

SSI: Server Side Include – включения на стороне сервера

SSI – это директивы, вставляемые прямо в HTML-код и служащие для передачи указаний Web-серверу. Встречая такие директивы, которые, называются SSI-вставками, Web-сервер интерпретирует их и выполняет соответствующие действия, например: вставка HTML-фрагмента из другого файла, динамическое формирование страничек в зависимости от некоторых переменных (например, типа браузера) и другие не менее приятные вещи.

Преимущества SSI проявляются, когда нам нужно поддерживать достаточно большой по объему сайт, имеющий определенную структуру и повторяющиеся элементы кода на всех страничках. Вообще, при применении серверных включений сайт удобно рассматривать как состоящий из отдельных блоков, каждый из которых отвечает за свою часть странички. Эти блоки практически неизменны и повторяются от страницы к странице. В эти блоки можно вынести такие элементы странички как: главное меню, рекламные вставки, повторяющиеся элементы оформления страничек и т.д. Физически эти блоки представляют собой просто HTML-файлы, содержащие часть кода, нужную для выполнения их задачи.

Для того чтобы сервер знал, что страничка не обычная, а содержит SSI-директивы, она имеет специальное расширение: *.shtml или *.shtm, наличие которого и заставляет web-сервер предварительно обрабатывать странички. Вообще-то, расширение может быть любое (в том числе html и htm) – в зависимости от конфигурации web-сервера, но в основном применяется именно *.shtml.

Полная страничка формируется web-сервером на лету, собирая код странички из таких вот блоков. Для того чтобы указать серверу, какой блок нужно вставить и в каком месте странички, используется специальная форма записи в виде комментария:

```
<!--#command param="value" -->
```

где

– признак начала SSI-вставки;

command – SSI-команда;

param – параметры SSI-команды.

Базовые директивы SSI

include virtual

Это самая популярная команда SSI – это команда включения содержимого одного файла в другой:

```
<!--#include virtual="/path/file.ssi" -->
```

где

include – команда вставки;

virtual – параметр, задающий путь к файлу;

"/path/file.ssi" – путь к включаемому файлу.

Результатом ее выполнения будет вставка содержимого файла `file.ssi` в месте появления данной директивы. При просмотре сформированного HTML-кода мы не увидим никаких признаков SSI, т.к. данный механизм действует абсолютно прозрачно для браузеров, они получают исключительно корректный HTML-код.

set

Команда установки значения переменной:

```
<!--#set var="pic" value="picture.gif" -->
```

где

pic – имя переменной;

picture.gif – значение переменной.

В данном случае мы определили переменную с именем `pic` и присвоили ей строковое значение `"picture.gif"`. Значение переменной `pic` теперь доступно внутри SSI-вставки, и мы можем его использовать по нашему усмотрению. Например, используя одну и ту же SSI-вставку, но с разными значениями определенной в ней переменной, можно получить различные результаты.

echo

Используется для вывода значения переменной:

```
<!--#echo var="pic" -->
```

Ее выполнение приведет к тому, что в месте появления команды напечатается значение переменной `pic`, т.е. `"picture.gif"`.

Переменная может участвовать в выражениях, в этом случае перед ней ставится знак `'$'`, показывающий, что это именно переменная, а не просто текст. Вот пример:

```
<!--#set var="A" value="green" -->  
<!--#set var="B" value="$A apple" -->
```

После такого присвоения переменная В будет содержать строку "green apple". Если же в текст понадобится просто вставить знак '\$' или какой-нибудь из других специальных знаков, то его нужно предварить слешем, вот так: '\$'. В некоторых случаях для избежания двусмысленности значение переменной может быть заключено в фигурные скобки: "\${A}". Например, для того чтобы указать, что переменной является именно \$A, а не \$ABCD необходимо заключать \$A в фигурные скобки:

```
<!--#set var="B" value="${A}BCD" -->
```

Примечание:

Веб-сервер Apache при использовании директивы echo заменяет символы "<", ">" и "&" на их html-эквиваленты, т.е. соответственно "<," ">" и "&". Для того, чтобы это не происходило необходимо указывать дополнительный параметр «encoding="none"», т.е.:

```
<!--#echo encoding="none" var="pic" -->
```

filesize file (filesize virtual)

Вставляет размер указанного файла.

```
<!--#filesize file="page.shtml"-->
```

В качестве результата выведет размер страницы page.shtml.

flastmod file (flastmod virtual)

Вставляет время последней модификации указанного файла.

```
<!--#flastmod file="page.shtml"-->
```

В качестве результата выведет время последней модификации страницы page.shtml в формате:

```
Wednesday, 20-Feb-2002 15:33:50 Московское время (зима)
```

Примечание

Директивы **filesize virtual** и **flastmod virtual** в отличие от директив **filesize file** и **flastmod file** позволяют обращаться к файлам, используя задание абсолютных путей (от корня сайта):

```
<!--#flastmod virtual="/some_directory/page.shtml"-->
```

exec cmd

exec cgi

Запускает внешнюю программу (**exec cmd**) или cgi-скрипт (**exec cgi**) и вставляет в содержимое страницы вывод.

config errmsg

config sizefmt

config timefmt

Изменяет различные параметры конфигурации SSI. **config errmsg** изменяет стандартное сообщение об ошибке на введенное пользователем. Сообщение об ошибке возникает при неправильном выполнении SSI-директивы, например, при отсутствии cgi-скрипта, который пытаетесь запустить.

```
Это стандартная ошибка при запуске скрипта, которого нет:  
<!--#exec cgi="/cgi-bin/nonexistence.pl"-->  
А теперь заменим сообщение об ошибке и повторим:  
<!--#config errmsg="Ошибка, пишите <a href=mailto:dh@dh.ru>вебмастеру</a>"-->  
<!--#exec cgi="/cgi-bin/nonexistence.pl"-->
```

Вывод:

```
Это стандартная ошибка, возникающая при запуске скрипта, которого нет:  
[an error occurred while processing this directive]  
А теперь заменим сообщение об ошибке и повторим:  
Ошибка, пишите вебмастеру
```

Директива **config sizefmt** изменяет формат вывода размера файла.

```
Размер файла этой страницы в килобайтах:  
<!--#fsize file="ssi2.shtml"-->  
Размер файла этой страницы в байтах:  
<!--#config sizefmt="bytes"-->  
<!--#fsize file="ssi2.shtml"-->
```

Вывод:

```
Размер файла этой страницы в килобайтах: 14к  
Размер файла этой страницы в байтах: 14,079
```

Директива **config timefmt** меняет формат вывода даты и времени.

```
Время модификации файла этой страницы в секундах с 01.01.1970:  
<!--#config timefmt="%s"-->  
<!--#flastmod file="page.shtml"-->  
Время модификации файла этой страницы в читабельном виде:  
<!--#config timefmt="%d.%m.%Y %H:%M:%S"-->  
<!--#flastmod file="page.shtml"-->
```

Вывод:

```
Время модификации файла этой страницы в секундах с 01.01.1970: 1015522642  
Время модификации файла этой страницы в читабельном виде: 07.03.2002 20:37:22
```

Параметры, используемые в **config timefmt**:

Формат	Описание	Пример
%a	Аббревиатура названия дня недели	Sun
%A	Полное название дня недели	Sunday
%b	Аббревиатура названия месяца	Jan
%B	Полное название месяца	January
%d	День месяца	01 (не 1)
%D	Дата в формате "%m/%d/%y"	01/31/90
%e	День месяца	1
%H	Часы в 24-часовом формате	13
%I	Часы в 12-часовом формате	01
%j	День года	235
%m	Номер месяца	01
%M	Минуты	03
%p	АМ РМ	АМ
%r	Время в формате "%I:%M:%S %p"	11:35:46 PM
%S	Секунды	34
%s	Время в секундах с 01.01.1970	957228726
%T	Время в формате "%H:%M:%S"	14:05:34
%U	Неделя года	49
%w	Номер дня недели	5
%y	Год в формате ГГ	95
%Y	Год в формате ГГГГ	1995
%Z	Временная зона	MSK

if/else

Применяется для управления выводом страницы по условию. Синтаксис такой:

```
<!--#if expr="УСЛОВИЕ1" -->
    HTML-код, который будет выводиться, если УСЛОВИЕ1 истинно
<!--#elif expr="УСЛОВИЕ2" -->
    HTML-код, который будет выводиться, если УСЛОВИЕ1 ложно,
    а УСЛОВИЕ2 истинно
<!--#else -->
    HTML-код, который будет выводиться, если все условия ложны
<!--#endif -->
```

Условие – это либо строка, которая является истинной, если непустая, или набор операторов сравнения строк. Операторы могут быть =, !=, <, <=, > и >. Если вторая строка заключена в "/"(слэши), то условие истинно, если в первой строке встречается хоть одно вхождение второй строки. Можно объединять несколько операторов сравнения с помощью операторов &&(И) и ||(ИЛИ). Для группирования условий используются "("(скобки).

Например, условный оператор для определения типа браузера:

```
<!--#if expr="$HTTP_USER_AGENT=/MSIE/" -->
    Internet Explorer
<!--#elif expr="$HTTP_USER_AGENT=/Mozilla/" -->
    Netscape Navigator
<!--#else -->
    Неизвестный (<!--#echo var="HTTP_USER_AGENT" -->)
<!--#endif -->
```

В зависимости от типа браузера посетителя будет выведено либо «Internet Explorer», либо «Netscape Navigator», если тип браузера определить не удалось, то будет выведено слово «Неизвестный» и в скобках будет указано значение переменной окружения \$HTTP_USER_AGENT

printenv

Выводит все переменные окружения. Параметров не имеет.

```
<!--#printenv -->
```

WWW сервер, который понимает технологию SSI, как правило, использует описанные ниже переменные окружения. Кроме того, если на сервере разрешено выполнение исполняемых модулей (CGI Script программы), то в документе могут быть использованы и переменные окружения CGI.

Переменные окружения сервера

При отправке запроса от браузера, на веб-сервер также пересылается техническая информация об определенных параметрах браузера и операционной системы. Веб-сервер в свою очередь одновременно с запрашиваемой информацией возвращает и некоторые свои параметры. Таким образом, браузер и веб-сервер обмениваются данными, которые называются переменные окружения. Эти переменные можно применять в своих целях и отображать их на веб-странице.

1. **"DOCUMENT_NAME"** локальное имя документа.
2. **"DOCUMENT_URI"** локальный путь к документу от базовой директории WWW сервера.
3. **"QUERY_STRING_UNESCAPED"** Строка, полученная от клиента, содержащая все shell-special characters escaped with %
4. **"DATE_LOCAL"** Текущая локальная дата и время.
5. **"DATE_GMT"** Дата и время по Гринвичу (Greenwich).
6. **"LAST_MODIFIED"** Дата последней модификации текущего документа.
7. **"REMOTE_ADDR"** IP адрес удаленного клиента.
8. **"QUERY_STRING"** Строка, полученная от клиента.
9. **"SERVER_SOFTWARE"** Имя HTTP server software.
10. **"SERVER_NAME"** Имя компьютера, на котором работает WWW сервер.
11. **"GATEWAY_INTERFACE"** Имя и версия Common Gateway Interface served WWW (HTTP) сервера (name/version).
12. **"SERVER_PROTOCOL"** Имя и версия HTTP сервера (name/version).
13. **"SERVER_PORT"** IP порт WWW (HTTP) сервера.
14. **"REQUEST_METHOD"** Тип метода запроса к серверу.
15. **"PATH_INFO"** Виртуальный путь, указанный в запросе (путь от базовой директории WWW сервера).
16. **"PATH_TRANSLATED"** Полный путь, указанный в запросе.
17. **"SCRIPT_NAME"** Имя программы для выполнения в CGI запросе.
18. **"REMOTE_HOST"** Имя компьютера удаленного клиента.
19. **"AUTH_TYPE"** Переменная для определения авторизованного метода доступа к серверу (authentication method).
20. **"REMOTE_USER"** Имя пользователя для авторизованного метода доступа.
21. **"REMOTE_IDENT"** Имя удаленного клиента, используемое для идентификации пользователя, согласно спецификации RFC931.
22. **"CONTENT_TYPE"** Тип передачи данных от клиента по методам POST или PUT.
23. **"CONTENT_LENGTH"** Длина в байтах переданных данных по методам POST или PUT.
24. **"HTTP_ACCEPT"** Список, разделенный запятыми, MIME типов, понимаемых просмотрщиком клиента.
25. **"HTTP_USER_AGENT"** Имя просмотрщика клиента (browser software).
26. **"HTTP_REFERER"** URL адрес HTML документа из которого сделан запрос клиентом.
27. **"HTTP_FROM"** Имя (подобное имени E-mail address) удаленного клиента.
28. **"HTTP_FORWARDED"** Имя Proxy Server, через который общается клиент.
29. **"ACCEPT_LANGUGE"** Список языков доступных для компьютера клиента.
30. **"HTTP_COOKIE"** Содержание ответа клиента на запрос от сервера

Примеры использования SSI

Теперь давайте рассмотрим реальный пример применения SSI для формирования сложного документа из нескольких SSI-вставок. Вначале напишем текст основного HTML-документа, полагая, что SSI-вставки находятся в каталоге «ssi»

index.shtml

```
<!--#set var="title" value="Что такое SSI?" -->
<!--#set var="keywords" value="SSI, SHTML, CGI, Apache" -->
<!--#set var="description" value="Пример использования SSI." -->
<!--#include virtual="ssi/header.shtml" -->
Здесь находится текст нашей странички.
<!--#include virtual="ssi/footer.shtml" -->
```

Теперь напишем код для этих SSI-вставок:

header.shtml

```
<html>
<head>
<title><!--#echo var="title" --></title>
<meta name="keywords" content="<!--#echo var="keywords" -->">
<meta name="description" content="<!--#echo var="description" -->">
</head>
<body>
```

footer.shtml

```
</body>
</html>
```

Как видите, основной документ предельно упрощен и состоит из директив, устанавливающих значения переменных title, keywords и description, которые и будут подставлены в код странички при обработке SSI-вставок, определяющих код для верхней и нижней частей странички. Реальный код SSI-вставок обычно гораздо сложнее и может включать в себя большее количество определяемых переменных и сложных условий, формирующих окончательный вид странички.

Первое преимущество SSI с точки зрения дизайнера заключается в том, что при таком подходе web-мастеру, занимающемуся поддержкой сайта, можно не бояться случайно испортить дизайн. Элементы сложной верстки скрыты за счет использования SSI, и поддержка содержимого страничек становится гораздо более легким и приятным делом.

Второе, не менее важное преимущество, – это возможность мгновенной замены дизайна сайта, не требующая переделывания страничек с информационным содержанием сайта. Для смены дизайна достаточно переписать SSI-вставки, формирующие внешний вид сайта. В нашем случае это файлы **header.shtml** и **footer.shtml**.

Версия страницы для печати

Часто возникает прикладная задача – красивый многоколоночный дизайн с верхней и нижней шапками, туча баннеров, но при печати все это не нужно – лишняя бумага, ненужная информация... Поэтому хочется сделать простой альтернативный вид страницы специально для печати. Чтобы это проделать, достаточно подготовить два варианта верхней и нижней шапок, один для экранного отображения, другой – для печати. В качестве переключения между этими вариантами используем переменную QUERY_STRING. Ниже приведены принципиальные структуры для самой страницы (**file.html**) и для верхней и нижней шапок (**top.html** и **bottom.html**).

Структура самой страницы (**file.html**):

```
<!--#include virtual="top.html" -->
здесь тело документа
<!--#include virtual="bottom.html" -->
```

Структура **top.html** и **bottom.html**

```
<!--#if expr="$QUERY_STRING = /for_printing/" -->
    шапка для печати
<!--#else -->
    шапка для просмотра
<!--#endif -->
```

Ссылка на каждой странице должна быть вида

```
<a href=<!--echo var="$DOCUMENT_URI" -->?for_printing>версия для печати</a>
```

Один шаблон отображения – разное содержание

Часто шаблоны используют таким образом: есть только один файл, который описывает структуру страницы, а основное содержание включается директивой:

```
<!--#include virtual="$QUERY_STRING.html"-->
```

ссылки, соответственно, будут иметь вид:

```
href="www.your_domain.ru/index.html?page1"
href="www.your_domain.ru/index.html?page2"
...
```

Проблема возникает, если пользователь набирает адрес непосредственно `http://www.your_domain.ru`, т.е. `QUERY_STRING=""`

Решение:

```
<!--#if expr="$QUERY_STRING" -->
    <!--#include virtual="$QUERY_STRING.html"-->
<!--#else -->
    <!--#include virtual="default.html"-->
<!--#endif -->
```

где **default.html** – страница корневого индекса (оглавления) и просто заглушка.