SSI

http://htmlbook.ru/webserver/ssi19.09.2011

Большинство страниц на сайте, несмотря на их разное содержание, имеет одинаковую структуру кода. Например, верхняя и нижняя часть документа практически не меняется от страницы к странице. В таком случае рекомендуется разделить шаблон страницы на несколько файлов, которые подключаются по мере необходимости. Однако традиционный HTML не позволяет делать подобные кунштюки, поэтому помочь здесь могут серверные языки вроде PHP, Python, Ruby и др. Но для большинства начинающих веб-разработчиков эти названия звучат как неведомые заклинания, они ещё не готовы заниматься веб-программированием. В таком случае, как альтернатива, подойдёт SSI.

SSI (**Server-Side Includes**, включения на стороне сервера) позволяет добавлять контент во множество страниц, причём незаметно для пользователя. Это значит, что при запросе документа браузеру передаётся уже готовый, полностью сформированный код. Особенностью SSI является то, что это технология работает только под управлением веб-сервера и представляет собой набор команд, вставляемых в HTML-файл.

Чтобы веб-сервер отличал рядовые HTML-файлы от SSI-файлов, им дают расширение .shtml. Конечно, можно указать, чтобы делалась проверка всех файлов на поиск специальных директив, но в таком случае возрастёт нагрузка на веб-сервер и несколько увеличится время загрузки файлов.

Вначале надо распорядиться, чтоб веб-сервер обрабатывал файлы с расширением .shtml. Для Apache создаём в корне сайта файл .htaccess и в нём пишем следующую строку.

```
        AddHandler server-parsed .shtml
```

Причём расширение .shtml является стандартным, так что всё должно работать и без этой команды.

Теперь проверяем, как это действует. Делаем два файла — index.shtml будет содержать директиву SSI, а внутри content.html хранится заголовок сайта. Содержание этих файлов представлено в примерах 1 и 2.

Пример 1. Файл index.shtml

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <!--#include file="content.html"-->
</body>
</html>
```

Пример 2. Файл content.html

```
<h1>Работает!</h1>
```

В примере 1 содержимое файла content.html встраивается в файл index.shtml в том месте, где стоит команда `<!--#include file="content.html"-->`. Обратите внимание, что файл content.html не содержит никаких тегов вроде <html> и <body>, иначе они также будут добавлены в исходный документ. По сути, это обычный текстовый документ, в котором имеются HTML-теги.

Если посмотреть итоговый код документа, то мы увидим следующее (пример 3).

Пример 3. Код, полученный в результате использования SSI

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <h1>Работает!</h1>
 </body>
</html>
```

Если всё сделано правильно, то после запуска файла index.shtml, вы увидите надпись «Работает!». В том случае, когда написано нечто другое или вообще ничего нет, возможны два варианта.

1. Отображается надпись [an error occurred while processing this directive]. Это означает, что SSI работает, но в коде содержится ошибка. Например, добавлены лишние пробелы или указанного файла нет.
2. Ничего не отображается. Следует посмотреть исходный код документа, если в нем видна строка `<!--#include file="content.html"-->`, значит веб-сервер не поддерживает SSI или оно не распространяется на расширение .shtml. Также вполне возможно, что файл запускается не под сервером, посмотрите, есть ли в адресной строке браузера http://.

Все упомянутые комплекты веб-серверов поддерживают SSI на исходном уровне, так что если страница как в примере 3 не отображается, необходимо проверить, что веб-сервер запущен и документ открывается в браузере под его управлением. Так, для домена test.lc открывать надо адрес http://test.lc, а не file:///W:/html/test.lc/www/index.shtml.

Возможности SSI не ограничены добавлением содержимого другого файла. С помощью SSI можно запускать серверные приложения, использовать переменные окружения, указывать размер файла, дату модификации документа и многое другое.

Директивы SSI

SSI поддерживает несколько команд называемых директивы, предназначенных для разных целей и расширяющих возможности по модификации веб-страниц.

Все директивы записываются в следующем виде.

```
<!--#директива параметр="значение"-->
```

Каждая директива начинается с ключевого набора <!--#, пробелы между символами не допускаются. После символа решетки идёт указание директивы, а возможные параметры указываются через пробел.

Имена директив, которые используются в SSI, описаны далее.

Директива config

Позволяет управлять некоторыми опциями SSI, такими как настройка формата вывода даты, времени, размера файла и установка текста сообщения об ошибке.

Параметр errmsg

errmsg устанавливает сообщение, отображаемое при возникновении ошибок. По умолчанию выводится текст [an error occurred while processing this directive], но вы можете поменять его на свой, да ещё написав по-русски. Синтаксис следующий.

```
<!--#config errmsg="Сообщение об ошибке"-->
```

Параметр timefmt

Параметр timefmt устанавливает формат даты и времени для директивы flastmod. Синтаксис следующий.

```
<!--#config timefmt="формат"-->
```

Для контроля выводимой информации могут применяться следующие шаблоны.

- %a — сокращенное название дня недели
- %A — полное название дня недели.
- %b — сокращенное название месяца.
- %B — полное название месяца.
- %c — формат даты и времени по умолчанию.
- %d — день месяца (от 01 до 31).
- %D — дата в формате мм/дд/гг.
- %e — день месяца без ведущего нуля (от 1 до 31).
- %H — часы в 24-часовом формате (от 00 до 23).
- %I — часы в 12-часовом формате (от 00 до 12).
- %j — день года (от 001 до 366).
- %m — номер месяца (от 01 до 12).
- %M — минуты (от 00 до 60).
- %p — выводит AM или PM в зависимости от времени и заданного формата.
- %r — время с автоматическим добавлением AM или PM.
- %T — время в формате чч:мм:сс.
- %y — год (от 00 до 99).
- %% — вывод символа %.

Разрешается совмещать между собой любые шаблоны, а также писать дополнительные символы. В примере 1 показано использование параметра timefmt.

Пример 1. Вывод даты и времени модификации файла

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <!--#config timefmt="Дата: %d-%m-%y, время: %T"-->
  <!--#flastmod file="gtm.css"-->
 </body>
</html>
```

В результате данного примера получим строку вида «Дата: 04-07-97, время: 19:24:08».

Параметр sizefmt

Параметр sizefmt определяет формат вывода размера файла. Синтаксис следующий.

```
<!--#config sizefmt="bytes | abbrev"-->
```

Значение bytes отображает размер файла в байтах (3,152), abbrev - в килобайтах (3k) или мегабайтах (6,1M), в зависимости от объёма документа. В примере 2 показано использование параметра sizefmt.

Пример 2. Формат вывода размера файла

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <!--#config sizefmt="abbrev"-->
  <p>Объем файла musa.mp3 - <!--#fsize file="musa.mp3"--></p>
 </body>
</html>
```

Директива include

Директива include вставляет содержимое другого файла в текущий документ. Файл обязательно должен быть доступен, иначе он не будет показан. У директивы include имеется два параметра: file и virtual.

Параметр file

Указывает путь к файлу относительно текущего документа. Можно использовать как текстовые файлы, так и HTML-документы, а также включать другие SSI-файлы (обычно с расширением .shtml). Синтаксис использования следующий.

```
<!--#include file="URL"-->
```

Параметр virtual

Задает виртуальный путь к документу на сервере. Синтаксис следующий.

```
<!--#include virtual="URL" -->
```

Между параметрами file и virtual есть определенная разница. Если вы применяете путь к документам относительно корня сайта (такой путь характеризуется слешем вначале, например /file.html), то следует использовать параметр virtual. При указании относительного пути, следует воспользоваться параметром file (пример 3).

Пример 3. Путь к файлу

```
<!DOCTYPE html>
 <html>
  <head>
   <meta charset="utf-8">
   <title>SSI</title>
  </head>
 <body>
  <!--#include virtual="/inc/header.html"-->
  <p>...</p>
  <!--#include file="../inc/footer.html"-->
 </body>
</html>
```

Директива echo

Директива echo предназначена для вывода значений переменных и даты, формат которой определяется параметром timefmt директивы config. У echo единственный параметр var, он определяет выводимое значение.

```
<!--#echo var="значение для вывода"-->
```

В примере 4 показано использование директивы echo для вывода переменной окружения. Об этих переменных пойдёт речь далее.

Пример 4. Вывод значения переменной окружения

```
<!DOCTYPE html>
 <html>
  <head>
   <meta charset="utf-8">
   <title>SSI</title>
  </head>
 <body>
  <p><!--#echo var="HTTP_USER_AGENT"--></p>
 </body>
</html>
```

Директива fsize

Директива fsize отображает размер определённого файла. Формат вывода задаётся с помощью параметра sizefmt директивы config. Синтаксис использования следующий.

```
<!--#fsize file="URL" | virtual="URL"-->
```

Путь к файлу определяется с помощью параметров file или virtual, которые имеют те же функции, что и для директивы include.

Директива flastmod

Отображает дату последней модификации указанного файла. Формат даты устанавливается через директиву config и параметр timefmt. Синтаксис следующий.

```
<!--#flastmod file="URL" | virtual="URL"-->
```

Путь к файлу определяется с помощью параметров file или virtual, которые имеют те же функции, что и для директивы include.

Директива exec

Директива exec вставляет результат выполнения команды или CGI-программы в HTML-документ. Эта директива включает два параметра: cmd и cgi.

Параметр cmd

Запускает указанную командную строку с использованием локального интерпретатора.

```
<!--#exec cmd="команда"-->
```

Например, строка `<!--#exec cmd="/usr/bin/date" -->` исполняет Unix-команду date.

Параметр cgi

Выполняет CGI-программу и результат её выполнения вставляет в указанное место. В качестве параметра указывается адрес программы.

```
<!--#exec cgi="URL"-->
```

В примере 5 показано одно из применений параметра cgi.

Пример 5. Использование директивы exec

```
<!DOCTYPE html>
 <html>
  <head>
   <meta charset="utf-8">
   <title>SSI</title>
  </head>
 <body>
  <p>Всего прочитало эту страницу <!--#exec cgi="/counter.pl"-->
  человек.</p>
 </body>
</html>
```

В данном примере запускается программа counter.pl, написанная на Perl, которая подсчитывает число посетителей данной страницы и записывает его в файл или в базу данных. После этого результат отображается в месте, где стоит строка <!--#exec cgi="/counter.pl"-->.

Переменные окружения

Когда браузер запрашивает от веб-сервера документ, он также пересылает на сервер техническую информацию об определённых параметрах браузера и операционной системы. Веб-сервер в свою очередь одновременно с документом возвращает некоторые свои характеристики. 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.

При использовании SSI общий синтаксис вывода определенной переменной окружения будет следующий.

```
<!--#echo var="переменная"-->
```

Некоторые переменные с их описанием перечислены в табл. 1. Заметьте, что все имена пишутся заглавными символами. Хотя это условие и необязательно, именно такая форма записи является традиционной и устоявшейся.

Табл. 1. Список переменных окружения

Переменная	Описание
DOCUMENT_ROOT	Путь к корневой папке сайта. Для локального веб-сервера значение может принимать вид z:/home/htmlbook.ru/www, а в других случаях зависит от операционной системы сервера и используемого программного обеспечения.
GATEWAY_INTERFACE	Версия CGI (Common Gateway Interface, общий шлюзовый интерфейс). Значение обычно равно CGI/1.1.
HTTP_ACCEPT	Типы файлов, которые способен принять браузер. В качестве значения возвращается список поддерживаемых MIME-типов разделенных между собой запятой, например: text/html, application/xhtml+xml.
HTTP_CONNECTION	Тип соединения браузера с веб-сервером. Так, значение keep-alive означает, что браузер поддерживает постоянное соединение с сервером. При этом в течение одного сеанса соединения разрешено делать несколько запросов. Повторного соединения в таком случае уже не происходит.
HTTP_HOST	Доменное имя сайта. Обычно различают имена с префиксом www (www.htmlbook.ru) и без него (htmlbook.ru). Переменная вернёт тот адрес сайта, который указан в адресной строке браузера.

HTTP_REFERER	Адрес страницы, с которой пользователь перешел на данный сайт, он еще называется реферер.
HTTP_USER_AGENT	Идентификатор используемого браузера и операционной системы. В качестве значения возвращается строка, содержащая ключевые слова. Например, следующая строка Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0.2) Gecko/20100101 Firefox/6.0.2 говорит, что пользователь использует браузер Firefox 6.0.2 под операционной системой Windows 7.
QUERY_STRING	Запрос, который указан в адресной строке после вопросительного знака (?). Обычно пишется в форме «переменная=значение», где переменные разделяются между собой амперсандом (&). Так, при написании адреса http://htmlbook.ru/?id=5&slv=34 будет возвращено значение id=5&slv=34.
REMOTE_ADDR	IP-адрес посетителя сайта.
REQUEST_METHOD	Метод отправки данных на сервер. По умолчанию применяется метод GET.
REQUEST_URI	Адрес запрашиваемого документа. Отсчёт ведётся от корня сайта, т.е. для полного адреса http://htmlbook.ru/1.html вернется значение 1.html.
SERVER_ADDR	IP-адрес компьютера, на котором размещается сайт.
SERVER_ADMIN	Адрес электронной почты администратора сайта.
SERVER_NAME	Имя сервера.
SERVER_PORT	Порт, по которому ожидается получение данных.
SERVER_PROTOCOL	Протокол для получения и отправки данных. Значение обычно равно HTTP/1.1.
SERVER_SOFTWARE	Программное обеспечение установленное на сервере. Для веб-сервера Apache возвращается номер версии (Apache/2.2.4), а также версия PHP (PHP/5.3.3).

В примере 1 показано использование переменных окружения для отображения на веб-странице требуемой информации.

Пример 1. Вывод значения переменной DOCUMENT_ROOT

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <p>Путь к корневой папке сайта: <!--#echo var="DOCUMENT_ROOT"--></p>
 </body>
</html>
```

В результате выполнения примера будет выведена следующая строка : Путь к корневой папке сайта: /home/htmlbook.ru/www.

Значения переменных окружения можно посмотреть с помощью программы на PHP, используя функцию phpinfo(), как показано в примере 2.

Пример 2. Использование phpinfo()

```
<?php
  phpinfo();
?>
```

В результате выполнения программы будет выведена таблица с разными параметрами, в том числе и переменными окружения в разделе «Apache Environment» (рис. 1).

Apache Environment

Variable	Value
HTTP_HOST	htmlbook.lc
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0.2) Gecko/20100101 Firefox/6.0.2
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	ru-ru,ru;q=0.8,en-us;q=0.5,en;q=0.3
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_ACCEPT_CHARSET	UTF-8,*
HTTP_DNT	1
HTTP_CONNECTION	keep-alive
HTTP_COOKIE	SESSf222ab4202606d2bfa8a87c180a3a7de=0004ff1717e1cb86a720871f088d3398; SESSa0519fd8af14afee3d002c00b6012f18=9b8e75233874749b0236117c3b340f0f; DRUPAL_UID=1
PATH	\usr\local\ImageMagick;\usr\local\php5;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\

Рис. 1. Apache Environment

Также можно написать программу на PHP, которая будет выводить все переменные окружения в виде таблицы (пример 3)..

Пример 3. Вывод переменных окружения

```
<?php
 print "<!DOCTYPE html>\n<html>\n<head><title></title></head>\n<body>\n";
 print "<table border=1>\n";
 foreach ($_SERVER as $a => $b) print "<tr><td>$a</td><td>$b</td></tr>\n";
 print "</table>\n";
 print "</body>\n</html>\n";
?>
```

Условные выражения

Сами переменные окружения редко применяются для их вывода на страницу. Гораздо полезнее сравнивать их с некоторым значением и в зависимости от этого сравнения принимать решение о выводе текста. Такое сравнение называется условным выражением и встречается практически во всех языках программирования. Условные выражения есть и в SSI, в общем виде они записываются так.

```
<!--#if expr="условие1"-->
  Если условие1 истинно, то будет выводиться этот текст.
<!--#elif expr="условие2"-->
  Если условие1 ложно, а условие2 истинно, тогда вместо первого текста выводится этот.
<!--#else-->
  Данный текст выводится, если условие1 и условие2 оба ложны.
<!--#endif-->
```

Не обязательно использовать эту конструкцию целиком, вполне можно вставлять её частично, например так.

```
<!--#if expr="условие1" -->
  Если условие1 истинно, что-нибудь вывести.
<!--#endif -->
```

В этом случае если условие выполняется, то будет выводиться текст или код внутри <!--#if--> и <!--#endif-->, в противном случае, т. е. когда условие1 ложно, то конструкция пропускается и текст не отображается.

В табл. 1 перечислены возможные выражения, которые возвращают истину при соблюдении описанных условий.

Табл. 1. Условные выражения

Условие	Описание
str	Строка str не пустая.
!str	Строка str пустая.
str1=str2	Значение str1 равно str2.
str1!=str2	Значение str1 НЕ равно str2.
str1<str2	Значение str1 меньше str2.
str1<=str2	Значение str1 меньше или равно str2.
str1>str2	Значение str1 больше str2.
str1>=str2	Значение str1 больше или равно str2.
str1 && str2	Строка str1 И строка str2 не пустые.
str1 || str2	Строка str1 ИЛИ строка str2 не пустые.

Сложные выражения можно брать в круглые скобки, например, написать так:

```
(str1>0) && (str1<5)
```

Это условие возвращает истину, когда str1 больше нуля и меньше пяти. Под истиной и ложью подразумевается выполнение логического или математического соответствия, так, 1>0 это истина, а 1<0 ложь.

Для использования переменных окружения в условных выражениях перед именем переменной следует поставить знак $. Текст и сравниваемые значения берутся в слэш (/текст/). Если внутри значения встречается символ /, то значение можно экранировать следующим образом.

```
\"/path/to/server\"
'/path/to/server'
```

Текст пишется внутри \" и \" или обрамляется одинарными кавычками. В примере 1 показано использование условия и переменной REMOTE_ADDR для сравнения IP-адреса со значением 127.0.0.1.

Пример 1. IP-адрес

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
```

```
 </head>
 <body>
  <!--#if expr="($REMOTE_ADDR = /127.0.0.1/)"-->
   <p>Вы зашли с локального адреса.</p>
  <!--endif-->
 </body>
</html>
```

SSI на практике

SSI используется не только для вывода каких-либо значений, но и для упрощения построения сайта и добавления некоторых удобных вещей, которых лишён HTML. Рассмотрим некоторые примеры применения SSI в деле.

Ссылка на главную страницу

Обычно ссылку на главной странице на саму себя не делают, чтобы не сбивать пользователей с толку. При этом на остальных страницах такая ссылка есть и она ставится на заголовок сайта. Пусть файл с главной страницей называется index.shtml, тогда нам надо определить, не совпадает ли имя открытого документа с этим и в зависимости от совпадения решаем, делать или нет ссылку. В примере 1 переменная DOCUMENT_URI сравнивается со значением /index.shtml и если оно не равно, иными словами, у нас любая страница кроме главной, тогда картинку делаем ссылкой.

Пример 1. Главная страница

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <!--#if expr="($DOCUMENT_URI != \"/index.shtml\")"-->
   <a href="/"><img src="logo.gif" alt="Логотип"></a>
  <!--#else-->
   <img src="logo.gif" alt="Логотип">
  <!--endif-->
 </body>
</html>
```

Определение Internet Explorer

В некоторых случаях для браузера Internet Explorer необходимо загружать отдельный код, который отличается для других браузеров. Тогда используем переменную HTTP_USER_AGENT и сравниваем её со значением MSIE. Обратите внимание, что если просто вывести HTTP_USER_AGENT, то текст будет совсем другой. В примере 2 показано, как задать один текст для всех версий IE и другой для остальных браузеров.

Пример 2. Определение IE

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
 </head>
 <body>
  <!--#if expr="$HTTP_USER_AGENT = /MSIE/"-->
   <p>Ваш браузер устарел, пожалуйста, замените Internet Explorer на другой.</p>
  <!--#else-->
```

```
   <p>Добро пожаловать на сайт!</p>
  <!--endif-->
 </body>
</html>
```

Создание шаблона

Все страницы сайта строятся, как правило, по одному шаблону и имеют некоторые повторяющиеся элементы вроде шапки и подвала. Чтобы не повторять их в каждом HTML-документе, их код можно вынести в отдельный файл и подключать одной строкой. Это позволяет не редактировать десятки файлов, а ограничиться всего одним.

Предположим, что типовой документ на сайте имеет структуру, показанную в примере 3. Различаться будет только содержимое <article>, а всё остальное кочует из файла в файл.

Пример 3. Типовой файл

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
  <!-- Стилевые файлы и скрипты -->
 </head>
 <body>
  <header>Шапка сайта</header>
  <article>Содержание текущего документа</article>
  <aside>Боковая панель</aside>
  <footer>Подвал</footer>
 </body>
</html>
```

Чтобы не писать каждый раз одно и то же, вынесем похожие фрагменты в текстовые файлы и поместим их в папку assets. Тогда шаблон будет иметь вид, как показано в примере 4.

Пример 4. Шаблон

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>SSI</title>
  <!--#include virtual="/assets/link.txt"-->
 </head>
 <body>
  <!--#include virtual="/assets/header.shtml"-->
  <article>Содержание текущего документа</article>
  <!--#include virtual="/assets/aside.shtml"-->
  <!--#include virtual="/assets/footer.txt"-->
 </body>
</html>
```

Обратите внимание, что файлs header.shtml и aside.shtml имеют расширение .shtml, для того, чтобы в них подключать директивы SSI.

Выделение текущего пункта меню

Аналогично главной странице пункт меню, который указывает на текущую страницу не делают ссылкой, а вставляют как рядовой текст, выделяя его с помощью стилей. В примере 5 показано содержимое файла aside.shtml, который используется в предыдущем примере.

Пример 5. Меню для сайта

```
<ul>
  <!--#if expr="($DOCUMENT_URI = '/1.shtml')"-->
  <li class="active">Эрик Картман</li>
  <!--#else-->
  <li><a href="1.shtml">Эрик Картман</a></li>
  <!--#endif-->
 <!--#if expr="($DOCUMENT_URI = '/2.shtml')"-->
  <li class="active">Кенни Маккормик</li>
  <!--#else-->
  <li><a href="2.shtml">Кенни Маккормик</a></li>
  <!--#endif-->
 <!--#if expr="($DOCUMENT_URI = '/3.shtml')"-->
  <li class="active">Стэн Марш</li>
  <!--#else-->
  <li><a href="3.shtml">Стэн Марш</a></li>
  <!--#endif-->
 <!--#if expr="($DOCUMENT_URI = '/4.shtml')"-->
  <li class="active">Кайл Брофловски</li>
  <!--#else-->
  <li><a href="4.shtml">Кайл Брофловски</a></li>
  <!--#endif-->
</ul>
```

На сайте сделано четыре файла с именами 1.shtml, 2.shtml, 3.shtml и 4.shtml. Чтобы SSI различал, какому пункту меню какой файл соответствует, используется переменная DOCUMENT_URI. Если её значение совпадает с именем открытого документа, то ссылка не добавляется.

Версия для печати

Версия для печати содержит то же самое содержание, что и текущая страница, но специально оптимизированное для печати документа, к примеру, может отсутствовать реклама, какие-то декоративные элементы. Чтобы различать «нормальную» страницу и её версию для печати, к адресу документа добавим ?print, а с помощью SSI будем проверять, есть эта добавка или нет. Поскольку адрес документа может быть произвольным, воспользуемся переменной DOCUMENT_URI для его получения и создания ссылки на печатную версию (пример 6).

Пример 6. Версия для печати

```
<!--#if expr="($QUERY_STRING = 'print')"-->
 <p>Версия для печати</p>
<!--#else-->
 <p>Обычная страница</p>
 <p><a href="<!--#echo var="DOCUMENT_URI"-->?print">Версия для печати</a></p>
<!--#endif-->
```

С помощью переменной QUERY_STRING идёт проверка, есть ли в адресе ключевое слово print или нет. Если оно присутствует, тогда выводится одна версия страницы, если этого ключевого слова нет, тогда отображается другая версия страницы со ссылкой.