ЗАДАНИЯ
к лабораторным работам
по дисциплине «Инфокоммуникации»

Лабораторная работа №1. Создание динамического HTML-документа

Задание:

Создать HTML-документ, который должен содержать:

- форматированный текст;
- многоуровневые нумерованные и ненумерованные списки;
- таблицу;
- изображения;
- гиперссылки на другие HTML-документы, гиперссылки в пределах HTML-документа, гиперссылку на e-mail;
- формы (input (text, checkbox, radio, submit, reset), textarea, select).
- слои;
- скрипты на языке JavaScript (в соответствии с вариантом).

При форматировании HTML-документа использовать каскадные таблицы стилей CSS (Cascading Style Sheets): связанные, внедренные и встроенные. Продемонстрировать приоритетность CSS.

Вариант 1

Калькулятор на четыре действия (с нажимающимися кнопками). Запомнить в cookie результат последнего вычисления и отображать это значение на индикаторе калькулятора при повторном посещении.

Вариант 2

За указателем мыши перемещается некоторое изображение (предусмотреть возможность выбора и смены изображения). Запомнить в cookie имя файла с последним изображением и отображать это изображение при повторном посещении.

Вариант 3

Меню, в котором при наведении указателя мыши пункт меню выделяется другим цветом, при щелчке на пункте меню открывается подменю (предусмотреть возможность выбора и смены цветов меню). Настроить и запомнить в cookie основной и дополнительный цвета меню и использовать эти цвета при повторных посещениях.

Вариант 4

Просмотр набора изображений со сменой подписей к изображениям с помощью кнопок «Назад» и «Далее». При просмотре первого изображения блокируется кнопка «Назад», при просмотре последнего – кнопка «Далее». Настроить и запомнить в cookie шрифт для подписей к изображению и использовать этот шрифт при повторных посещениях.

Например:

Фото 1. Фудзияма.

Вариант 5

Сборка мозаики. Элементы мозаики перетаскиваются указателем мыши или щелчком указывается положение того или иного фрагмента в мозаике. Запомнить в cookie состояние мозаики и восстановить его при повторном посещении.

Содержание отчета:

- титульный лист;
- задание;
- исходные тексты (CSS и скрипт);
- выводы по работе.

Методические указания:

Язык разметки гипертекста HTML (HyperText Markup Language)

HTML (HyperText Markup Language) - язык разметки гипертекста, используемый для создания документов, независимых от аппаратно-программной платформы. HTML - это не язык программирования, а описательный язык разметки.

HTML-документ состоит из текста, который представляет собой содержимое документа, и тегов, которые определяют структуру и внешний вид документа при его отображении броузером. Структура HTML-документа проста:

```
<html>
<head>
<title>HTML-документ</title>
</head>
<body>
<i>Internet</i>
</body>
</html>
```

Текст всего документа заключается в теги <html>. Текст документа состоит из заголовка и тела, которые выделяются соответственно тегами <head> и <body>. В заголовке указывают название HTML-документа и другие параметры, которые броузер будет использовать при отображении документа. Тело - это та часть, в которую помещается собственно содержимое HTML-документа. Тело включает предназначенный для отображения текст и управляющую разметку документа (теги), которые используются броузером. Теги содержат указания о способе отображения текста. С помощью тегов, кроме того, создаются ссылки на файлы, содержащие дополнительные данные (графику, звук), и обозначаются точки привязки (гиперссылки или якоря), посредством которых данный документ связывается с другими документами.

HTML-тег состоит из имени, за которым может следовать необязательный список атрибутов тега. Текст тега заключается в угловые скобки < >.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами конца строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута - одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в кавычки, особенно если они содержат несколько разделенных пробелами слов.

Регистр символов в именах тегов и атрибутов не учитывается.

Как правило, теги состоят из начального и конечного элементов, между которыми размещаются текст и другие элементы документа. Имя конечного тега совпадает с именем начального, но перед именем конечного тега ставится косая черта / (<html>…</html>).

Конечные теги никогда не содержат атрибутов.

При использовании вложенных тегов их нужно закрывать, начиная с самого последнего и двигаясь к первому.

Некоторые теги не имеют конечного элемента.

В некоторых случаях конечные теги можно опускать. Тем не менее, рекомендуется включать конечные теги, чтобы избежать ошибок при воспроизведении документа.

HTML предоставляет множество тегов, которые могут использоваться при создании документов. Ниже в таблице приведены некоторые из них.

Тег	**Описание**
<html>…</html>	начало и конец всего документа
<head>…</head>	начало и конец заголовка документа
<title>…</title>	заголовок документа
<body>…</body>	начало и конец тела документа
…	отображение текста курсивом
…	отображение текста жирным шрифтом
<hn>…</hn> n=1…6	заголовок уровня n
…	установка шрифта, его цвета и размера
<table>…</table>	таблица
… … …	нумерованный список
… … …	ненумерованный список
<marquee>...</marquee>	бегущая строка
<img>...</img>	вставка изображения
<a>...	гиперссылка
<form>...</form>	форма
...	оформление абзаца
 	перевод строки
<hr>...</hr>	горизонтальная линия

Каскадные таблицы стилей (CSS – Cascading Style Sheets)

Каскадные таблицы стилей – важная часть разработки Web-приложений. Каскадные таблицы стилей определяют макет HTML-документа в формате, отделенном от собственно информационного наполнения HTML-документа.

Стили можно реализовать тремя способами:

- связанные таблицы стилей (Linked Style Sheets) – таблица стилей определяется в отдельном текстовом файле с расширением .css и ее стиль связывается с одной или несколькими страницами. Связанные стили воздействуют на отдельный Web-узел;

– внедренные таблицы стилей (Global Style Sheets) – стили можно внедрить непосредственно в текст на HTML. Внедренные стили воздействуют на отдельную страницу;
– встроенные таблицы стилей (Inline Style Sheets) – встроенные стили создаются с помощью атрибута style. Встроенные стили воздействуют на отдельный тег.

Пример связанной таблицы стилей:
Таблица стилей (в файле lss.css)

```
h1 {font-size:10; color:red}
p {color:#0000ff; font-style:italic}
```

HTML-документ

```
<html>
<head><link href=lss.css rel=stylesheet></head>
<body>
<h1>Заголовок нового стиля</h1>
<p>Абзац нового стиля</p>
</body>
</html>
```

Пример внедренной таблицы стилей:

```
<html>
<head><style><!h2 {font-weight:bold; color:green}></style></head>
<body><h2>Заголовок нового стиля</h2></body>
</html>
```

Пример встроенной таблицы стилей:

```
<html>
<h1 style="font-size:40pt; color:blue">Заголовок нового стиля</h1>
<p style="font-size:40; color:magenta">Абзац нового стиля</h1>
</html>
```

Слои

Слой – это некий прямоугольный элемент, содержащий в себе любую разметку HTML. Слоем может быть как простая строка текста, так и сложная форма, сверстанная в таблице.

С помощью JavaScript можно изменять размеры слоя, его видимость, перемещать слой и т.п.

В общем случае слой – это часть HTML-документа, выделенная тегом div, которому присвоен некоторый идентификатор id:

```
<div id=layer1>
…
</div>
```

Также необходимо, чтобы слой был описан с помощью стилевых таблиц:

```
<style type=text/css>
#layer1 {position:absolute; top=0; left=0; z-index=1; visibility:visible; width:100px; height:120px;}
</style>
```

С помощью стилевых таблиц описываются следующие параметры слоя:

– position –точка отсчета координат положения слоя, возможные значения: absolute и relative;
– top, left – координаты верхнего левого угла слоя;
– z-index – уровень слоя;

– visibility – видимость слоя, возможные значения: visible и hidden;
– width; height – ширина и высота слоя.

Пример скрытия и отображения слоя:

```
<html>
<head>
<script language=JavaScript>
function showlayer(layername)
{eval('document.all[" '+layername+' "].style.visibility="visible" ');}
function hidelayer(layername)
{eval('document.all[" '+layername+' "].style.visibility="hidden" ');}
</script>
</head>
<body>
<style type=text/css>
#mylayer {position:absolute; top:0; left:400; z-index:1; visibility:visible; width:100px;
height:100px;}
</style>
<div id=mylayer>
<img src=dove.gif border=1>
</div>
<p><button onclick="showlayer('mylayer');">Показать слой</button>
<p><button onclick="hidelayer('mylayer');">Спрятать слой</button>
</body>
</html>
```

Пример перемещения слоя:

```
<html>
<head>
<script language=JavaScript>
function movelayer(layername,newtop,newleft)
{eval('document.all[" '+layername+' "].style.pixelTop=newtop');
eval('document.all[" '+layername+' "].style.pixelLeft=newleft');}
</script>
</head>
<body>
<style type=text/css>
#mylayer {position:absolute; top:0; left:400; z-index:1; visibility:visible; width:100px;
height:100px;}
</style>
<div id=mylayer>
<img src=dove.gif border=1>
</div>
<p><button onclick="movelayer('mylayer',0,600);">Переместить слой</button>
</body>
</html>
```

Объектная модель броузера

Объектная модель - это набор связанных между собой объектов, обеспечивающих доступ к содержимому страницы и ряду функций броузера.

Объект window

Объект window находится в вершине иерархии и является контейнером для других объектов. Он представляет текущее окно броузера.

Свойства объекта window

Свойство	Описание
parent	возвращает родительское окно для данного окна
self	возвращает ссылку на текущее окно
top	возвращает ссылку на самое ближнее к пользователю окно
name	возвращает имя окна, заданное тегом <frameset>
opener	возвращает окно, создавшее данное окно
closed	указывает на то, что окно закрыто
status	задает текст, отображаемый в строке состояния броузера
defaultStatus	возвращает текст, отображаемый в строке состояния броузера
returnValue	позволяет событию или диалоговой панели возвращать значение
document	возвращает ссылку на объект document
event	возвращает ссылку на глобальный объект event
history	возвращает ссылку на объект history
location	возвращает ссылку на объект l ocation
navigator	возвращает ссылку на объект navigator
screen	возвращает ссылку на глобальный объект screen

Методы объекта window

Методы, предоставляемые объектом window, позволяют управлять самим окном, а также выполнять ряд действий внутри него.

Методы open и close

Для открытия нового окна можно воспользоваться методом open. Полный синтаксис метода open выглядит следующим образом:

newWnd=window.open(URL, name, features, replace),

где:

- URL адрес документа, отображаемого в новом окне. Если адрес не задан, отображается пустое окно;
- name строка, задающая имя окна;
- features строка, задающая параметры нового окна;
- replace - указывает, замещает ли новое окно текущее в списке history или нет.

Параметр features	Значение	Описание
fullscreen	yes|no|1|0	Полноэкранное или обычное окно (по умолчанию обычное)
channelmode	yes|no|1|0	Отображение полосы каналов
toolbar	yes|no|1|0	Отображение панели инструментов
location	yes|no|1|0	Отображение адресной строки
directories	yes|no|1|0	Отображение панели ссылок
status	yes|no|1|0	Отображение строки состояния
menubar	yes|no|1|0	Отображение строки меню
scrollbars	yes|no|1|0	Отображение линеек прокрутки
resizeable	yes|no|1|0	Разрешение изменения размера окна
width	число	ширина окна в пикселах (min 100)
height	число	высота окна в пикселах (min 100)
top	число	Вертикальная координата верхнего левого угла окна
left	число	Горизонтальная координата верхнего левого угла окна

Закрыть окно позволяет метод close. Синтаксис метода close выглядит следующим образом:

newWnd.close()

Для закрытия текущего окна можно воспользоваться одним из двух способов: window.close() или self.close().

Методы alert, prompt, confirm

Эти методы позволяют отображать различные диалоговые панели.

window.alert(«Сообщение») выводит строку и ожидает, когда пользователь щелкнет кнопку OK.

string=window.prompt(«Вопрос», «Значение по умолчанию») служит для ввода информации. Введенная пользователем строка возвращается при щелчке на кнопке OK. При щелчке на кнопке Cancel возвращается значение null.

truefalse=window.confirm(«Вопрос») используется для получения подтверждения. При щелчках на кнопках OK и Cancel возвращается true или false соответственно.

Методы focus, blur

С помощью этих методов можно программно перемещаться между несколькими открытыми окнами и изменять текущее активное окно. Метод blur перемещает фокус из одного окна в другое (аналогично клавише Tab), метод focus перемещает фокус на окно, где находится исполняемый код, написанный на JavaScript.

Методы setTimeout, setInterval, clearTimeout, clearInterval

Методы setTimeout, setInterval используются для управления таймером. Метод setTimeout создает таймер, который выполняет указанные действия по истечении заданного числа миллисекунд, например

window.setTimeout(«действие», миллисекунды).

Для выполнения действий, повторяющихся с определенным интервалом времени, используется метод setInterval, например

window.setInterval(«действие», интервал в миллисекундах).

Методы clearTimeout, clearInterval отменяют действие методов setTimeout, setInterval соответственно.

События объекта window

Событие	Описание
onBeforeUnload	возникает перед выгрузкой страницы
onBlur	возникает при потере фокуса
onError	возникает при ошибке
onFocus	возникает при получении фокуса
onHelp	возникает при нажатии клавиши F1
onLoad	возникает в момент загрузки страницы
onResize	возникает при изменении размеров окна
onScroll	возникает при прокрутке содержимого окна
onUnload	возникает непосредственно перед выгрузкой страницы

Объект history

Объект history содержит информацию объект адресах страниц (в формате URL), которые посещались в данной сессии. Данный объект имеет одно свойство length и три метода. Используя методы объекта, можно перемещаться по списку history вперед и назад.

Метод	Описание
back	загружает предыдущую страницу из списка history
forward	загружает следующую страницу из списка history
go(n)	загружает n-ю страницу из списка history

Объект navigator

Объект navigator обеспечивает получение информации о броузере.

Свойство	Описание

appCodeName	кодовое имя броузера
appName	название броузера
appVersion	версия броузера
userAgent	часть заголовка, посылаемого Web-серверу
javaEnabled	включена ли поддержка языка Java
cookieEnabled	включена ли поддержка cookies

Объект screen

Для получения информации о клиентском броузере используются значения свойств объекта screen.

Свойство	Описание
colorDepth	максимальное число цветов, поддерживаемых в данной системе
height	высота экрана в пикселах
width	ширина экрана в пикселах
pixelDepth	число бит на пиксел
updateInterval	временной интервал обновления экрана

Объект document

Для работы с документами HTML используется объект document. Пользуясь его свойствами и методами, можно получить информацию о текущем документе, загруженном в окно броузера, а также управлять отображением содержимого этого документа.

Свойство	Описание
alinkColor	цвет активной ссылки
anchors	массив локальных меток, размещенных в документе (метки используются для организации ссылок внутри документа)
applets	массив объектов, соответствующих апплетам Java, расположенным в документе
bgColor	цвет фона
cookie	значение cookie для данного документа
embeds	массив plug-in объектов, содержащихся в документе
fgColor	цвет текста
forms	массив, содержащий в виде объектов все формы, расположенные в документе
images	массив растровых изображений, включенных в документ
lastModified	дата последнего изменения документа
linkColor	цвет ссылки
links	массив, содержащий все ссылки в документе
location	полный адрес URL документа
referrer	адрес URL страницы, ссылающейся на текущую
title	заголовок документа
URL	полный адрес URL документа
vlinkColor	цвет ранее посещавшейся ссылки

Метод	Описание
open	открытие потока вывода в новое окно броузера
write	вывод указанного текста в окно броузера
writeln	вывод указанного текста в окно броузера с переводом строки
close	закрытие потока вывода
clear	очистка содержимого выбранной области
createElement	создание экземпляра объекта для указанного тега
elementFromPoint(x ,y)	элемент, находящийся в указанных координатах

Событие	Описание
onCllick	возникает при щелчке одной из кнопок мыши
onDblClick	возникает при двойном щелчке одной из кнопок мыши
onMouseDown	возникает при нажатии одной из кнопок мыши
onMouseMove	возникает при перемещении указателя мыши
onMouseOut	возникает при выводе указателя мыши из области элемента
onMouseOver	возникает при попадании указателя мыши в область элемента
onMouseUp	возникает при отпускании ранее нажатой кнопки мыши
onDragStart	возникает при перетаскивании элемента
onSelectStart	возникает при выборе содержимого элемента
onKeyDown	возникает при нажатии клавиши
onKeyPress	возникает при нажатии и удерживании клавиши
onKeyUp	возникает при отпускании заранее нажатой клавиши
onKeyHelp	возникает при нажатии клавиши F1 или аналогичной для получения справки

Объект date

Объект Date и его методы используются для работы с датой и временем. Дата в языке JavaScript представляется так же, как и языке Java - это число миллисекунд, прошедших с 1 января 1970 года. Для создания экземпляра объекта Date используется конструктор new:

myDate=new Date()

Метод	Описание
getDate	возвращает день месяца как целое число от 1 до 31
getDay	возвращает день недели как целое число от 0 (воскресенье) до 6 (суббота)
getMonth	возвращает номер месяца как целое число от 0 (январь) до 11 (декабрь)
getYear	возвращает две последние цифры года
getTime	возвращает число миллисекунд между 1 января 1970 года, 00:00:00 и датой, заданной объектом Date
getHours	возвращает число часов как целое от 0 до 23
getMinutes	возвращает число минут как целое от 0 до 59
getSeconds	возвращает число секунд как целое от 0 до 59
setDate	устанавливает день месяца
setMonth	устанавливает номер месяца
setYear	устанавливает год
setTime	устанавливает время
setHours	устанавливает число часов
setMinutes	устанавливает число минут
SetSeconds	устанавливает число секунд

Использование cookie

Cookie – это механизм, позволяющий серверу хранить информацию на клиентском компьютере и при необходимости извлекать ее. С помощью механизма cookie сервер может хранить на клиентском компьютере некоторый именованный информационный элемент. Это может быть имя пользователя, информация о настройках, служебная информация, используемая в данной сессии и т.п. Обычно данный механизм применяется для сохранения информации, введенной пользователем. На каком-то узле пользователь вводит свои данные в поля формы, она отсылается на сервер, и информация при этом сохраняется на компьютере пользователя.

Механизм cookies поддерживается с помощью свойства cookie объекта document. Минимально должно быть установлено значение атрибута name.

Атрибут	Описание
name=value;	каждый информационный элемент хранится в виде пары name=value; name задает название элемента, value - его значение
expires=date	задает «срок годности» информационного элемента: если этот атрибут не указан, срок годности истекает при закрытии броузера, установка срока годности, равного дате в будущем, приводит к сохранению элемента, равного дате в прошлом - удалению элемента (дата указывается в формате GMT)
domain= domainname	задает имя домена, из которого «видно» содержимое данного информационного элемента
path=path	задает маршрут, на котором «видно» содержимое данного информационного элемента
secure	задает защищенность информации

Следующий пример демонстрирует создание cookies.

```
<html>
<head>
<title>Cookies</title>
<script language=JavaScript>
function doCookie()
{myname="myname=";
if (document.cookie != -1)
{value=document.cookie;
alert("Hello, "+value)
}
else
{name=prompt("What is your name?", "I don't no");
document.cookie=myname+name+";";
}
}
</script>
</head>
<body onLoad="doCookie();">
</body>
</html>
```

В следующем примере запрашивается имя пользователя при первом посещении, сохраняется в виде информационного элемента, при последующих посещениях отображается в виде приветствия.

```
<html>
<head>
<title>Cookies</title>
<script language=JavaScript>
function doCookie()
{myName="myName=";
if (document.cookie.indexOf(myName) != -1)
{start=document.cookie.indexOf(myName);
end=document.cookie.indexOf(";");
value=document.cookie.substring(start+myName.length, end);
alert("Hello, "+value)
}
else
{name=prompt("What is your name?", "I don't no");
document.cookie=myName+name+";";
```

```
}
}
</script>
</head>
<body onLoad="doCookie();">
</body>
</html>
```

Использование графики

Управление графикой с помощью JavaScript базируется на доступе к коллекции images и управлении свойствами отдельных элементов этой коллекции.

Атрибут name позволяет обращаться к графическому изображению по имени. Например, изображение, описанное как <img src="First.gif" name="First">, доступно из JavaScript как document.first.

Коллекция images содержит все графические изображения, включенные в состав данного HTML документа. для доступа к первому элементу можно обратиться к 0-му элементу коллекции: document.images[0].

Таким образом, если описанное выше графическое изображение было первым, можно обратиться к нему одним из следующих способов:

```
document.images[0]
document.images["first"]
document.first
document["first"]
```

Объект image

Объект image может использоваться для задания свойств графических изображений, включенных в состав данной страницы, а также для загрузки изображений в кэш-память и их последующего отображения.

В следующей таблице перечислены свойства объекта image

Свойство	Описание	Только чтение	Только <img>
border	атрибут border тега img	да	да
complete	булево значение, указывающее, загружено изображение или нет	да	нет
height	высота изображения	да	нет
hspace	атрибут hspace тега img	да	да
lowsrc	атрибут lowsrc тега img	нет	нет
name	атрибут name тега img	нет	да
prototype	позволяет добавлять свойства к объекту image	-	нет
src	атрибут src тега img	нет	нет
vspace	атрибут vspace тега img	да	да
width	атрибут width тега img	да	нет

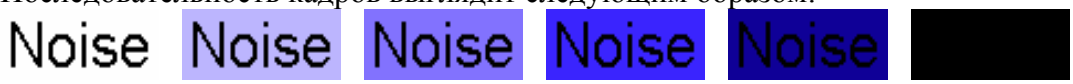
Создание анимационных изображений

С помощью динамической смены растровых изображений в сценарии JavaScript можно получить эффект анимации. Например, это выглядит так, как будто какое-либо слово, или рисунок, периодически тонет в цветном шуме, и затем проявляется вновь. Исходный текст сценария приведен ниже.

```
<html>
<head>
<title>Animation with JavaScritp</title>
<script language="JavaScript">
i=1;
```

```
bForward=true;
function showNextImage()
{if(bForward)
{i++;
if(i>5)
{bForward=false;}
}
else
{i--;
if(i<2)
{bForward=true;}
}
document.Img.src="noise0"+i+".gif";
setTimeout("showNextImage()",500);
}
</script>
</head>
<body bgcolor=white>
<img src="noise01.gif" name="Img">
<script language="JavaScript">
showNextImage();
</script>
</body>
</html>
```

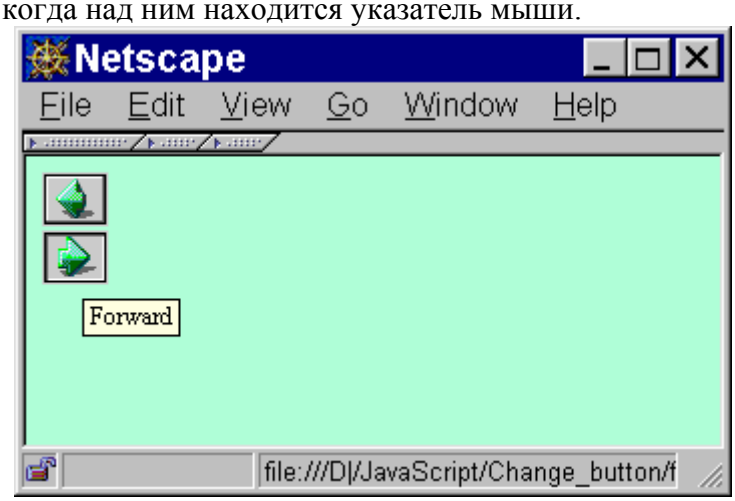
Последовательность кадров выглядит следующим образом:



Кадры выводятся в прямой, а затем в обратной последовательности.

Изменение внешнего вида графических ссылок

Этот прием может быть использован, например, для изменения графического элемента в том случае, когда над ним находится указатель мыши.



Если расположить указатель мыши над одной из этих кнопок, кнопка изменит свой внешний вид и появится всплывающая подсказка.

```
<html>
<body bgcolor="#B0FFD8">
<font face=”Arial, Helvetica" size=1>
<p>
<a href="back.htm"
onMouseOver="document.btn1.src='back_down.gif'"
```

```
onMouseOut="document.btn1.src='back_up.gif'">
<img src="back_up.gif" name="btn1" border=0 alt="Back"></a>
<br>
<a href="forward.htm"
onMouseOver="document.btn2.src='forward_down.gif'"
onMouseOut="document.btn2.src='forward_up.gif'">
<img src="forward_up.gif" name="btn2" border=0 alt="Forward"></a>
</font>
</body>
</html>
```

Лабораторная работа №2. Конфигурирование и администрирование web-сервера (на примере web-сервера Apache)

Задание:

Установить и настроить web-сервер. Проверить правильность настройки.

Создать два виртуальных сервера.

Расположить корневые каталоги документов серверов соответственно в <disk:>\infocom\virthost2\www и <disk:>\infocom\virthost3\www.

Файлы регистрации доступа и ошибок расположить в <disk:>\infocom\access.log и <disk:>\infocom\error.log. Файлы с описанием групп и пользователей расположить в <disk:>\infocom\security\groups и <disk:>\infocom\security\users.

В корневом каталоге документов одного из виртуальных серверов создать несколько каталогов и файлов. Определить различные права доступа к различным каталогам и файлам:

- доступ разрешен всем;
- доступ разрешен отдельным пользователям;
- доступ разрешен одной группе пользователей;
- доступ разрешен всем зарегистрированным пользователям;
- доступ запрещен всем.

Перенести определение прав доступа к одному из каталогов и одному из файлов в файл htaccess, расположенный непосредственно в каталоге, для которого определяются права доступа.

В корневом каталоге документов другого виртуального сервера организовать расширенную индексацию.

Содержание отчета:

- титульный лист;
- задание;
- дерево созданных каталогов;
- секция №3 файла конфигурации httpd.conf;
- файлы htaccess;
- выводы по работе.

Методические указания:

Web-сервер Apache

Apache – один из широко используемых в Internet web-серверов. В настоящее время программное обеспечение Apache установлено примерно на половине Web-узлов всего мира.

Основной файл конфигурации httpd.conf, используемый для управления web-узлом. В нем определяются базовые операции, указывается, как сервер должен работать с локальными ресурсами, отвечая на запрос, указывается, с какими файлами пользователи могут выполнять определенные операции. Через него осуществляется управление работой Apache. Настройка конфигурационного файла web-сервера – самый ответственный шаг при его установке. Сервер перечитывает конфигурационный файл при запуске. Если сервер работает, то при изменении файла конфигурации его следует перезапустить.

В файле httpd.conf директивы группируются в три основных раздела:

Section 1. Global Environment – директивы, которые управляют работой Apache в целом, влияют на общее функционирование Apache;

Section 2. Main Server Configuration – директивы, которые определяют параметры основного сервера, отвечающего на запросы, все эти директивы могут быть переопределены для виртуальных серверов;

Section 3. Virtual Hosts – установки для виртуальных серверов.

Виртуальные серверы (хосты)

Виртуальные серверы (хосты) – несколько web-серверов с различными IP-адресами, использующих один экземпляр программы Apache.

Сферы применения виртуальных серверов:

- создание отдельных web-серверов со своими адресами для различных компаний, организаций и индивидуальных пользователей;
- организация виртуальных серверов для отделов фирм, каждый отдел будет иметь собственное доменное имя и свой web-сервер;
- организация общедоступного и внутреннего web-серверов в виде двух виртуальных серверов;
- использование виртуального сервера для проверки или разработки web-сервера.

Директива VirtualHost

```
<VirtualHost ip_address_of_host>
…
</VirtualHost>
```

Пример: создать виртуальный сервер virthost2 с ip-адресом 127.0.0.2.

Структура каталогов выглядит следующим образом:

<disk:>\infocom\apache2 – каталог сервера Apache

<disk:> \infocom\virthost2\www – каталог для документов виртуального сервера

<disk:> \infocom\access.log – файл регистрации доступа

<disk:> \infocom\error.log – файл регистрации ошибок

Section 3 в файле httpd.conf будет иметь следующий вид:

```
<VirtualHost 127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot “<disk:>/infocom/virthost2/www”
CustomLog “<disk:>/infocom/access.log” common
ErrorLog “<disk:>/infocom/error.log”
</VirtualHost>
```

Для каждого виртуального сервера используется своя директива DocumentRoot, так как именно по этой причине и создаются виртуальные серверы. Файлы регистрации доступа и ошибок могут быть одними и теми же.

Для организации доступа к каталогам и файлам используются следующие директивы.

Директива AuthType

```
AuthType type
```

Задает тип контроля полномочий. Возможное значение – Basic.

Директива AuthName

```
AuthName realm
```

Задает область (realm), в которой действительны имена и пароли пользователей. Каждая пара имя/пароль действует в определенной области. Возможное значение – Test.

Директива AuthGroupFile

```
AuthGroupFile file_name
```

Задает имя файла, в котором содержится информация об именах групп и именах пользователей, входящих в эти группы. Этот файл текстовый и имеет следующий формат:

```
name_group: name_user name_user …
name_group: name_user name_user …
…
```

Директива AuthUserFile

```
AuthUserFile file_name
```

Задает имя файла, в котором содержится информация об именах пользователейи их паролях. Пароли хранятся в зашифрованном виде. Этот файл текстовый и имеет следующий формат:

name_user:password
name_user: password
…

Пример: имена групп, имена и пароли пользователей хранятся соответственно в файлах <disk:>\home\security\groups и <disk:>\home\security\users. Все имена и расположение каталога security выбрано произвольно. Единственное соображение безопасности заключается в том, что этот каталог лучше хранить выше каталога, указанного в директиве DocumentRoot.

Файл groups

adm: admin
group1: anna alex
group2: user1 user2 user3

Файл groups может быть создан с помощью простейшего текстового редактора

Файл users

admin:$apr1$yd5.....$Bkv.cPIQk/L7EOF2d2DNO.
anna:$apr1$Ze5.....$k/uC6j.ysELGdCK66BD6v0
alex:$apr1$Gf5.....$TZUpPFixB0h4pwvpfRkxl0
user1:$apr1$Sj5.....$HBg8Io5hbm/9quebHF3O01
user2:$apr1$Og5.....$WKYIDjTsbk.J.PDIhXu5x1
user3:$apr1$Xi5.....$AfQzJlIxyjiFh6KX23cE30

Для создания и изменения файла users используется Apache-утилита <disk:>\program files\apache group\apache\bin\htpasswd.exe. Для получения справочной информации ее можно запустить с ключом -?. Для создания файла используется ключ –c. Формат команды …\htpasswd.exe [-switch] name_user.

Для добавления второго и последующих пользователей или при измененииии пароля существующего пользователя ключ –c можно не указывать.

После создания файлов groups и users в блочную директиву <VirtualHost> можно добавить следующие строки:

<VirtualHost 127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot “<disk:>/infocom/virthost2/www”
CustomLog “<disk:>/infocom/access.log” common
ErrorLog “<disk:>/infocom/error.log”
<Directory “<disk:>/infocom/virthost2/www”>
AuthType Basic
AuthName Test
AuthGroupFile “<disk:>/infocom/security/groups”
AuthUserFile “<disk:>/infocom/security/users”
</Directory>
</VirtualHost>

После создания этих файлов можно установить права доступа к каталогам и файлам для всех пользователей, для групп пользователей, для отдельных пользователей или запретить доступ всем пользователям.

Директива Require

- Require [user name_user name_user …] # доступ разрешен всем перечисленным пользователям
- Require [group name_group name_ group …] # доступ разрешен всем пользователям из групп

– Require [valid-user] # доступ разрешен всем пользователям перечисленным в файле users

Пример:

```
<VirtualHost 127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot “<disk:>/infocom/virthost2/www”
CustomLog “<disk:>/infocom/access.log” common
ErrorLog “<disk:>/infocom/error.log”
<Directory “<disk:>/infocom/virthost2/www”>
AuthType Basic
AuthName Test
AuthGroupFile “<disk:>/infocom/security/groups”
AuthUserFile “<disk:>/infocom/security/users”
Require valid-user
#Require user anna user1
#Require group adm
</Directory>
</VirtualHost>
```

Если последовательно убирать комментарии, то будет разрешен доступ или всем зарегистрированным пользователям, или пользователям anna и user1, или пользователям, принадлежащим группе adm

Директива Require разрешает доступ пользователям на персональной основе. Для разрешения/запрещения доступа всем пользователям или из конкретных ip-адресов, хостов, доменов используются директивы Allow from и Deny from. Эти директивы могут иметь следующие параметры: all – доступ разрешен/запрещен всем, имя хоста – доступ разрешен/запрещен только данному хосту, имя домена – доступ разрешен/запрещен только хостам из данного домена.

Порядок, в котором применяются директивы Allow from и Deny from, определяется не порядком их следования в конфигурационном файле, а директивой Order.

По умолчанию сначала выполняется директива Deny from, а затем Allow from. Если клиент упомянут в директиве Deny from, ему запрещается доступ при условии, что он не упомянут в Allow from. Если клиент не назван ни в одной из этих директив, ему доступ разрешается.

Порядок можно изменить, использовав директиву Order allow,deny. Это значит, что доступ клиенту, который указан в директиве Allow from разрешен, если только он не упомянут в Deny from. Если ни в одной из этих директив этот клиент не указан, доступ ему запрещается.

Директива Files

```
<Files name_file>
…
</Files>
```

Эта директива используется для организации доступа к файлу. Установки директивы Files отменяют установки директи вы <Directory>.

Пример: каталоге www содержит файл secret.html

```
<VirtualHost 127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot “<disk:>/infocom/virthost2/www”
CustomLog “<disk:>/infocom/access.log” common
ErrorLog “<disk:>/infocom/error.log”
<Directory “<disk:>/infocom/virthost2/www”>
```

AuthType Basic
Auth Test
AuthGroupFile “<disk:>/infocom/security/groups”
AuthUserFile “<disk:>/infocom/security/users”
Require valid-user
#Require user anna user1
<Files secret.html>
Require user anna user1
#Require valid-user
</Files>
</Directory>
</VirtualHost>

При изменении прав доступа в файле httpd.conf каждый раз требуется перезапускать Apache. Для того, чтобы избежать перезапуска, можно поместить директивы, определяющие права доступа к каталогам и файлам непосредственно в каталог. Для определения имени файла с директивами используется директива AccessFileName.

Директива AccessFileName

AccessFileName file_name

Пример: AccessFileName htaccess

Если будут изменяться директивы в файле htaccess, то это не потребует перезапуска Apache.

Пример: использование файла htaccess

Файл httpd.conf

<VirtualHost 127.0.0.2>
ServerName virthost2
ServerAdmin admin@virthost2
DocumentRoot “<disk:>/infocom/virthost2/www”
CustomLog “<disk:>/infocom/access.log” common
ErrorLog “<disk:>/infocom/error.log”
<Directory “<disk:>/infocom/virthost2/www”>
AuthType Basic
Auth Test
AuthGroupFile “<disk:>/infocom/security/groups”
AuthUserFile “<disk:>/infocom/security/users”
Require valid-user
</Directory>
</VirtualHost>

Файл htaccess

<Files secret.html>
Require user anna user1
</Files>

В файл htaccess также можно помещать и директивы из блочной директивы <Directory>, но только без ее указания.

В том случае, если каталог не содержит файла index.html, Apache создает файл Index of/, который в общем случае выглядит не очень хорошо. Для изменения внешнего вида можно использовать улучшенную индексацию.

Директива IndexOptions

IndexOptions FancyIndexing

Данная директива позволяет применять расширенную индексацию.

Директива IndexIgnore

IndexIgnore file1 file2 …

Директива позволяет исключить файлы из списка. Несколько директив IndexIgnore дополняют друг друга.

Пример: исключение из списка всех файлов с расширением jpg и родительского каталога (например, из соображений безопасности).

```
<Directory …>
FancyIndexing on
IndexIgnore *.jpg ..
</Directory>
```

Директива AddIcon

Первый аргумент этой директивы – имя файла со значком, второй – тип файла, к которым значок следует добавить. Если в качестве второго аргумента использовать значение ^^DIRECTORY^^, то новый значок получит подкаталог.

Пример:

```
Alias “/icons” “<disk:>/program files/apache group/apache/icons”
AddIcon icons/first.gif *.html
AddIcon icons/second.gif ^^DIRECTORY^^
```

Директива AddDescription

Позволяет добавить описание для определенных файлов.

Пример:

```
AddDescription “Это файлы с HTML-документами” *.html
```

Директивы HeaderName и ReadmeName

Позволяют формировать верхний и нижний колонтитул индекса. Эти директивы имеют только один аргумент – имя файла. Для того, чтобы имена файлов, указанных в директивах HeaderName и ReadmeName не отображались в перечне, для них используется директива IndexIgnore.

Лабораторная работа №3. Создание тестовой системы и счетчика посещений страницы средствами CGI и PHP

Задание:

Во всех вариантах задания необходимо разработать CGI-модуль и PHP-скрипт для решения одной задачи.

Во всех вариантах заданий необходимо разработать HTML-документ, содержащий форму и CGI-модуль для обработки информации, введенной в форму, а также HTML-документ с текстом на PHP для обработки информации, введенной в форму.

Информация вводится в форму клиентом и отсылается серверу. По результатам обработки переданной информации динамически генерируется HTML-документ, который сервер возвращает клиенту. В исходном HTML-документе содержится несколько вопросов с несколькими вариантами ответа на каждый вопрос. В динамически генерируемом возвращаемом HTML-документе содержится результат тестирования (например, оценка). Тест состоит из пяти вопросов с тремя вариантами ответов на каждый вопрос.

Сравнить скорость работы CGI-модуля и PHP-скрипта.

Вариант 1

Проверка знания правил дорожного движения и текстовый счетчик посещения страницы.

Вариант 2

Проверка знания таблицы умножения (с генерацией сомножителей датчиком случайных чисел) и текстовый счетчик посещения страницы.

Вариант 3

Любой психологический тест и текстовый счетчик посещения страницы.

Вариант 4

Проверка знания языка HTML и текстовый счетчик посещения страницы.

Вариант 5

Экзамен по дисциплине и текстовый счетчик посещения страницы.

Содержание отчета:

- титульный лист;
- задание;
- краткое описание использованных средств и методов;
- исходные тексты;
- выводы по работе.

Методические указания:

CGI (Common Gateway Interface) – общий шлюзовой интерфейс

Один из способов формирования динамических HTML-документов заключается в использовании приложений CGI.

CGI – это интерфейс для запуска внешних программ под управлением web-сервера.

Приложение CGI – программа, использующая CGI-интерфейс, получает информацию от удаленного пользователя, обрабатывает ее, и возвращает результат (динамически сформированный HTML-документ, гиперссылка на существующий HTML-документ, графическое изображение и т.д.) Так как CGI-приложение – это программа, она должна быть оттранслирована для той операционной системы, под управлением которой работает web-сервер.

На стороне клиента размещается форма ввода, содержащая некоторые поля для ввода данных и кнопку для отсылки данных. После заполнения полей и нажатия кнопки данные в

запросе клиента пересылаются на сторону сервера, где web-сервер передает присланные данные CGI-приложению.

После обработки полученных данных CGI-приложение создает документ и передает его web-сервер, который в ответе сервера возвращает документ на сторону клиента.

Передача информации от клиента к серверу и передача сформированного документа от сервера к клиенту изображена на рис. 1.

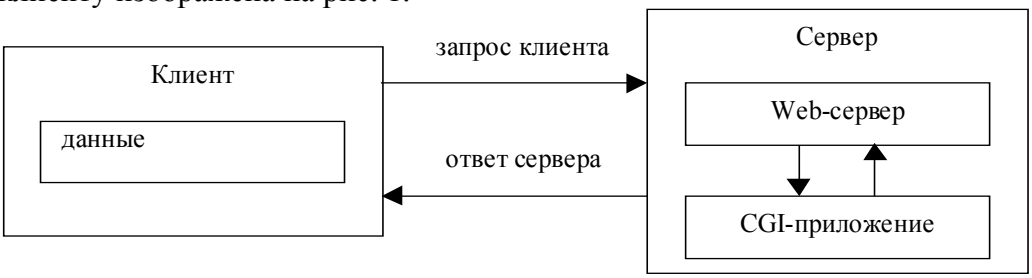


Рис. 1. Передача информации от клиента к серверу и передача сформированного документа от сервера к клиенту.

Для создания формы на стороне клиента для занесения данных используется тег <form>.

```
<form action=url method=get|post>
…
</form>
```

Атрибут action=url определяет url GCI-приложения, предназначенного для обработки присланных данных. По умолчанию используется текущий url.

Атрибут method=get|post указывает метод передачи данных серверу. По умолчанию используется метод get.

Метод get

Метод get предполагает передачу данных GCI-приложению через переменные среды (environment variables), устанавливаемые на стороне сервера.

В зависимости от web-сервера и операционной системы могут использоваться разные переменные среды.

Для передачи данных, присланных методом get, используется переменная QUERY_STRING. Значением переменной QUERY_STRING будет строка, содержащая данные в формате name1=value1&name2=value2& … &nameN=valueN, где name – это имя поля формы, value – значение поля формы.

Метод post

При использовании метода post GCI-приложение получает присланные данные через стандартный поток ввода.

Количество байт переданных данных можно получить через переменную среды CONTENT_LENGTH.

Формирование HTML-документа

Вне зависимости от метода передачи данных, результат своей работы GCI-приложение должно направить в стандартный поток вывода.

Чаще всего GCI-приложение используется для создания HTML-документов на основе данных, полученных от клиента. В этом случае, первой строкой должен быть заголовок HTTP Content-type: text/html, за которой необходимо вывести пустую строку, отделяющую заголовки HTTP от данных HTML-документа.

Web-сервер возвращает результат, сформированный GCI-приложением, клиенту, возможно дополняя его заголовками HTTP.

GCI-приложение может сформировать полный ответ (со всеми заголовками HTTP). В этом случае web-сервер ничего не изменяет в результате работы GCI-приложения, только пересылает его клиенту как есть.

Пример: на стороне клиента в поля формы заносятся имя и возраст, в зависимости от возраста возвращаются разные приветствия (рассматриваются два варианта: для методов get и post).

Вариант 1

HTML-документ, содержащий форму:

```
<html>
<form action=http://localhost/cgi/hello.exe method=get>
<p>ИМЯ<input type=text name=name>
<p>ВОЗРАСТ<input type=text name=age>
<p><input type=submit>
</form>
```

CGI-приложение (файл hello.cpp)

```
#include <iostream.h>
void main()
{
int age;
char *name;
char *query_string=getenv("QUERY_STRING");

//query_string=”name=Maria&age=18”
//из строки извлекаются подстроки “Maria” и “18”
//и присваиваются переменным name и age соответственно

cout<<”Content-type: text/html\n\n”;
cout<<”<html>”;
if(age<=16) cout<<”Привет, ”;
if(age>16) cout<<”Здравствуйте, ”;
cout<<name<<”</html>”;
}
```

Вариант 2

HTML-документ, содержащий форму:

```
<html>
<form action=http://localhost/cgi/hello.exe method=post>
<p>ИМЯ<input type=text name=name>
<p>ВОЗРАСТ<input type=text name=age>
<p><input type=submit>
</form>
```

CGI-приложение (файл hello.cpp)

```
#include <iostream.h>
void main()
{
int age;
char *name;
int length=atoi(getenv("CONTENT_LENGTH"));
char * string=new char[length];
scanf("%s",string);

//string=”name=Maria&age=18”
```

```
//из строки извлекаются подстроки “Maria” и “18”
//и присваиваются переменным name и age соответственно

delete string;
cout<<”Content-type: text/html\n\n”;
cout<<”<html>”;
if(age<=16) cout<<”Привет, ”;
if(age>16) cout<<”Здравствуйте, ”;
cout<<name<<”</html>”;
}
```

Лабораторная работа №4. Создание электронного магазина (средствами PHP и MySQL)

Задание:

Написать скрипт, позволяющий организовать электронный магазин.

Список товаров хранится в базе данных на стороне сервера. Покупатель должен иметь возможность просмотреть все имеющиеся в наличии товары и сделать заказ. Покупатель должен иметь возможность сделать запрос, например, указав интервал цен, который его устраивает или какие-либо другие данные.

До тех пор, пока покупатель выбирает отдельные товары, его заказ хранится на стороне клиента в виде cookie.

После того как покупатель сформировал заказ, заказ отсылается на сторону сервера, где покупка товара учитывается в базе данных.

Вариант 1

В базе данных содержится информация о книгах: автор, название, изображение обложки, издательство, год выпуска, цена.

Вариант 2

В базе данных содержится информация об автомобилях: модель, изображение автомобиля, год выпуска, тип кузова, мощность двигателя, цвет, цена.

Вариант 3

В базе данных содержится информация о туристических поездках: страна, город, изображение городской достопримечательности, количество дней, дата поездки, класс отеля, цена.

Вариант 4

В базе данных содержится информация о журналах: название, изображение обложки, год выпуска, номер, издательство, число страниц, цена.

Вариант 5

В базе данных содержится информация о местах в отеле: название отеля, класс номера, изображение номера, количество мест в номере, цена.

Содержание отчета:

- титульный лист;
- задание;
- краткое описание использованных средств и методов;
- описание базы данных;
- исходные тексты;
- выводы по работе.

Методические указания: