выложенным товаром, отложил кое-что в корзину, потом прикинул сколько еще жить до полочки, посидел в задумчивости перед экраном минут двадцать и с сожалением решил выложить из корзины совершенно ненужные ему сейчас галоши. Возникает вопрос, а как web-сервер распознает его при каждом новом обращении. И вообще, как web-сервер различает каждого из сотен посетителей, обращающихся к различным страницам сайта? Одному нужно одно, другому – другое. И о каждом нужно помнить, чтобы после очередного обращения не

спрашивать его удивленно: «А ты кто?» и заставлять его заново заполнять регистрационную форму.

Ответ на первый взгляд может показаться странным: никак!

Давайте попробуем разобраться. Здесь мы предполагаем, что с момента прочтения нашей первой статьи («ColdFusion, или Возможно, лучшее решение для создания динамических сайтов», журнал «Системный администратор» №1, октябрь 2002г.) вы успели установить ColdFusion и уже хотя бы немного ориентируетесь в его среде.

Создадим два шаблона ColdFusion. В одном из них (назовем его setMyName.cfm) определяется переменная: myName, после чего немед-

ленно вызывается другой шаблон sayHi.cfm:

```
<cfset myName="Alexander Mejenkov">
<cflocation url="sayHi.cfm">
```

Тег <cflocation> и осуществляет обращение к новому шаблону sayHi.cfm, в котором мы предполагаем использовать значение созданной на предыдущей странице переменной.

Поместим в файл sayHi.cfm код:

```
<cfoutput>
    Well, hello there, #myName#
</cfoutput>
```

Теперь самое время запустить setMyName.cfm. Как только мы сделаем это, то немедленно получим сооб-

щение об ошибке: «Error resolving parameter MYNAME». Похоже, что ни web-сервер, ни ColdFusion и в самом деле нисколько не позаботились о том, чтобы сохранить информацию о нашем визите!

Проблема заключается в том, что браузер общается с web-сервером, а следовательно и с ColdFusion, при посредстве HTTP-протокола. А суть и основное назначение этого протокола заключается в том, чтобы как можно скорее обслужить запрос клиента, выдать ему запрашиваемую страницу и, завершив с ним сеанс обмена, быть готовым обслуживать следующего клиента. Это означает, что web-сервер, обеспечивающий работу HTTP-протокола, после того как обслужит полученный запрос и отправит ответ клиенту, совершенно забывает о проведенной транзакции. HTTP не сохраняет открытым соединение с клиентом и поэтому не может знать и помнить, чем занимался один и тот же клиент между его последовательными вызовами. Именно поэтому ColdFusion-сервер, который установил переменную myName на одной странице, не может помнить о ее существовании на другой странице. Каждое следующее обращение, пусть даже одного и того же клиента HTTP, а следовательно и ColdFusion, рассматривают как абсолютно нового клиента.

## Application framework

Для решения проблемы сохранения данных между последовательными вызовами одного и того же клиента в ColdFusion введено понятие application framework – структуры (каркаса) приложения, когда каждое приложение имеет свой собственный четко определенный каркас (структуру). Именно этот подход позволяет набору ColdFusion шаблонов (файлов) работать как единое приложение, используя общие переменные. Ключом этого подхода является файл специального шаблона который называется Application.cfm.

## Управление Session- и Client-переменными

Session-переменные доступны броу-

зеру клиента для текущей сессии (сеанса связи) и для данного приложения.

Session-переменные хранятся в памяти сервера и время их жизни ограничено установками сервера ColdFusion и вашего приложения. Причем если в приложении время жизни будет задано больше, чем на сервере, то будут действовать установки сервера. Session-переменные имеют тип структуры. Для доступа к ним необходимо использовать префикс Session. За счет того, что Session-переменные хранятся в памяти сервера, доступ к ним чрезвычайно быстр.

Client-переменные подобны Session в том смысле, что они уникальны для данного пользователя, однако они доступны для браузера клиента в различных сессиях данного приложения. Другим отличием является место хранения. Client-переменные могут храниться в cookies на клиентской машине (естественно, что для этого на компьютере клиента должно быть разрешено использование cookies), а также в реестре Windows-сервера или в базе данных.

Управление Client- и Session- переменными определяется установками, которые задаются в файле Application.cfm, который (если он есть) выполняется перед загрузкой каждой страницы. Обратите внимание на то, что этот файл должен называться именно так и начинаться с заглавной буквы «A». Это существенно для UNIX-систем. Файл может быть один для всего приложения и располагаться в корневом каталоге приложения, или их может быть несколько и располагаться они могут в подкаталогах, в которых находятся файлы, решающие конкретную задачу.

Для управления как Session-, так и Client-переменными ColdFusion создает уникальную пару идентификационных переменных CFID и CFTOKEN. CFID – это последовательно увеличивающий свое значение счетчик, а CFTOKEN – содержит уникальное случайное число. Типичные значения этих переменных CFID=3 CFTOKEN=54579676. Копии CFID и CFTOKEN сохраняются в памяти сервера в виде Session-переменных. Вторая копия этих переменных сохраня-

ется в виде cookies на клиентской машине. Когда пользователь осуществляет запрос к ColdFusion-приложению, значения CFID и CFTOKEN, как и все cookies, направляются на сервер. Далее сервер ColdFusion пытается сопоставить полученные значения с парой, хранящейся в своей памяти. Если они совпадают, пользователь и/или его сессия идентифицируются. В случае если вам известно, что на пользовательской машине использование cookies запрещено, вам придется несколько усложнить свое приложение, предусмотрев пересылку вручную этих переменных через URL- или FORM-переменные. Например, в теге <cflocation> для этого предусмотрен специальный атрибут ADDTOKEN = «Yes».

Для задания установок управления Session- и/или Client-переменными используется тег <cfapplication>, включаемый в файл Application.cfm.

Рассмотрим синтаксис тега <cfapplication>:

```
<cfapplication name = "application_name"
  clientManagement = "Yes" or "No"
  clientStorage = "datasource_name"
  or "Registry" or "Cookie"
  setClientCookies = "Yes" or "No"
  sessionManagement = "Yes" or "No"
  sessionTimeout =
    #CreateTimeSpan(days, hours,
    minutes, seconds)#

  applicationTimeout =
    #CreateTimeSpan(days, hours, minutes,
    seconds)#
  setDomainCookies = "Yes" or "No">
```

- name – имя вашего приложения;
- clientManagement – необязательный параметр. Разрешает использование Client-переменных. Значение по умолчанию – «No»;
- clientStorage – необязательный параметр. Определяет место хранения Client-переменных. По умолчанию – реестр;
- setClientCookies – необязательный параметр. Значение «Yes» разрешает использование cookies на клиентской машине. Значение по умолчанию – «Yes». Если этот атрибут установлен в «No», ColdFusion автоматически не пытается сохранить CFID и CFTOKEN в виде cookies на клиентской машине. В этом случае вы должны вручную включить CFID и CFTOKEN в URL каждой страницы, на которой использу-

- ются Session- или Client-переменные;
- `sessionManagement` – параметр необязателен. Значение «Yes» разрешает использование Session-переменных. Значение по умолчанию – «No»;
- `sessionTimeout` – необязательный параметр. Определяет время жизни Session-переменных в виде объекта `date/time`, возвращаемого функцией `CreateTimeSpan()`. Аргументы этой функции задаются в числовых величинах: дни, часы, минуты и секунды, разделенные запятыми. Значение по умолчанию определяется на странице `Variables` в ColdFusion Administrator;
- `applicationTimeout` – необязательный параметр. Определяет время жизни Application-переменных (в данной статье мы их не рассматриваем) в виде объекта `date/time`, возвращаемого функцией `CreateTimeSpan()`. Аргументы этой функции задаются в числовых величинах: дни, часы, минуты и секунды, разделенные запятыми. Значение по умолчанию определяется на странице `Variables` в ColdFusion Administrator;
- `setDomainCookies` – необязательный параметр. Устанавливает CFID и CFTOKEN cookies для всего домена, а не только для текущего хоста. Значение по умолчанию – «No». Этот атрибут следует устанавливать в «Yes» только для приложений, используемых в кластерах.

Типичное применение тега `<cfapplication>` выглядит следующим образом:

```
<cfapplication
name="myBestApplication"
    ClientManagement="No"
    SessionManagement="Yes"
    SessionTimeout=#CreateTimeSpan(0,0,1,0)#
    SetClientCookies="Yes">
```

При использовании Session-переменных возникает проблема момента окончания сессии. Фактически сессия уже может закончиться, а cookies все еще будут жить на машине клиента. Например, в приведенном выше теге `<cfapplication>` время жизни cookies установлено 1 час. В этом случае любой человек с машины клиента в течение 1 часа, после того как

настоящий клиент покинул свое рабочее место, сможет продолжить сессию.

Решение проблемы:

```
<cfif IsDefined("Cookie.CFID") AND
IsDefined("Cookie.CFTOKEN")>
<cfset localCFID = Cookie.CFID>
<cfset localCFTOKEN = Cookie.CFTOKEN>
<cfcookie name = "CFID"
value=#localCFID#>
<cfcookie name = "CFTOKEN"
value=#localCFTOKEN#>
</cfif>
```

Как видно в приведенном выше коде CFID и CFTOKEN cookies, созданные сервером ColdFusion в процессе обработки файла `Application.cfm`, надо переписать (пересоздать), но без объявления времени жизни, точнее опустив атрибут `EXPIRES`. Благодаря странному поведению cookies (по крайней мере в MS IE) они создаются в памяти, даже когда пользователь запретил их использование на своей машине.

При подобном переопределении (как в приведенном выше коде) cookies будут созданы лишь в памяти машины клиента, а не будут записаны на его жесткий диск. И тогда, как только приложение браузера будет закрыто, информация о cookies также будет удалена из памяти и, как следствие, сессия будет закончена с закрытием окна браузера.

### Плюсы и минусы Session-переменных

Благодаря тому что Session-переменные хранятся в памяти сервера, в качестве Session-переменных можно использовать не только простые типы данных, но и сложные типы, такие как массивы, структуры и запросы.

Но эта же причина (хранение Session-переменных в памяти сервера) не позволяет их использовать в ситуации, когда несколько серверов работают в одном кластере.

У Session-переменных есть и другой, более существенный недостаток. Он проистекает из того, как ColdFusion управляет этими переменными. Может случиться так, что ColdFusion спутает разные потоки для различных браузеров, и переменные, установленные для одного браузера будут использованы в другом потоке, где обрабатываются запросы от второго браузера. Эта проблема возникнет

не только для Session-переменных, но для всех переменных application- и server- области определения. Allaire называет эти переменные «shared scope» (переменные разделяемой области видимости).

Почему это происходит?

В процессе работы ColdFusion сервер способен обрабатывать одновременно несколько запросов, выполняя каждый из них в отдельной области памяти. Приложения, которые могут выполнять одновременно несколько запросов, называются многопоточными («multi-threaded applications») приложениями. То есть поток – это отдельная область памяти.

Проблема повреждения данных возникает, когда в многопоточном приложении возникает попытка одновременного доступа к «shared scope»-переменным. Решение проблемы заключается в ограничении доступа к переменным так, чтобы, пока ColdFusion-сервер обрабатывает «shared scope»-переменную, ни один другой процесс не мог бы к ней обратиться. Подобный запрет временно превращает ColdFusion-сервер в однопоточное приложение. То есть проблема заключается не столько в самом ColdFusion, сколько в небрежности программиста, не предусмотревшего возможности одновременного обращения разных клиентов к одним и тем же данным. Ограничение доступа к переменным достигается использованием тега `<cflock>`.

Этот тег имеет несколько атрибутов:

- `name` – в основном используется для обратной совместимости версий или для специальных случаев, которые для данного случая не имеют значения;
- `scope` – определяет одну из разделяемых областей видимости (shared scopes): `session`, `application` или `server`;
- `timeout` – время, в течение которого программа должна пытаться получить подтверждение блокировки (время ожидания может возникнуть в случае, если другой `<cflock>` уже успешно заблокировал сервер);
- `throwontimeout` – логическое значение (Yes/No), определяет, нужно ли генерировать событие ошибки при невозможности получить блоки-

ровку по истечении времени timeout. При использовании этого параметра вы наверняка захотите поместить свой <cflock>-код в блок <cftry><cfcatch>;

- type – «readonly» или «exclusive». Блокировка типа Readonly на самом деле не совсем блокировка. Ее лучше сравнить со своего рода телохранителем: если какой-нибудь другой процесс попытается получить блокировку, этот тип блокировки воспрепятствует или, лучше сказать, не допустит этого. Блокировка Readonly используется, когда нет нужды изменять значение переменных из shared scope области видимости. Блокировка Readonly существенно улучшает быстродействие по сравнению с Exclusive-блокировкой. Блокировку Exclusive следует использовать во всех случаях, когда требуется создать, изменить или удалить «shared scope»-переменные.

## Client-переменные

Client-переменные – это другой подход к созданию постоянных глобальных переменных. Но если, как уже отмечалось, Session-переменные хранятся в памяти сервера, то Client-переменные хранятся на физическом носителе: либо в реестре машины, где установлен сервер ColdFusion, либо в указанном ODBC источнике данных, либо в виде cookies на клиентской машине.

Client-переменные, так же как и Session-переменные, используют одинаковую систему идентификации пользователя с помощью CFID и CFTOKEN, однако Client имеют некоторые преимущества. Поскольку для хранения Client-переменных можно указать центральную базу данных, здесь не возникает проблемы доступа к ним в многосерверном окружении. Все серверы в кластере могут использовать эту базу данных. Для использования этой центральной базы данных надо просто сконфигурировать ODBC источник данных, указав на любую пустую базу данных (естественно, предварительно ее создав), для которой у ColdFusion есть native (родной) ODBC драйвер (например Access), а затем в ColdFusion-администраторе указать (поставив check

box) на использование этого источника данных для хранения Client-переменных.

## Проблемы с Client-переменными и их решение

Казалось бы: если Client-переменные лишены всех недостатков Session-переменных, то почему бы их не использовать всегда и вообще не отказываться от Session-переменных. Дело в том, что у Client-переменных есть свои недостатки и перед тем, как отказываться от Session-переменных, необходимо устранить их.

Первый недостаток: поскольку Client-переменные хранятся на физическом носителе, то невозможно «в лоб» использовать сложные типы данных (такие как массивы, структуры и запросы). Эта проблема решается с помощью WDDX-технологии. (WDDX-технология – тема одной из следующих статей).

Другая проблема связана со временем жизни. Если у Session-переменных время жизни можно задать с точностью до секунды, используя функцию CreateTimeSpan (days, hours, minutes, seconds), то Client-переменные в этом смысле гораздо грубее. Единственная возможность для ограничения их времени жизни – это задать в ColdFusion-администраторе число дней, по истечении которых со дня последнего визита пользователя, относящиеся к нему переменные будут удалены («purge data for clients that remain unvisited for n days»).

Поскольку Client-переменные, так же как и Session-переменные, используют пару cookies CFID и CFTOKEN, то используя тот же прием по переписыванию этих cookies без указания параметра EXPIRED, можно ограничить время использования Client-переменных текущим сеансом. То есть как только окно браузера будет закрыто, эти cookies будут стерты и при следующем обращении клиент не будет идентифицирован. Но все равно для ограничения их времени жизни в базе данных нет такой гибкости определения времени с точностью до секунды. Хотя cookies и будут уничтожены с закрытием окна браузера, с физического носителя они будут стерты лишь по окончании числа дней, указанных в администраторе.

Brian Kotek, написавший ряд советов для ColdFusion в CNET Builder.com (<http://builder.cnet.com/>), предложил прекрасный custom tag, «ClientTimeout.cfm», для решения этой проблемы и предоставления возможности детализации времени жизни Client-переменных. Ниже приводится код этого тега:

```
<CFPARAM NAME="CLIENT.CheckLastVisit"
  "DEFAULT="#CreateODBCDateTime(
  Now())#">

<CFSET Compare = DateCompare(
  DateAdd(«n», (ATTRIBUTES.Timeout * -1),
  CreateODBCDateTime(Now())) ,
  CLIENT.CheckLastVisit)>

<CFIF Compare IS NOT -1>
<CFSET CALLER.TimedOut = "Yes">
<CFELSE>
<CFSET CALLER.TimedOut = "No">
</CFIF>

<CFSET CLIENT.CheckLastVisit =
  CreateODBCDateTime(Now())>
```

Для использования этого тега его необходимо вызвать ДО обращения к любой из ColdFusion страниц – обычно это в Application.cfm файле.

```
<cf_ClientTimeout timeout="15">
```

Атрибут timeout – это число минут, в течение которых вы хотите, чтобы Client-переменные оставались активными после запроса очередной страницы.

Какие переменные когда использовать?

Cookies обычно используются для не слишком важных данных, потеря которых никоим образом не может повлиять на работу приложения. Кроме того, поскольку cookies привязаны к конкретному компьютеру, их необходимо создавать на каждом компьютере, с которого клиент запускает приложение.

Session-переменные обычно используются для ответственных данных, поскольку эта информация хранится в памяти сервера и никогда не передается на машину клиента. Например, Session-переменные могут использоваться для идентификации пользователя.

Пользовательские предпочтения, содержимое покупательской корзины или права доступа клиента являются хорошими кандидатами для Client-переменных.