МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«09» 04 2022г.

Разработка корпоративных сайтов

Методические указания по выполнению лабораторных работ
для направления 09.03.02 Информационные технологии в бизнесе

Курск 2022

УДК 681.3(075)
Составитель: Л.А. Лисицин

Рецензент
Кандидат технических наук, доцент *Халин Ю.А.*

Разработка корпоративных сайтов [Текст]: методические указания по выполнению лабораторных работ для направления 09.03.02 Информационные технологии в бизнесе / Юго-Зап. гос. ун-т; сост.: Л.А. Лисицин. Курск, 2021. 70 с.: ил. 12. табл. 1. Библиогр. с. 70.

Содержат сведения по технологиям разработки сайтов. Материал ориентирован на практическую работу студентов в компьютерной среде.

Отражен порядок выполнения лабораторных работ и правила оформления отчетов.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальности «Информационные технологии в бизнесе».

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл.печ. л. __. Уч.-изд. л. __. Тираж 50 экз. Заказ . Бесплатно.
Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

Оглавление

Лабораторная работа 1. HTML. Создание статической web-страницы

Цель работы – научиться создавать HTML-документы, представляющие собой статические web-страницы.

Статическая web-страница – это страница, содержание и внешний вид которой не меняются во время просмотра пользователем.

Для создания и редактирования HTML-документов будем использовать текстовый редактор ***Блокнот***.

Откройте ***Блокнот*** и наберите следующий текст:

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

        <body>
                <p>Ура!!! Это моя первая страница</p>
        </body>

 </html>
```

Сохраните набранный текст в файле с расширением имени ***html***. Для этого выполните команду ***Файл – Сохранить как*** (см. рис. 1). В поле **Имя файла** вместо предлагаемого программой имени **.txt* введите *MyFirstPage.html*. Выберите в поле **Кодировка** кодировку *UTF-8*.

Откройте созданный файл в браузере и убедитесь, что ваша страничка работает.

Как и любой язык, HTML поставляется с набором правил. Эти правила относительно простые и сводятся к определению границ, чтобы знать, где что-то начинается и где заканчивается.

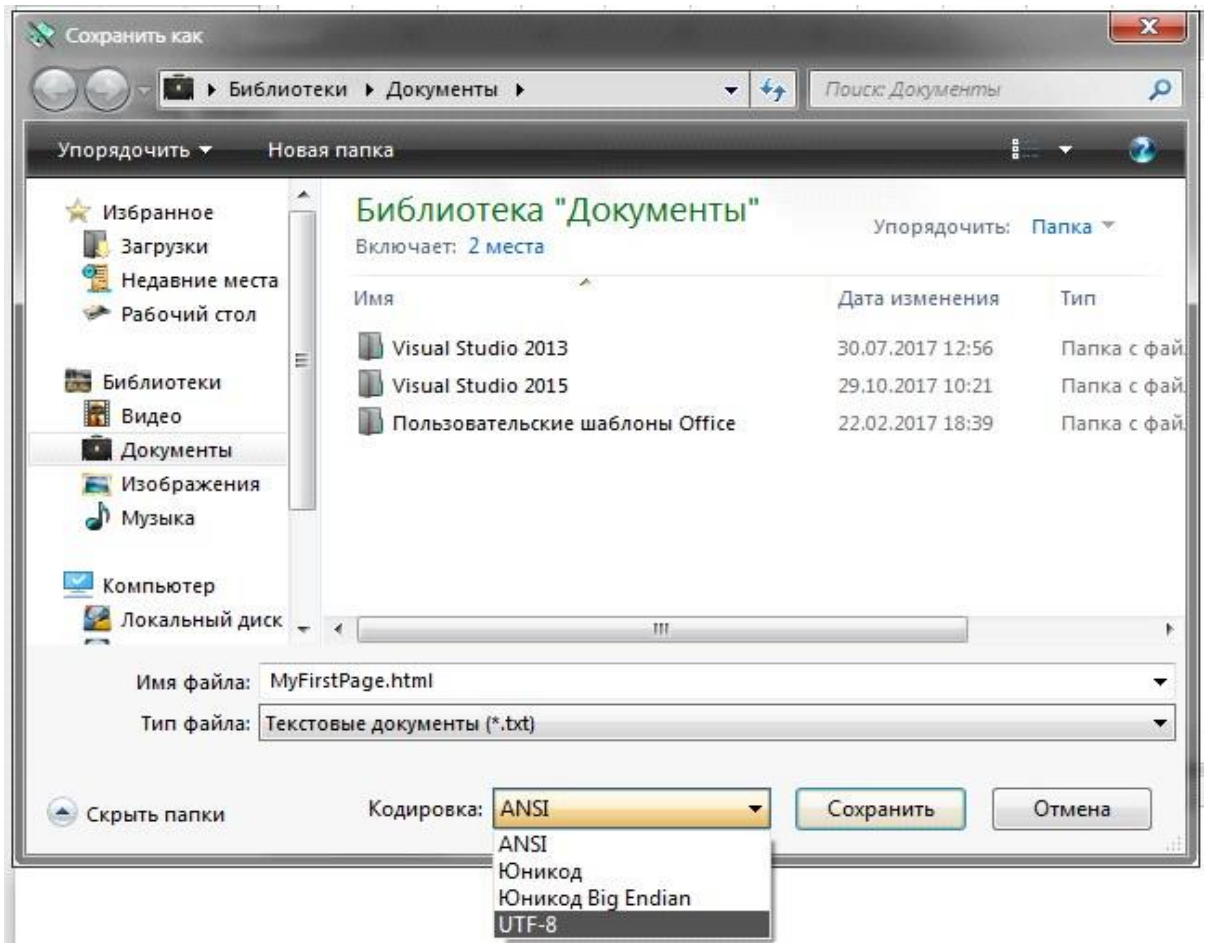


Рис. 1. Сохранение HTML-файла Ниже

приведён пример абзаца в HTML:

```
<p>Если Тетрис и научил меня чему-то, так это тому, что ошибки
накапливаются, а достижения исчезают.
</p>
```

Пример применения тегов <P> и <FONT>

Приведенный ниже HTML-код обеспечивает вывод 4-х абзацев текста:

```
<html>
<head>
  <title>
   Тэги P и FONT
  </title>
</head>
<body bgcolor=blue text=yellow>
<P>
 1-ый абзац текста
```

```
</P>
<P align=center>
  2-ой абзац текста
</P>
<P align=right>
  3-ий абзац текста
</P>
<P>
   <FONT face=courier color=red size=18pt>
   4-ый абзац текста
   </FONT>
</P>
</body>
</html>
```

Параметр ***align*** задает режим выравнивания текста.
Параметр ***face*** задает имя шрифта.
Параметр ***color*** задает цвет шрифта.
Параметр ***size*** задает размер шрифта.

Задание

Создать HTML-документ, в разметке документа использовать:

- тег для определения кодировки кириллицы <meta>;
- тег комментария <!-- -->;
- теги форматирования текста: <p>,
, , , <hr>, <h1> □ <h6>, <b>, <i>, <u>, <sub>, <sup>, <pre>, <tt>;
- продемонстрировать отличия тегов <p> и
, и ;
- тег для разметки изображения <img>;
- тег для разметки гиперссылок <a>, разместить ссылки на другой документ, в пределах размечаемого документа, на email;
- с помощью параметров тега <body> изменить цвет фона документа, цвет текста, цвета непосещенных и посещенных ссылок документа, используя цвета из web-безопасной (гарантированной) палитры.

Лабораторная работа 2. Макет страницы. Требования к иллюстрациям в Internet.

\Javascript. Создание динамических web-страниц\

Цель работы – научиться использовать различные способы доступа к свойствам и методам объектов для внесения изменений в HTML-документ.

В динамических web-страницах изменения в HTML-документе производятся через свойства, методы и события объектов, входящих в состав объектной модели документа. Рассмотрим сценарий (пример 1), в котором используются типичные способы доступа к свойствам и методам объектов.

Пример 1

```
<HTML ID='DOCUM'>
<CENTER><H1 ID='zag'> ВАСЬКА</h1>
Вариант, совместимый с Mozilla<P>
<FORM NAME='f1'>
<INPUT TYPE=BUTTON name="bot" onclick=bm()
value='Увеличить'>
</FORM >
<P><IMG SRC="VaskaM.jpg" ID='Vas' onclick=bm()>
</center>
<!--В скрипте закомментирован вывод окна со списком объектов ->
<SCRIPT>
/*
 var msg="
 for(i=0; i<document.all.length;i++)
 msg+=i + ' ' + document.all[i].tagName +' ID=' + document.all[i].id+'\n'
alert(msg)
*/
flag=true function
bm()
{ if(flag)
{ document.images[0].src='VaskaB.jpg' flag=false;
document.forms[0].bot.value="Уменьшить"
document.forms[0].bot.style.background='red'
  document.all.bot.style.color='black';
}
else
{ document.getElementById("Vas").src='VaskaM.jpg';
```

```
 //обращение к кнопке по индексу   document.forms[0].bot.value="Увеличить"
document.forms[0].bot.style.background='green'
document.all.bot.style.color='white'; flag= true
}
}
</script>
</HTML>
```

В примере 1 сценарий используется для замены фотографии и изменения цвета кнопки и надписи на ней. Друг друга меняют маленькая и большая фотографии кота Васьки.

Замена фотографии и изменение вида кнопки в примере происходят при щелчке мышкой по кнопке или по самой фотографии. В тегах *<INPUT>* и *<IMG>* помещён атрибут *onclick=bm()*, при помощи которого в ответ на щелчок мышкой вызывается функция *bm()*. Такой атрибут называют обработчиком событий. Имя файла с фотографией является свойством *SRC* объекта *IMG*. Замена фотографии делается в сценарии двумя равнозначными способами:

```
document.images[0].src='VaskaB.jpg'
document.getElementById("Vas").src='VaskaM.jpg'
```

В первом варианте доступ к фотографии происходит через коллекцию **images**, в которой все расположенные на странице фотографии (в примере одна фотография) пронумерованы. Нумерация начинается с нуля. Во втором варианте используется метод ***getElementById()*** объекта ***document***. Параметром в методе ***getElementById()*** служит **ID** (идентификатор) объекта **IMG.** Весь оператор читается так: «свойству src объекта, имеющего *ID=Vas* и входящего в состав объекта *document*, присвоить значение *VaskaM.jpg*. Сам объект **document** является дочерним по отношению к объекту **window**. Поэтому полное обращение к свойству src следовало бы писать так:

```
window.document.images[0].src='VaskaB.jpg'
window.document.getElementById("Vas").src='VaskaM.jpg'
```

Обычно, за исключением тех случаев, когда нужно воспользоваться свойствами или методами самого объекта ***window***, слово ***window*** опускают.

Более сложная иерархия объектов наблюдается при обращении к свойствам кнопки:

```
document.forms[0].bot.value="Уменьшить"
document.forms.f1.bot.style.background='red'
```

Первый оператор служит для изменения надписи на кнопке. Доступ к кнопке происходит по её имени (name='bot'). Доступ к форме осуществляется через коллекцию *forms* и номер формы в коллекции.

С помощью второго оператора меняется цвет кнопки. Доступ к форме осуществляется через коллекцию **forms** и имя формы ***f1***. Своеобразие доступа к свойствам, задаваемым стилем, состоит в том, что **style** выступает формально как дочерний объект. В рассматриваемом примере родительским по отношению к *style* является объект с именем **bot**, т. е. кнопка.

Для обращения к объектам удобна коллекция ***all*** объекта document. В коллекцию **all** входят все объекты HTML-документа, поэтому выбирать объект нужно по его **ID** или по его имени. Коллекция **all** использована для изменения цвета надписи на кнопке:

```
document.all.bot.style.color='black'
```

Все описанные способы доступа к объектам HTML-документа базируются на объектной модели документа (DOM). Наиболее трудной задачей, стоящей перед разработчиком сценариев, является разработка сайтов, которые будут одинаково выполняться на всех браузерах. Рассматриваемый пример HTML-документа подобран так, что он (почти) одинаково выполняется на браузерах Internet Explorer 6.0 и Mozilla Firefox 2.0, несмотря на то, что объектные модели в этих браузерах отличаются друг от друга. Для того чтобы убедиться в этом, уберите символы комментариев /* и */ из первой и пятой строчек скрипта и откройте страницу в обоих браузерах. Появится окно со списком всех объектов, созданных браузером для открытого документа. В созданной браузером Internet Explorer модели 12 объектов, а у Mozilla Firefox – 10.

Рассмотрим подробнее фрагмент скрипта.

```
var msg=''
for(i=0; i<document.all.length;i++)
msg+=i + ' ' + document.all[i].tagName +' ID=' +
document.all[i].id+'\n' alert(msg)
```

Коллекция **all** всех объектов HTML-документа имеет свойство **length**, равное количеству всех объектов модели документа. Любой объект имеет свойство **tagName**, значение которого совпадает с именем объекта. В приведённом фрагменте скрипта в цикле формируется строковая переменная ***msg***, состоящая из номеров, имён и ***ID*** объектов. Сочетание символов **\n** служит для перевода строки.

Метод ***getElementById()*** – входит в стандарт W3C и поддерживается всеми браузерами. Коллекция ***all*** разработана фирмой Microsoft для браузера *Internet Explorer*. В настоящее время эта коллекция поддерживается почти

всеми брауэерами. Как ни странно, в новом браузере *Edge*, встроенном в ОС Windows 10, коллекция **all** не поддерживается.

Задание 1

Создайте HTML-документ, который в окне браузера отображается в виде следующих трёх строк:

- ДОСТУП К СВОЙСТВАМ И МЕТОДАМ.
- Коллекция all.
- Метод getElementById().

Первую строку поместите в контейнер ***<H2>...</h2>***, вторую – в контейнер ***<P> …</p>***, третью – в контейнер ***<DIV> … ***. Напишите скрипт для изменения цветов фона и букв надписей при щелчке по этим строкам. При щелчке по первой строке цвет букв должен меняться с чёрного на белый или с белого на чёрный, а фон – с жёлтого на синий или с синего на жёлтый. Также должны меняться цвета третьей строки.

При щелчках по второй строке цвет букв на ней должен меняться с красного на белый и наоборот, а цвет фона – с белого на зелёный и наоборот.

Для изменения первой и третьей строк примените метод ***getElementById()***, а для второй строки – коллекцию **all**.

Размещение кода на языке JavaScript в HTML-документе

Сценарий в HTML-документе можно размещать тремя способами:

1) в открывающем теге в качестве значения атрибута – событие (см. пример 1);

2) в контейнере, ограниченном тегами *<SCRIPT> ...</script>;* 3) в отдельном файле.

Контейнеры со сценариями могут размещаться в любом месте HTML-документа. Количество сценариев в одном HTML-документе не ограничено.

Выполнение сценариев происходит при загрузке HTML-документа и при наступлении определенных событий. Во время загрузки HTML-документа браузер последовательно просматривает встречающиеся сценарии и пытается их выполнить. Если встретился вызов функции, то браузер ищет её описание в текущем и во всех ранее загруженных сценариях. Поэтому описание функции должно размещаться либо в одном сценарии с вызовом, либо в сценариях, расположенных выше вызова функции.

Сценарий можно хранить в отдельном файле. Для вставки сценария в HTML-документ служит атрибут **SRC** тега **<SCRIPT>.** Сценарий, размещённый в отдельном файле, можно использовать на многих страницах сайта. В примере 2 рассматривается одна из страниц сайта сети магазинов.

Пример 2 html><!-- СКРИПТ загружается из файла primJs.js-->

```
<HEAD>
<TITLE>Сеть</title>
</head>
<body>
<SCRIPT language="JavaScript" src="PrimJs.js">
</script>
<H2 align=center style="color:green">Магазин "ПОДАРКИ"</h2>
Адрес: Лесная ул., д.2<P>
Транспорт: трамваи 7, 23, автобусы 56, 93
</body>
</html>
```

Содержимое файла сценария primJs.js:

```
// Файл primJs.js
a="background-color:#00ffff; color:#ff00ff;" a+="font-
size:24pt; font-family:'Times New Roman'" naim='Сеть
магазинов "ВСЁ ДЛЯ ДОМА"' var da=new Date()
d=da.getDate()+"."+(da.getMonth()+1)+"."+da.getFullYear() document.write('<P
align=center style= "'+a+'">'+ naim+'</p><P>Сегодня '+d+'</p>')
```

Общие для страниц всех магазинов сети заголовок с названием сети и сегодняшняя дата формируются скриптом, размещённым в отдельном файле *primJs.js* . Метод *write*(*HTML-код*) служит для вставки в страницу размеченного текста (HTML-кода). Этот метод применяется только при загрузке страницы, так как его применение после окончания загрузки приведёт к стиранию старого содержимого страницы.

Задание 2

Добавьте в пример 2 три страницы. Две страницы должны отображать информацию о магазинах *Посуда* и *Мебель*. Третья страница – главная в сайте сети магазинов *ВСЁ ДЛЯ ДОМА*. На ней должны быть ссылки на страницы магазинов, входящих в сеть. На страницах магазинов должен использоваться скрипт из файла **primJs.js.**

Лабораторная работа 3. Использование стиля CSS при оформлении сайта.

Цель работы – научиться использовать различные способы доступа к свойствам и методам объектов для внесения изменений в HTML-документ.

Каскадная таблица стилей (Cascade Style Spreadsheet, CSS) используется для создания правил о цвете, шрифте и макете наших страниц. CSS также определяет, когда эти правила должны использоваться на основе информации об устройстве, подключающемся к странице, или в ответ на действие пользователя. CSS может использоваться не только HTML, но и любым языком на основе XML.

Применение CSS дает разработчику следующие возможности :

– CSS позволяет значительно сократить размер кода и сделать его читабельным;

– CSS позволяет задавать такие параметры, которые нельзя задать только языком HTML (например, отменить подчеркивание у гиперссылок);

– CSS позволяет быстро изменять внешний вид большого числа страниц.

– ***Стиль*** в CSS представляет правило, которое указывает webбраузеру, как надо форматировать элемент. Форматирование может включать установку цвета фона элемента, установку цвета и типа шрифта и так далее.

– ***Определение стиля*** состоит из двух частей: ***селектор***, который указывает на элемент, и ***блок объявления стиля*** – набор команд, которые устанавливают правила форматирования. Например:

– `div{ background-color:red; width: 100px; height: 60px;}`

– В данном случае селектором является ***`div`***. Этот селектор указывает, что этот стиль будет применяться ко всем элементам div.

– После селектора в фигурных скобках идет *блок объявления стиля*. Между открывающей и закрывающей фигурными скобками определяются команды, указывающие, как форматировать элемент.

– Каждая команда состоит из ***свойства*** *и* ***значения***. Так, в следующем выражении:

– `background-color:red;`

– ***`Background-color`*** представляет название свойства, а ***`red`*** – значение свойства.

– Свойство определяет конкретный стиль. Свойств CSS существует множество. Например, *`background-color`* определяет цвет фона. После двоеточия идет значение для этого свойства. Например, выше указанная команда определяет для свойства *`background-color`* значение *`red`*. Иными словами, для фона элемента устанавливается цвет *red*, т. е. красный.

– После каждой команды ставится точка с запятой, которая отделяет данную команду от других.

Способы определения стилей

Первый способ заключается во встраивании стилей непосредственно в элемент с помощью атрибута ***style***:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Стили</title>
   </head>
   <body>
      <h2 style="color:blue;">Стили</h2>
      <div style="width: 100px; height: 100px; background-color:
red;"></div>
   </body>
</html>
```

Здесь определены два элемента – заголовок ***h2*** и блок ***div***. У заголовка определен синий цвет текста с помощью свойства `color`. У блока div определены свойства ширины (`width`), высоты (`height`), а также цвета фона (`background-color`).

Второй способ состоит в использовании элемента `style` в документе html. Этот элемент сообщает браузеру, что данные внутри являются кодом CSS, а не HTML:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>Стили</title>
      <style>
h2{
         color:blue;
      }
div{
         width: 100px;
height: 100px;
background-color: red;
      }
      </style>
   </head>
   <body>
      <h2>Стили</h2>
      <div></div>
   </body>
</html>
```

Результат будет абсолютно тем же, что и в предыдущем случае.

Часто элемент ***style*** определяется внутри элемента ***head***, однако может также использоваться в других частях HTML-документа.

Элемент ***style*** содержит наборы стилей. У каждого стиля указывается вначале ***селектор***, после чего в фигурных скобках идут все те же определения свойств CSS и их значения, что были использованы в предыдущем примере.

Второй способ делает код HTML «прозрачнее» за счет вынесения стилей в элемент `style`. Но также есть и третий способ, который заключается в вынесении стилей во внешний файл.

Создадим в одной папке с html странице текстовый файл, который переименуем в *styles.css* и определим в нем следующее содержимое:

```
h2{
    color:blue;
} div{     width: 100px;
height: 100px;
background-color: red;
}
```

Это – те же стили, что были внутри элемента style. Изменим код html-страницы:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Стили</title>
        <link rel="stylesheet" type="text/css" href="styles.css"/>
    </head>
    <body>
        <h2>Стили</h2>
        <div></div>
    </body>
</html>
```

Здесь уже нет элемента ***style***, но есть элемент ***link***, который подключает выше созданный файл ***styles.css***:

```
<link       rel="stylesheet"        type="text/css"
href="styles.css"/>
```

Таким образом, при определении стилей во внешнем файле структура страницы отделяется от ее оформления. При таком определении стили модифицировать легче, чем если бы они были определены внутри элементов или в элементе ***style***, и такой способ является предпочтительным в HTML5.

Использование стилей во внешних файлах позволяет уменьшить нагрузку на web-сервер с помощью механизма кэширования, поскольку web-браузер может кэшировать css-файл и при последующем обращении к web-странице извлекать нужный css-файл из кэша.

Также возможна ситуация, когда все эти подходы сочетаются, а для одного элемента одни свойства css определены внутри самого элемента, другие свойства css определены внутри элемента ***style***, а третьи находятся во внешнем подключенном файле.

Например:

```
<!DOCTYPE html>
<html>
   <head>
      <link rel="stylesheet" type="text/css" href="styles.css"/>
      <style>
div{
            width:200px;
         }
      </style>
   </head>
   <body>
      <div style="width:120px;"></div>
   </body>
</html>
```

А в файле *style.css* определен следующий стиль:

```
div{
width:50px;
height:50px;
   background-color:red;
}
```

В данном случае в трех местах для элемента ***div*** определено свойство ***width***, причем с разными значениями. Какое значение будет применяться к элементу в итоге? Здесь у нас действует следующая система приоритетов.

Если у элемента определены встроенные стили (inline-стили), то они имеют высший приоритет, то есть в примере итоговой шириной элемента ***div*** будет 120 пикселей.

Далее, в порядке приоритета идут стили, которые определены в элементе ***style.***

Наименее приоритетными стилями являются те, которые определены во внешнем файле.

Задание. Использование параметра float – большая помощь для адаптивного стиля. Блоки с содержимым, которые обычно помещаются рядом на большом экране, автоматически создают больше «рядов» с меньшим количеством элементов в каждом для размещения экранов с меньшей шириной. Создайте страницу со следующим кодом, а затем поэкспериментируйте с размером окна браузера, чтобы увидеть изменение размера в действии:

```
<style>
        .thumbnail { float:left;
            width:80px;
            height:80px;
            margin:5px;
        }
    </style>
    <div>
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
        <img class="thumbnail" src="yourPictureHere.jpg" >
    </div>
```

Содержимое до и после плавающего элемента будет пытаться разместиться вокруг него с обтеканием. Если мы не хотим, чтобы это произошло, мы можем добавить правило к стилю этого элемента, чтобы очистить плавающий эффект. Для этого надо добавить

```
clear: left; // для левой стороны
clear: right // для правой стороны
clear: both // для обеих сторон
```

в зависимости от того, с какими сторонами мы имеем дело.

Лабораторная работа 4. Разработка web-приложения на языке PHP. Создание базы данных

Цель работы – научиться создавать базы данных MySQL с помощью программы NetBeans.

В лабораторных работах 5–8 приводится пример разработки web-приложения ***Wish list***, после развертывания которого пользователи получают возможность размещения и совместного использования информации в списках желаний, например создания списков подарков к свадьбе, дню рождения или другим праздникам.

Приложение поддерживает регистрацию и авторизацию пользователей.

Любой зарегистрированный пользователь может просмотреть списки пожеланий других пользователей и имеет возможность редактирования собственных списков желаний.

В этом лабораторном практикуме для создания, выполнения и отладки проектов на PHP используются IDE NetBeans для PHP и локальный web-сервер XAMPP.

Загрузку IDE NetBeans для PHP можно осуществить с сайта https://netbeans.apache.org/download/index.html.

XAMPP можно загрузить с электронного адреса: https://www.apachefriends.org/download.html.

Создание базы данных MySQL с помощью программы NetBeans

IDE NetBeans поставляется с включенной поддержкой для MySQL. Для получения доступа к серверу баз данных MySQL в IDE NetBeans необходимо настроить свойства сервера MySQL.

1. Щелкните правой кнопкой мыши узел Databases («Базы данных») в окне Services («Службы») и выберите Register MySQL Server («Зарегистрировать MySQL»). Открывается диалоговое окно свойств сервера MySQL (рис. 3).

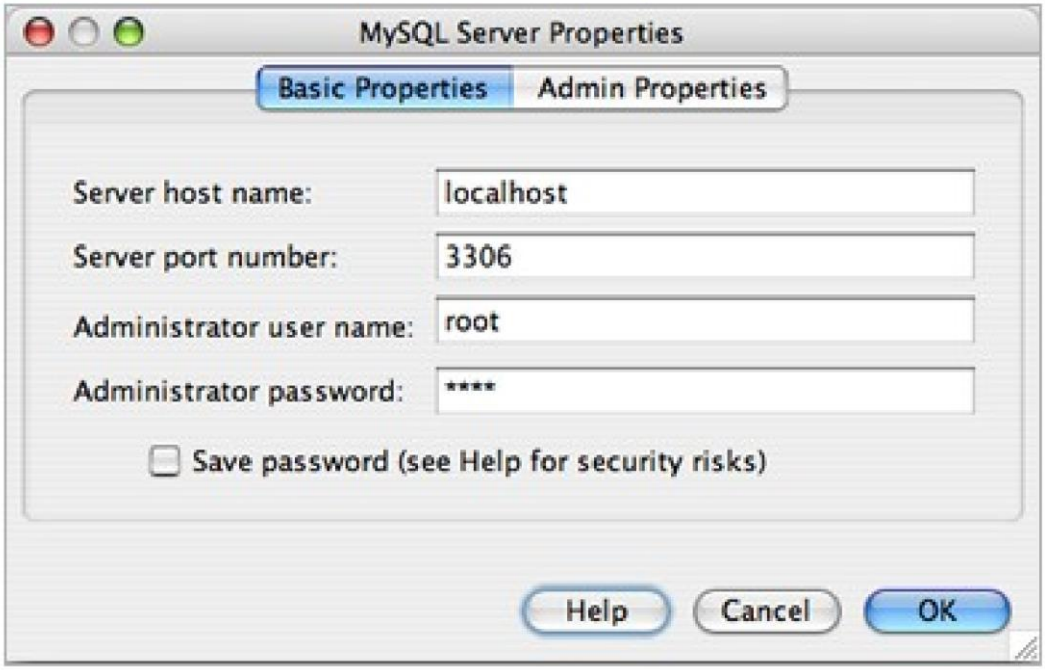


Рис. 3. Диалоговое окно свойств MySQL Server

2. Убедитесь, что имя узла и порт сервера указаны правильно.Обратите внимание, что среда IDE вводит localhost как имя узла сервера по умолчанию и 3306 как номер порта сервера по умолчанию.

3. Введите имя администратора (если оно не отображается). Вам необходим доступ с правами администратора, чтобы иметь возможность создавать и удалять базы данных.

4. Введите пароль администратора. По умолчанию установлено пустое значение (пустой пароль является допустимым).

5. Нажмите вкладку «Свойства администратора» в верхней части диалогового окна.

Отобразится соответствующая вкладка, предоставляющая возможность ввода сведений для управления сервером MySQL
(рис. 4).

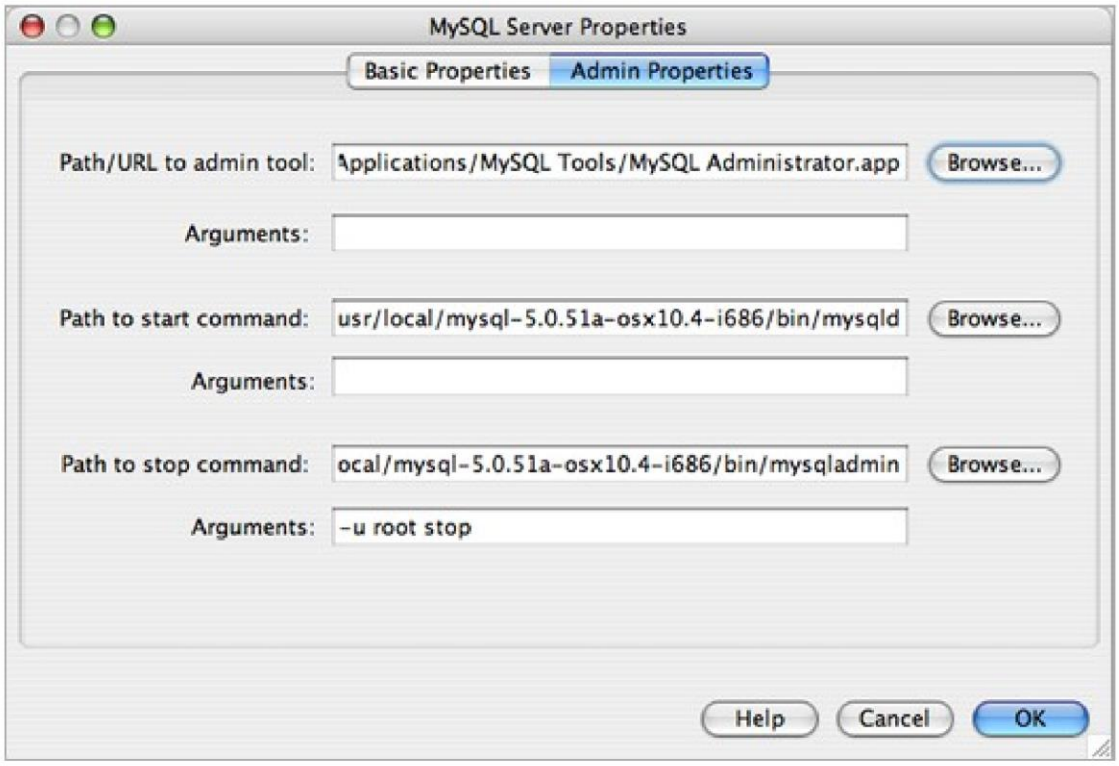


Рис. 4. Внешний вид вкладки «Свойства администратора»

6. В поле «Путь/URL-адрес к средству администрирования» введите путь к средству администрирования MySQL (например, ***MySQL Admin Tool, PhpMyAdmin*** или другому подходящему webсредству) или найдите его при помощи кнопки «Обзор».

7. В поле «Путь к команде запуска» введите соответствующийпуть MySQL или найдите его при помощи кнопки «Обзор». Для получения команды запуска найдите файл ***mysqld*** в папке ***bin*** каталога установки MySQL. Также может потребоваться другая команда запуска при установке MySQL в составе установки XAMPP.

8. В поле «Путь к команде остановки» введите путь к командеостановки MySQL или найдите его при помощи кнопки «Обзор». Обычно требуется ввести путь к файлу ***mysqladmin*** в папке ***bin*** каталога установки MySQL. При использовании команды ***mysqladmin*** введите ***-u root stop*** в поле «Аргументы» для получения прав пользователя ***root*** на остановку сервера.

9. Если настройка выполнена корректно, нажмите кнопку«ОК».

Запуск сервера MySQL

Перед попыткой подключения к серверу базы данных MySQL необходимо убедиться в том, что сервер запущен на компьютере. Если сервер базы данных не подключен, вы увидите ***disconnected*** рядом с именем

пользователя в узле MySQL Server в окне «Службы» и не сможете развернуть узел.

Для подключения к серверу баз данных убедитесь, что сервер базы данных MySQL запущен на компьютере, щелкните правой кнопкой мыши ***Базы данных > узел MySQL Server*** в окне «Службы» и выберите «Подключить». Может отобразиться запрос на ввод пароля для подключения к серверу.

После подключения сервера вы сможете развернуть узел MySQL Server и просмотреть все доступные базы данных MySQL.

Технология создания базы данных и подключения к ней

Язык SQL является широко распространенным способом взаимодействия с базами данных. Для этого в IDE NetBeans имеется встроенный редактор SQL. Обычно редактор SQL доступен с помощью параметра ***«Выполнить команду»*** из контекстного меню узла подключения (или дочерних узлов узла подключения). После установления подключения к серверу MySQL можно создать новый экземпляр базы данных в редакторе SQL. Для продолжения работы создайте экземпляр с именем ***MyNewDatabase***:

1. В окне «Службы» среды IDE щелкните правой кнопкой мыши узел сервера MySQL Server и выберите «Создать базу данных». Откроется диалоговое окно «Создание базы данных MySQL».
2. В диалоговом окне «Создание базы данных MySQL» введите имя новой базы данных ***MyNewDatabase***. Не устанавливайте флажок (рис. 5).

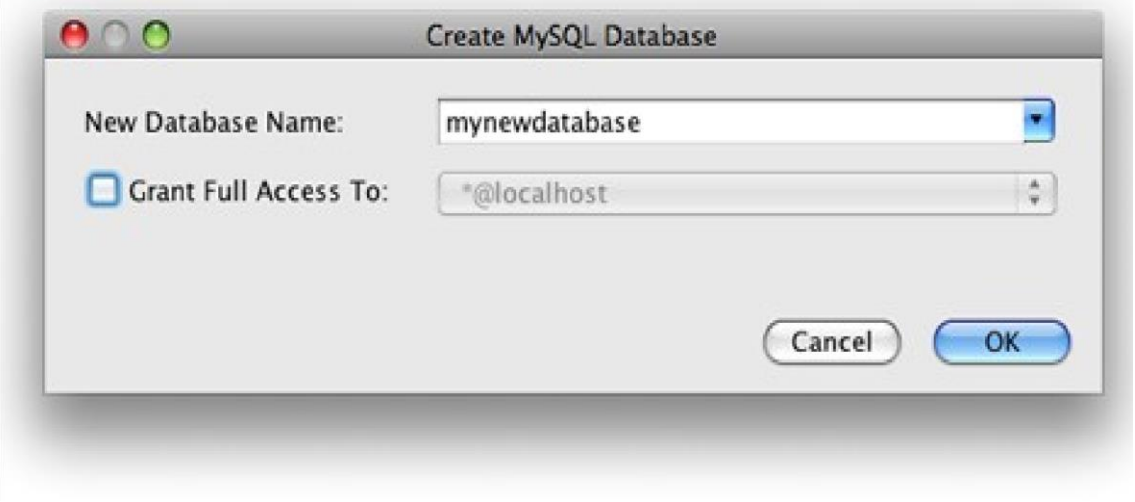


Рис. 5. Диалоговое окно «Создание базы данных MySQL»

Определенному пользователю можно предоставить полный доступ. По умолчанию только администратор обладает правами на выполнение определенных команд. Раскрывающийся список позволяет присваивать эти права определенным пользователям.

3. Нажмите кнопку «ОК». В узле «Сервер MySQL» окна «Службы» будет выведена новая база данных.

4. Щелкните узел новой базы данных правой кнопкой мыши ивыберите «Подключение», чтобы установить соединение с базой данных. Открытые подключения к базе данных отображаются в узле «Установленные подключения» в окне «Службы».

Создание таблиц базы данных

После установления подключения к базе данных ***MyNewDatabase*** можно начинать изучение принципов создания таблиц, заполнения их данными и изменения данных в таблицах.

База данных ***MyNewDatabase*** в настоящее время пуста. В среде IDE таблицу базы данных можно добавить при помощи диалогового окна «Создание таблицы» или посредством ввода запроса SQL и его запуска напрямую из редактора SQL. Можно использовать оба метода.

Использование редактора SQL

1. В проводнике баз данных разверните узел подключения ***MyNewDatabase***  и обратите внимание, что там содержится три подпапки: «Таблицы», «Представления» и «Процедуры».

2. Щелкните правой кнопкой мыши папку ***Tables*** («Таблицы») и выберите ***Execute Command*** («Выполнить команду»). В главном окне редактора SQL отобразится пустой холст.

3. В редакторе SQL введите следующий запрос. Это определение создаваемой таблицы ***Counselor***.

```
CREATE TABLE Counselor (
  id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
firstName VARCHAR (50),     nickName VARCHAR (50),
lastName VARCHAR (50),     telephone VARCHAR (25),     email
VARCHAR (50),
   memberSince DATE DEFAULT '0000-00-00',
   PRIMARY KEY (id)
        );
```

После выполнения запроса в окне «Вывод» будет создан отклик от механизма SQL, указывающий на успешность выполнения или на ошибку.

1. Чтобы выполнить запрос, нажмите кнопку «ВыполнитьSQL» на панели задач в верхней части (Ctrl-Shift-E) или щелкните правой кнопкой мыши в редакторе SQL Editor и выберите ***Выполнить оператор***. В среде IDE

будет создана таблица базы данных ***Counselor***, а на экране появится сообщение о выполнении запроса.

2. Для проверки изменений щелкните правой кнопкой мышиузел «Таблицы» в проводнике баз данных и выберите ***Обновить***.

Обратите внимание, что новый узел таблицы ***Counselor***  теперь отображается ниже узла «Таблицы»в проводнике баз данных. Если развернуть узел таблицы, можно увидеть созданные столбцы (поля), начинающиеся первичным ключом  .

Использование диалогового окна «Создание таблицы»

1. В проводнике баз данных щелкните правой кнопкой мышиузел «Таблицы» и выберите «Создать таблицу». Откроется диалоговое окно «Создание таблицы».
2. Введите ***Subject*** в текстовое поле «Имя таблицы».
3. Нажмите кнопку «Добавить столбец».
4. В поле Name («Имя») столбца введите id. Выберите ***SMALLINT*** в качестве типа данных из раскрывающегося списка ***Type***. Нажмите кнопку «ОК».
5. Установите флажок Primary Key («Первичный ключ») в диалоговом окне Add Column. В этом действии выполняется определение первичного ключа таблицы. Все таблицы, созданные в реляционных базах данных, должны содержать первичный ключ. Обратите внимание, что при выборе флажка «Ключ» выполняется автоматическая установка флажков «Индекс» и «Уникальный», при этом отменяется выбор флажка «Значение отсутствует». Это объясняется тем, что первичные ключи применяются для определения уникальной строки базы данных и по умолчанию используются в индексе таблицы. Поскольку все строки должны иметь уникальный идентификатор, первичные ключи не могут иметь значение Null.
6. Повторите эту процедуру, добавив оставшиеся столбцы, какпоказано в следующей таблице.
7. Выполняется создание таблицы ***Subject***, в которой будут содержатся данные для каждой из следующих записей (рис. 6).

- **Имя:** тема
- **Описание:** описание темы
- **Идентификатор таблицы Counselor:** идентификатор, соответствующий идентификатору в таблице Counselor.

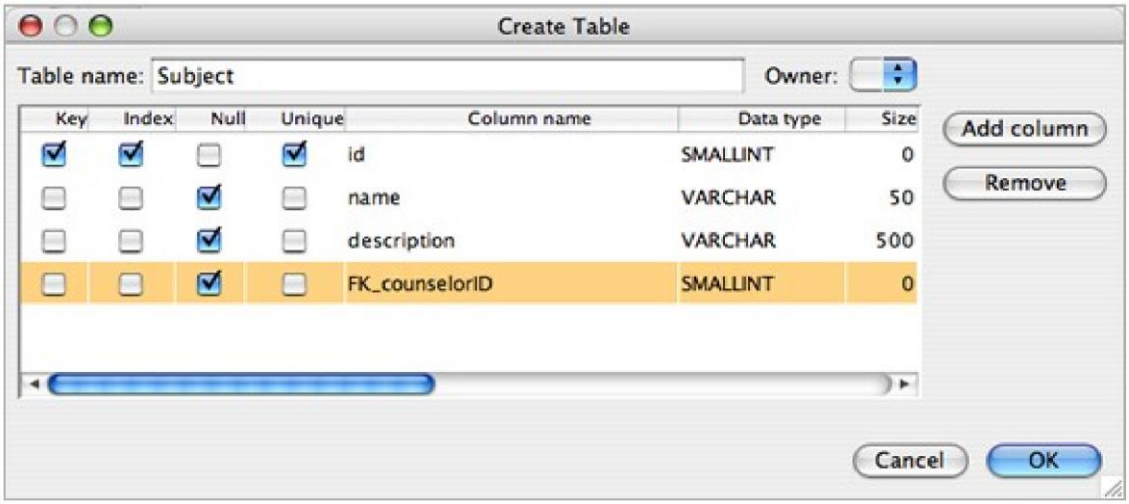


Рис. 6. Структура таблицы ***Subject***

Убедитесь, что все поля в диалоговом окне «Создание таблицы» соответствуют полям в примере выше и нажмите кнопку «ОК». IDE создает таблицу ***Subject*** в базе данных и можно увидеть, что новый узел таблицы Subject  отображается непосредст-венно под пунктом «Таблицы» в проводнике баз данных.

Создание пользователя базы данных

Перед созданием базы данных необходимо создать соответствующего пользователя, которому будет предоставлено право на выполнение любых операций в базе данных. Создание пользователя базы данных включает в себя следующие действия:

- Подключение к серверу *MySQL* пользователя с ролью *root*.
- Подключение к базе данных системы *MySQL* пользователя с ролью *root*. Поскольку выполнение команды *SQL* невозможно без подключения к какой-либо базе данных, это действие позволяет запустить команду *SQL*, необходимую для создания пользователя.
- Выполнение оператора создания пользователя в *MySQL*.

1. Запустите *IDE*, перейдите в окно «Службы» (Ctrl-5) и разверните узел 'Базы данных' (рис. 7).

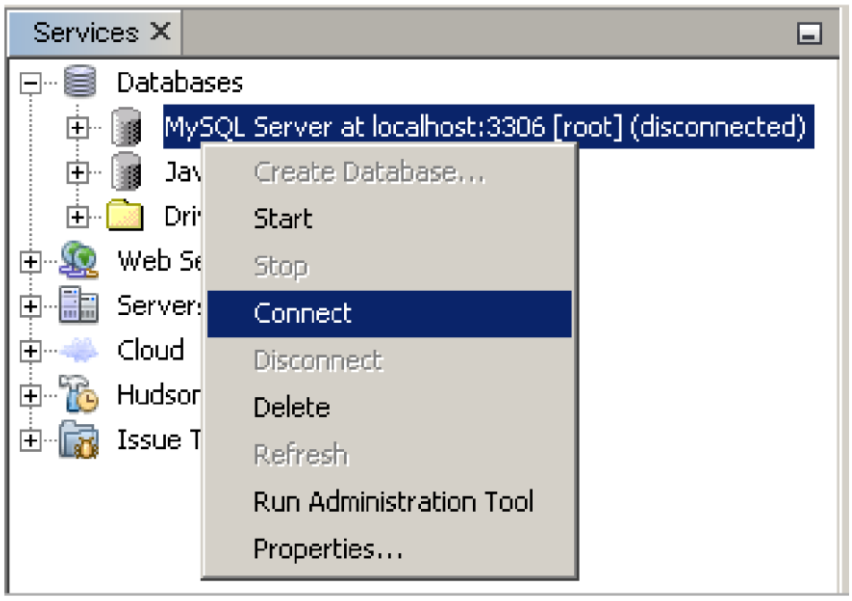


Рис. 7. Контекстное меню «Службы»

2. Для подключения к серверу базы данных *MySQL* перейдите к узлу *MySQL Server* и выберите *Connect* в контекстном меню.

3. *IDE NetBeans* устанавливает соединение с сервером *MySQL*, проверяет наличие доступных баз данных с помощью сервера, обнаруживает системную базу данных *mysql* и добавляет соответствующий новый узел *mysql* к дереву баз данных (рис. 8).

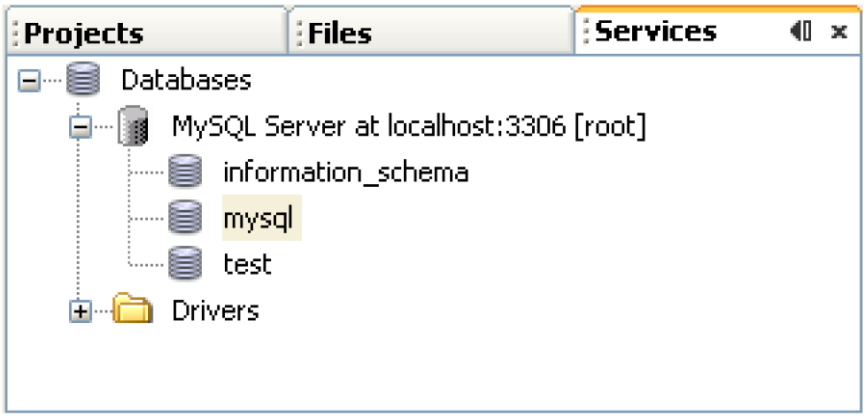


Рис. 8. Добавление нового узла к дереву баз данных

4. Для выполнения команды *SQL* необходимо подключение к базе данных. Поскольку доступна только система *MySQL*, следует подключиться к этой системе. Для подключения к системной базе данных перейдите к узлу *mysql* и выберите *Connect* в контекстном меню. Если подключение на данный момент отсутствует, появится диалоговое окно *New Database Connection*. В поле *User Name* по умолчанию вводится значение ***root.*** В поле *Password* введите пароль пользователя ***root***.

Если вы уже подключались к базе данных *mysql*, это диалоговое окно не отобразится. Вместо этого в дереве просто появится новый узел подключения.

В диалоговом окне *New Database Connection* появится сообщение *Connection established*. Нажмите кнопку *OK*. К дереву баз данных добавлен новый узел с именем jdbc:mysql://localhost:3306/mysql.

5. Перейдите к узлу jdbc:mysql://localhost:3306/mysql и выберите в контекстном меню *Execute Command*.

Откроется окно *SQL Command*. В окне *SQL Command* введите следующие команды:

```
CREATE USER 'phpuser'@'localhost'
IDENTIFIED BY 'phpuserpw'
```

Выберите в контекстном меню *Run Statement*. Если команда выполнена успешно, в строке состояния выводится следующее сообщение: *SQL Statement(s) executed successfully*. Если выводится другое сообщение, проверьте синтаксическую правильность введенных команд и выполните советы, относящиеся к данному сообщению.

Создание базы данных Wishlist

Для создания базы данных выполните следующие действия:

1. Перейдите к узлу *MySQL Server at localhost:3306* и выберите из контекстного меню *Create Database*. Появится диалоговое окно *Create MySQL Database*. Заполните поля следующим образом:

- В поле имени *Database Name* введите *wishlist*.
- Установите флажок ***Grant full access to user*** и выберите в раскрывающемся списке ***phpuser@localhost***. Нажмите кнопку *OK*.

Функция *Grant full access to user*, предоставляющая пользователю полные права доступа, срабатывает не всегда. Если она не работает, подключитесь к базе данных как пользователь *root* и отправьте запрос *SQL*.

```
GRANT ALL ON wishlist.* TO phpuser@localhost.
```

2. Подключение к базе данных появится в дереве. Однако это подключение создано для пользователя *root*. Вам требуется подключение для пользователя *phpuser*.

Установление подключения к базе данных Wishlist

На предыдущем этапе вы создали базу данных wishlist с подключением для пользователя root. Теперь необходимо создать подключение для пользователя phpuser:

1.В окне «Службы» щелкните правой кнопкой мыши узел «Базы данных» и выберите «Создать подключение». Открывается мастер создания подключений. На панели «Обнаружение драйвера» в мастере создания подключений выберите *MySQL (Connector/ J Driver)*. Нажмите «Далее». Открывается панель «Настройка соединения». В поле «База данных» введите ***wishlist***.

2. В полях *User Name* и *Password* введите соответственно имя и пароль пользователя, указанные в разделе «Создание владельца (пользователя) базы данных» (в нашем примере – это phpuser и phpuserpw). Установите флажок «Запомнить пароль». Нажмите «Проверить подключение». Если соединение установлено успешно, нажмите *OK*. В дереве баз данных будет отображаться соответствующий новый узел подключения. Подключение для пользователя root к базе данных wishlist можно удалить.

 Нажмите на подключение

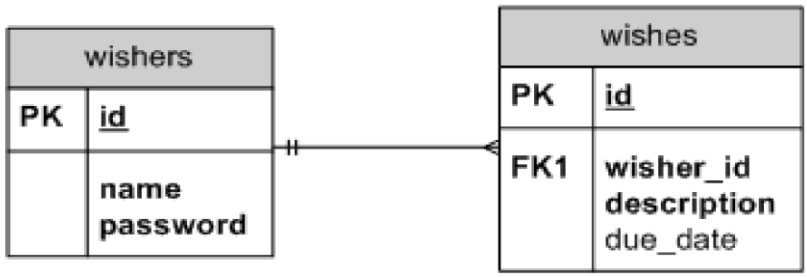
jdbc:mysql://localhost:3306/wishlist и

выберите «Удалить».

Проектирование структуры базы данных Wishlist

Для размещения и сохранения всех необходимых данных требуются две таблицы:

Таблица 1



- Таблица *wishers* – для сохранения имен и паролей зарегистрированных пользователей.
- Таблица *wishes* – для содержания описания требований.

Таблица *wishers* содержит три поля:

1) *id* – уникальный идентификатор пользователя. Это поле используется в качестве первичного ключа;

2) *name* – имя;

3) *password* – пароль.

Таблица wishes содержит четыре поля:

1) id – уникальный идентификатор пользователя. Это поле используется в качестве первичного ключа;

2) *wisher_id* – идентификатор пользователя, оставившего пожелание. Это поле используется в качестве внешнего ключа; 3) *description* – описание;

4) *due_date* – требуемая дата исполнения пожелания.

Таблицы связаны посредством идентификатора пользователя. Все поля таблицы *wishes* являются обязательными для заполнения, за исключением *due_date*.

Создание таблиц

1.Для подключения к базе данных щелкните правой кнопкой мыши узел подключения jdbc:mysql://localhost:3306/wishlist и выберите *Connect* в контекстном меню. Если пункт меню недоступен, пользователь уже подключен. Перейдите к действию 2.

2.В том же контекстном меню выберите *Execute Command*. Откроется пустое окно *SQL Command*.

3.Для создания таблицы *wishers*. Введите следующий запрос SQL (отметьте, что как набор символов следует прямо установить UTF-8 для интернационали-зации):

```
CREATE TABLE wishers(
 id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 name CHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT
NULL UNIQUE,
 password CHAR(50) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL
)
```

Можно получить уникальный номер, автоматически создаваемый *MySQL*, задав свойство *AUTO_INCREMENT* для поля. *MySQL* создаст уникальный номер посредством увеличения на единицу последнего номера в таблице и автоматически добавит его к значению поля с этим свойством. В нашем примере автоматически должно увеличиваться значение в поле *ID*.

Щелкните запрос правой кнопкой мыши, затем выберите *Run Statement* в контекстном меню.

Механизмом хранения по умолчанию для *MySQL* является *MyISAM*, не поддерживающий внешние ключи. Если нужна поддержка внешних ключей, используйте в качестве механизма хранения *InnoDB.*

Для создания таблицы *Wishes* введите следующий запрос *SQL*:

```
CREATE TABLE wishes(
 id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 wisher_id INT NOT NULL,
 description CHAR(255) CHARACTER SET utf8 COLLATE utf8_general_ci
NOT NULL,  due_date DATE,
 FOREIGN KEY (wisher_id) REFERENCES wishers(id)
)
```

Щелкните запрос правой кнопкой мыши, затем выберите *Run Statement* в контекстном меню. Для проверки того, что новые таблицы добавлены к базе данных, перейдите к окну *Services*, а затем к узлу подключения jdbc:mysql://localhost:3306/wishlist. Нажмите правую кнопку мыши и выберите *Refresh*. В дереве появятся узлы *wishers* и *wishes*.

Ввод тестовых данных

Для тестирования приложения необходимо наличие некоторых данных в базе данных. В приведенном ниже примере показано, каким образом можно добавить данные для двух пользователей и четырех желаний.

В узле подключения jdbc:mysql://localhost:3306/wishlist щелкните правой кнопкой мыши и выберите *Execute Command*. Откроется пустое окно *SQL Command*.

Для добавления данных пользователя введите следующие команды:

```
INSERT INTO wishers (name, password)
VALUES ('Tom', 'tomcat');
```

Щелкните запрос правой кнопкой мыши и выберите из контекстного меню *Run Statement*.

Оператор не содержит значения для поля идентификатора. Значения вводятся автоматически, поскольку указан тип поля AUTO_INCREMENT.

Введите данные другого тестового пользователя:

```
INSERT INTO wishers (name, password)
VALUES ('Jerry', 'jerrymouse');
```

Для добавления пожеланий (*wishes*) введите следующие команды:

```
INSERT INTO wishes (wisher_id, description, due_date)
```

```
 VALUES (1, 'Sausage', 080401);
INSERT INTO wishes (wisher_id, description)
 VALUES (1, 'Icecream');
INSERT INTO wishes (wisher_id, description, due_date)
 VALUES (2, 'Cheese', 080501);
INSERT INTO wishes (wisher_id, description)
 VALUES (2, 'Candle');
```

Выберите запросы, щелкните каждый правой кнопкой мыши по каждому из них и выберите *Run Selection* в контекстном меню.

Для просмотра тестовых данных щелкните соответствующую таблицу правой кнопкой мыши и выберите из контекстного меню *View Data*.

Лабораторная работа 6. Разработка web-приложения на языке PHP. Чтение из базы данных

Цель работы – научиться с помощью NetBeans создавать и настраивать проект PHP для чтения из базы данных.

В этой лабораторной работе рассматривается создание и настройка проекта PHP для разработки приложения, создание списка страниц в приложении и определение отношений между ними. Кроме того, основная функциональность разрабатывается и тестируется на данных, введенных в пример базы данных в лабораторной работе
5.

Создаваемый в этой лабораторной работе код PHP выполняет следующие функции.

1. Получает имя лица, введенного пользователем.
2. Проверяет наличие этого лица в базе данных. Если лицо отсутствует в базе данных, выполняется выход с сообщением об ошибке.
3. Отображение таблицы желаний этого лица.

Текущий документ является частью краткого учебного курса «Создание приложения, управляемого базой данных, в IDE NetBeans для PHP».

Создание проекта PHP

Выберите *Файл* > *Создать проект* (Ctrl-Shift-N в Windows).

Создайте новый проект PHP с именем ***wishlist***. После создания проекта PHP по умолчанию он будет содержать файл индекса ***index.php***.

Определение блок-схемы приложения

В рамках приложения возможны следующие варианты использования:

1.Пользователь просматривает список *Wish list* другого пользователя.

2.Пользователь регистрируется в качестве нового пользователя.

3.Пользователь входит в систему и создает собственный список желаний *Wish list*.

4.Пользователь входит в систему и редактирует свой список желаний.

Для реализации этих базовых функциональных возможностей потребуется добавить следующие файлы PHP (рис. 9):

1. Первая страница *index.php* для входа в систему, регистрации и перехода к спискам *Wish list* других пользователей.
2. Страница *wishlist.php* для просмотра списка *Wish list* конкретного пользователя.

3. Страница *createNewWisher.php* для регистрации в качестве автора желаний.

4. Страница *editWishList.php* для редактирования списка его владельцем.

5. Страница *editWish.php* для создания и редактирования желаний.

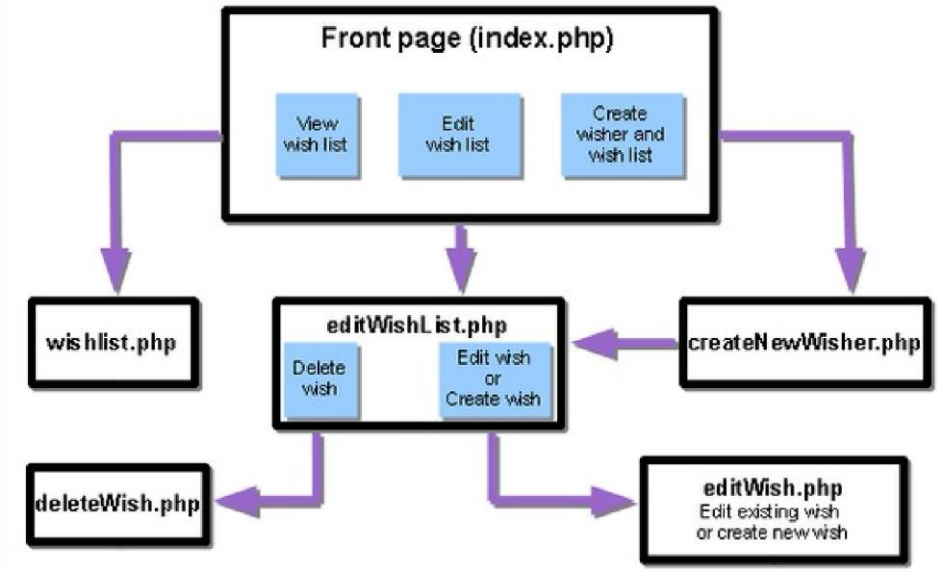


Рис. 9. Блок-схема приложения

После выполнения предварительных действий можно приступить к реализации базовой функциональности приложения. Начать следует с просмотра списка *Wish list* одного из пользователей. Для этой функции не требуется выполнять проверку допустимости, ее можно легко протестировать, поскольку в базу данных уже внесены тестовые данные. Функциональность компонента реализуются на двух страницах – *index.php* и *wishlist.php*.

Добавление формы к index.php

Файл *index.php* не содержит код PHP, таким образом, можно просто удалить следующий блок.

Файл *index.php* используется в двух целях:

1. Отображение страницы с элементами управления для вводаданных.

2. Передача введенных данных в другой файл PHP для обработки данных. Данные передаются в файл с именем *wishlist.php*, который создается в следующем разделе.

Эти действия выполняются с использованием формы HTML. Каждая форма HTML содержит следующее:

1. Набор полей, соответствующих элементам управления настранице.

2. «Действие», выполняемое после отправки пользователемданных в форме. Действие представлено путем к странице для обработки данных.

Для добавления формы к *index.php* выполните следующее.

- Перейдите к окну «Проекты», разверните узел проекта и узел «Файлы исходного кода», затем дважды щелкните файл index.php. Файл index.php откроется в основной области редактора IDE. Файл содержит шаблон для ввода кодов PHP и HTML.
- Удалите блок PHP. Файл *index.php* не содержит код PHP. Откройте «Палитру» из меню «Окно» или нажав *Ctrl-Shift-8*.
- Из раздела **HTML** палитры (рис. 10) перетащите форму в раздел <body> файла index.php.

Рис. 10. Файл index.php и открытая палитра

Откроется диалоговое окно «Вставить форму» (рис. 27). В поле «Действие» введите путь к файлу, в которой форма будет передавать данные. В данном случае введите ***wishlist.php***. (Этот файл будет создан в том же месте, что и файл ***index.php***.)

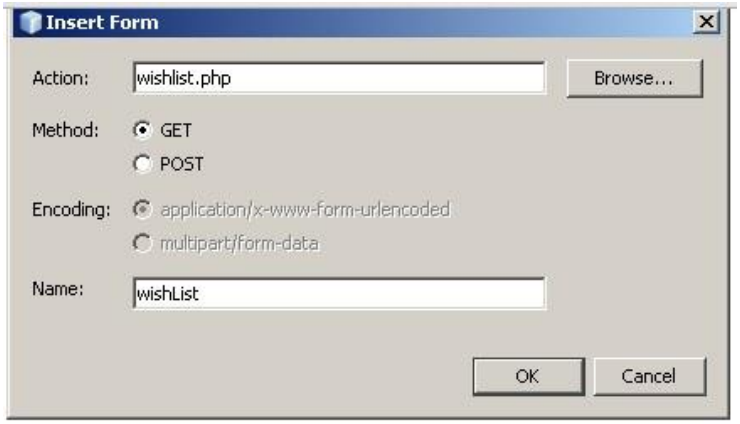


Рис. 11. Диалоговое окно «Вставить форму»

Выберите метод ***GET*** для передачи данных. Присвойте форме произвольное имя, например wishList. Нажмите кнопку *OK* после выполнения действия.

Теперь файл выглядит следующим образом (рис. 12).

Рис. 12. Файл index.php с формой

Между открывающим и закрывающим тегами формы введите текст «Показать список желаний:».

Перетащите компонент «Ввод текста» из раздела **Формы HTML** палитры в пространство после текста «Показать список желаний:». Откроется диалоговое окно «Вставка ввода текста».

Присвойте вводу название ***user***. Выберите тип ввода text. Оставьте все поля пустыми и нажмите кнопку «ОК». Добавьте пустую сроку над тегом </form>. В эту пустую строку перетащите компонент «Кнопка» из раздела **Формы HTML** палитры.

Откроется диалоговое окно «Вставить кнопку». Введите Go в поле «Метка» и нажмите кнопку «ОК».

Теперь форма выглядит так, как показанный ниже код, с одним отличием. В коде ниже атрибут *method* явно указан в теге <form>. *IDE NetBeans* не добавил атрибут метода к используемой форме, поскольку значением по умолчанию этого атрибута является ***GET***. Однако явное указание атрибута ***method*** упрощает понимание кода.

```
<form action="wishlist.php" method="GET" name="wishList">
    Show wish list of: <input type="text" name="user" value=""/>
            <input type="submit" value="Go" />
</form>
```

Открывающий тег ***<form>*** содержит атрибут action. Атрибут *action* указывает файл, в который форма передает данные. В данном случае файл имеет имя ***wishlist.php*** и находится в той же папке, что и файл *index.php*. (Этот файл будет создан в разделе *Создание wishlist.php* и *тестирование приложения*.)

Открывающий тег ***<form>*** также содержит метод для применения к переданным данным (GET). PHP использует массив $_GET или $_POST для значений, переданных этой формой, в зависимости от значения атрибута *method*. В данном случае PHP использует $_GET.

Компонент ввода ***text*** – текстовое поле для ввода имени пользователя, список пожеланий которого необходимо просмотреть. Начальное значение текстового поля – пустая строка. Имя этого поля – ***user***. PHP использует имя поля при создании массива для значений поля. В данном случае массив для значений этого поля – ***htmlentities($_GET["user"])***.

Компонент ввода **submit** со значением *Go*. Тип *submit* означает, что поле ввода отображается на странице как кнопка. Значение *Go* – это метка поля. При нажатии пользователем кнопки данные в компоненте text передаются в файл, указанный в атрибуте action.

Создание страницы wishlist.php и тестирование приложения

В разделе «Добавление формы к index.php» была создана форма, с помощью которой пользователь отправляет имя лица, список пожеланий которого необходимо просмотреть. Имя передается странице *wishlist.php*. Однако этой страницы не существует. Если выполнить *index.php*, при отправке имени возникнет ошибка «404: Файл не найден». В этом разделе будет создан файл wishlist.php, затем будет выполнено тестирование приложения.

Для создания ***wishlist.php*** и тестирования приложения выполните следующие действия.

В созданном проекте ***wishlist*** щелкните правой кнопкой мыши узел «Исходные файлы» и выберите «Создать > Файл PHP» в контекстном меню. Откроется мастер создания web-страниц PHP.

Введите wishlist в поле «Имя файла» и нажмите кнопку «Готово».

Щелкните правой кнопкой мыши узел «Источники» и выберите «Выполнить проект» в контекстном меню или щелкните значок «Выполнить главный проект» на панели инструментов, если проект задан как главный.

В поле *Show wish list* of введите *Tom* и нажмите *Go*. Появится пустая страница со следующим URL-адресом: ***http://localhost:90/Lesson2/wishlist.php?user=tom.***

Наличие этого URL-адреса означает, что главная страница функционирует правильно.

Установка подключения и получение идентификатора автора пожеланий

Дважды щелкните файл *wishlist.php*. Открывшийся шаблон отличается от *index.php*. Файл должен начинаться и заканчиваться тегами <html></html> и <body></body>, поскольку файл будет содержать также код HTML.

```
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title></title>
   </head>
   <body>
      <?php
      // put your code here
      ?>
   </body>
</html>
```

Для отображения заголовка после тега открытия <body> и перед генерируемым тегом <?php введите следующий блок кода:

```
Wish List of <?php echo htmlentities($_GET["user"])."<br/>";?>
```

Теперь код должен выглядеть следующим образом:

```
<body>Wish List of <?php echo htmlentities($_GET["user"])."<br/>";?>     <?php


   // put your code here


</body>
```

Блок кода PHP выводит на экран данные, которые поступают посредством метода ***GET*** в поле ***user***. Это данные передаются со страницы ***index.php***, где имя владельца списка *Wish list – Tom* было введено в текстовом поле ***user***.

Удалите раздел в шаблоне блока PHP с комментарием. В этом месте введите или вставьте следующий код. Этот код открывает подключение к базе данных.

```
$con = mysqli_connect("localhost", "phpuser", "phpuserpw"); if
(!$con) {
```

```
    exit('Connect Error (' . mysqli_connect_errno() . ') '
        . mysqli_connect_error());
}
//set the default client character set mysqli_set_charset($con, 'utf-8');
```

В соответствии с кодом производится попытка подключения к базе данных и выдается сообщение об ошибке в случае неудачи.

Под фрагментом кода, описывающим подключение к базе данных, в том же блоке PHP укажите следующий код:

```
mysqli_select_db($con, "wishlist");

$user = mysqli_real_escape_string($con, htmlentities($_GET["user"]));

$wisher = mysqli_query($con, "SELECT id FROM wishers WHERE name='" .
$user . "'");

if (mysqli_num_rows($wisher) < 1) {     exit("The person " .
htmlentities($_GET["user"]) . " is not found.
Please check the spelling and try again");
}
$row = mysqli_fetch_row($wisher);
$wisherID = $row[0]; mysqli_free_result($wisher);
```

Этот код получает идентификатор автора желаний, чей список был запрошен. Если автор пожеланий отсутствует в базе данных, код уничтожает/завершает процесс и отображает сообщение об ошибке.

Осуществляется выбор данных из базы данных ***wishlist*** с помощью подключения ***$con***. Критерием отбора является имя, полученное со страницы ***index.php*** как ***user***.

Синтаксис оператора SQL SELECT может быть кратко описан следующим образом:

1. После оператора SELECT укажите поля, из которых должны быть получены данные. Все поля отмечены звездочкой (*).
2. После блока FROM укажите имя таблицы, из которой требуется извлечь данные.
3. Блок WHERE является необязательным. Укажите в нем условия отбора.

Запрос ***mysqli*** возвращает объект результата. Выполняется выборка строки из результатов выполненного запроса и извлекается значение строки идентификатора, которое сохраняется в переменной ***$wisherID***.

Наконец, освобождается результат ***mysqli***.

Для физического закрытия подключения необходимо освободить все ресурсы, использующие подключение. В противном случае внутренняя система подсчета ссылок PHP сохранит нижележащее подключение к базе данным открытым, даже если ***$con*** неприменимо после вызова ***mysqli_close()***.

Для *MySQL* параметр **htmlentities($_GET["user"])** используется с escape-символом для предотвращения SQL-инъекций. Рекомендуется исключить из участия в запросах *MySQL* такие строки, которые могли бы быть подвержены подобной атаке.

На данный момент блок PHP готов. Файл ***wishlist.php*** теперь выглядит следующим образом:

```
Wish List of <?php echo htmlentities($_GET["user"]) . "<br/>"; ?>

<?php

 $con = mysqli_connect("localhost", "phpuser", "phpuserpw");   if
(!$con) {
    exit('Connect Error (' . mysqli_connect_errno() . ') '
          . mysqli_connect_error());
                                                                    }

 //set the default client character set   mysqli_set_charset($con, 'utf-8');
 mysqli_select_db($con, "wishlist");
              $user          =          mysqli_real_escape_string($con,
htmlentities($_GET["user"]));
 $wisher = mysqli_query($con, "SELECT id FROM wishers WHERE name='" .
$user . "'");
 if (mysqli_num_rows($wisher) < 1) {                exit("The person " .
htmlentities($_GET["user"]) . " is not found. Please check the spelling and try
again");
 }
 $row = mysqli_fetch_row($wisher);
 $wisherID = $row[0];
 mysqli_free_result($wisher);
 ?>
```

Отображение таблицы желаний

В этом разделе будет добавлен код для отображения таблицы HTML желаний, связанных с автором желаний. Автор желаний определяется идентификатором, полученным в коде предыдущего раздела.

Под блоком PHP введите или вставьте следующий блок кода HTML:

```
<table border="black">
   <tr>
      <th>Item</th>
      <th>Due Date</th>
   </tr>
</table>
```

Этот код открывает таблицу, указывает цвет ее границ (черный) и определяет вид заголовка таблицы, содержащего столбцы ***Item*** и ***Due Date***. Тег ***</table>*** закрывает таблицу.

Введите следующий код блока PHP над закрывающим тегом ***</table>:***

```
<?php
$result = mysqli_query($con, "SELECT description, due_date FROM wishes
WHERE wisher_id=" . $wisherID); while ($row = mysqli_fetch_array($result))
{
   echo "<tr><td>" . htmlentities($row["description"]) . "</td>";     echo
"<td>" . htmlentities($row["due_date"]) . "</td></tr>\n";
}
mysqli_free_result($result);
mysqli_close($con);
?>
```

Посредством запроса ***SELECT*** желания со сроками их выполнения для указанного пользователя извлекаются в соответствии с идентификатором, который, в свою очередь, был извлечен в действии 4. Кроме того, желания и сроки их выполнения сохраняются в массиве ***$result***.

С помощью цикла отдельные элементы массива ***$result*** выводятся на экран в качестве строк таблиц, пока массив непуст.

Теги ***<tr></tr>*** формируют строки, теги ***<td></td>*** – ячейки внутри строк, а после символа ***\n*** начинается новая строка.

Функция ***htmlentities*** преобразует все символы, имеющие эквивалентные сущности HTML, в сущности HTML. Это помогает предотвратить межсетевые_сценарии.

В конце функции освобождают все ресурсы (результаты ***mysqli***) и закрывают подключение к базе данных. Имейте в виду, что для физического закрытия подключения необходимо освободить все ресурсы, использующие подключение к базе данных. В противном случае внутренняя система подсчета ссылок PHP сохранит подключение к базе данным открытым, даже если подключение неприменимо после вызова ***mysqli_close()***.

Убедитесь, что названия полей базы данных введены точно так, как они указаны при создании таблицы базы данных.

Лабораторная работа 7. Программирование на JavaScript. Обработка событий

Цель работы – научиться использовать имеющиеся в модели документа события для внесения изменений в страницу.

Наиболее часто в сценариях используется событие ***onclick***. Для того чтобы обратить внимание пользователя на определённый элемент HTML-документа, можно менять свойства этого элемента при наведении на него курсора мыши, а при снятии курсора – восстанавливать прежние значения свойств. Например, можно менять цвет или размер элемента. Попадание курсора мыши на элемент фиксируется событием ***onMouseOver***. Парное для него событие ***onMouseOut*** происходит при снятии курсора мышки с элемента. Эту пару событий удобно применять для изменения свойств элементов или замены элементов на время удержания кнопки мыши нажатой.

Реакция на событие в отдельном элементе

Так как в объектной модели объекты могут быть вложены друг в друга, то событие, происходящее в дочернем объекте, одно-временно происходит и в родительском объекте. JavaScript предос-тавляет различные способы локализации влияния события на иерархию объектов. Простейшей способ локализации (пример 1) заключается в размещении сценария в теге, на который должно воздействовать событие.

Пример 1

```
<html>
  <body>
    <P align=right ID='alfa'
     onMouseOver="document.all.alfa.align='center'
    "onMouseOut="this.align='left'">
     События onMouseOver и onMouseOut </p>
</body>
</html>
```

Страница в примере 1 состоит из одной строки, заключённой в контейнер ***<P> …</p>.*** В объектной модели страницы событие, происходящее с объектом **P**, происходит также и с родительским объектом ***BODY***. Чтобы локализовать реакцию на событие только пределами строки, т. е. объекта **P**, сценарий реакции на события помещен в тег ***<P >***.

В результате исполнения сценария изменяется положение текста ***d*** строке. Первоначально строка выровнена по правому краю окна. При

попадании на неё курсора строка выравнивается по центру, а после снятия курсора выравнивается по левому краю окна. Для обращения к объекту используется коллекция *all*, которая правильно воспринимается браузерами Internet Explorer 6.0 и Mozilla Firefox 2.0. Ключевое слово **this** означает ссылку на текущий объект.

Если при наступлении события нужно произвести много действий, то удобно написать сценарий в виде функции и поместить её отдельно от элемента в специально предназначенный для сценариев контейнер ***<script> …</script>***. В примере 2 каждое из событий **onMouseOver** и **onMouseOut** вызывает два действия – выравнивание и изменение цвета текста в строке.

Пример 2

```
<html>
  <p  align=right ID='alfa' onMouseOver="M_Over()"
      onMouseOut="M_Out()">
Событие onMouseOver</p>
   <script>
    function M_Over()
    {
       document.all.alfa.align='center'
       document.all.alfa.style.color='FF00FF'
     }
   function M_Out()
   {
     document.all.alfa.align='left'
     document.all.alfa.style.color='0000FF'
    }
   </script> </html>
```

Задание 1

Разработайие HTML-документ, отображающийся в окне браузера в виде следующих четырёх строк:

1. Пять событий с мышкой
2. Щёлкните по мне мышкой
3. На этом тексте нажмите, подержите и отпустите левую кнопку мышки
4. Медленно проведите курсором мышки по этой надписи

Первая строка – заголовок страницы. Вторая строка меняется при щелчке мышкой следующим образом:

– шрифт увеличивается до 48 pt;

– цвет шрифта меняется на белый;– цвет фона меняется на голубой.

Повторный щелчок мышкой возвращает вторую строку к первоначальному виду.

Фон третьей строки меняется, когда курсор мышки находится на ней и нажимается или отпускается левая кнопка мышки. При нажатии фон становится зелёным, а при отпускании – жёлтым.

При попадании курсора мышки на четвёртую строку её фон становится красным, а при снятии – голубым.

Фиксация события в родительском элементе

Если реакцию на какое-либо событие требуют несколько элементов, расположенных на странице, то можно вызвать функцию для обработки этого события только в родительском элементе. В функции определяется, на каком элементе произошло событие, и выполняются соответствующие действия. Удобство такого подхода состоит в том, что весь алгоритм преобразований находится в одном месте, а недостаток – в сложности самой функции. Рассмотрим сценарий (пример 3), в котором для изменения свойств любого из трёх объектов, находящихся в окне браузера, служит одна функция.

Пример 3

```
<html>
  <head>
    <title>Реакция на событие</title>
    <meta http-equiv="Content-Type" content="text/html;      charset=windows-1251">
   <style>H1 {color:FF00FF}
      #k1{position:absolute; left:50;top:200;width:300;    height:100;background-color:blue}
      #k2{position:relative; left:50;top:25; width:200; height:50;background-color:yellow}
    </style>
  </head>
  <BODY ID="B" onclick="rodEl(event)">
    <H1 ID="HH" >ЦВЕТ  </H1>
    <DIV ID="k1" >
    <DIV ID="k2" > </div>
    </div>
  <SCRIPT>
```

```
/*      Функция запускается при щелчке мышкой по любой точке документа
*/
        flag=0;
        function rodEl(evt)
        {
         var e = evt || window.event;  //e -это объект event
         var elem = e.target || e.srcElement; //elem - элемент (объект),
          // на котором произошло событие
         id1=elem.id
         z1=document.getElementById(id1)
switch(id1)
        {
          case "k1": //Изменение цвета внешнего прямоугольника
z=z1.style.backgroundColor            if(z!="red")z="red"            else
z="green"           z1.style.backgroundColor=z
          break
          case "k2":   //Изменение цвета внутреннего прямоугольника
z=z1.style.backgroundColor
         if(z!='rgb(0, 255, 255)'){z='rgb(0, 255, 255)'}
            else{z='rgb(0, 255, 0)'}          z1.style.backgroundColor=z
        break
        case "B": //Изменение цвета заднего плана документа
z=z1.style.backgroundColor
          if(z!='rgb(190, 190, 190)'){z='rgb(190, 190, 190)'}
else {z='rgb(0, 190, 190)'}            z1.style.backgroundColor=z
break
      case "HH": //Изменение цвета слова "Цвет"          if(flag==0)
     {  document.getElementById(id1).style.color='rgb(170,0,170)';
flag=1;
     }
else
    {   document.getElementById(id1).style.color='rgb(0,255,170)';          flag=0;
    }
 }
}
```

```
</SCRIPT>
</BODY>
</HTML>
```

В примере 3 родительским по отношению к элементу *H1* и двум элементам **DIV** является элемент **BODY**. Поэтому в теге ***<BODY>*** вызывается функция ***rodEl()***, служащая для обработки события **onclick**.

В момент наступления события вся информация о нём запоминается в объекте ***event***. Этот объект по-разному описывается в стандарте W3C и в браузере *Internet Explorer*. Новые версии *Internet Explorer* поддерживают и стандарт W3C.

В стандарте W3C объект **event** передаётся в функцию в качестве параметра, а для обращения к объекту, на котором произошло событие, служит свойство ***event.target***.

В *Internet Explorer* объект ***window.event*** – глобальный, и поэтому передавать его в функцию в виде параметра не нужно. Для обращения к объекту, на котором произошло событие, в *Internet Explorer* служит свойство ***window.event.srcElement.***

В примере 3 первые две строчки тела функции ***rodEl()*** служат для кроссплатформенного (в любом браузере) обращения к объекту, на котором произошло событие.

В следующих строках функции ***rodEl()*** сначала определяется **Id** элемента, по которому пользователь щёлкнул мышкой, а затем с помощью оператора **Switch** делается переход к изменению свойств указанного элемента.

Задание 2

Создайте страницу с любым изображением и подписью под ним. При щелчке по подписи надпись должна менять свой цвет. Щелчок по изображению должен вызывать замену изображения и подписи. Функция для обработки события должна вызываться из родительского по отношению к изображению и подписи объекта.

Предотвращение всплывания события. Свойство cancelBubble

В примере 3 рассматривалась простая страница с небольшим количеством элементов, но даже в таком простом случае функция реакции на событие получилась сложной. Проще для каждого элемента написать свою функцию обработки события, а распространение события вверх по дереву иерархической структуры от «детей» к «родителям» (в этом случае говорят о всплывании события) блокировать с помощь специально для этого предназначенного свойства **cancelBubble** объекта event. Изменим пример 3 так, чтобы для реакции на щелчок по каждому из четырёх элементов страницы служила своя функция.

Пример 4

```
<HTML><HEAD><TITLE>Реакция на событие</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<STYLE>H1 {color:#ff00ff}
        #k1{position:absolute; left:50;top:200;
width:300; height:100;background-color:blue}
#k2{position:relative; left:50;top:25;       width:200;
height:50;background-color:yellow}
</STYLE></HEAD>
<BODY ID="B" bgcolor="AAAAAA" onclick="rodEl()" style="height:600px">
      <H1 ID="HH" onclick="H_l()">ЦВЕТ</H1>
      <DIV ID="k1" onclick="D_l()">
            <DIV ID="k2" style="background-color:yellow"
onclick="D_2(this)"></div>
      </div>
</BODY>
<SCRIPT>
/*     Функция запускается при щелчке мышкой по
      любой точке документа */
function rodEl()
{    //Изменение цвета заднего плана документа var
z=document.all.B.bgColor
if(z!="#777777"){z="#777777"}       else{z="AAAAAA"}
   document.all.B.bgColor =z
}
function D_l()
 {     //Изменение цвета внешнего прямоугольника

  var  z=document.all.k1.style.backgroundColor
      if(z!="red") z="red"
else z="green"
      document.all.k1.style.backgroundColor=z
}
/*function D_2()
{      //Изменение цвета внутреннего прямоугольника     var
z=document.all.k2.style.backgroundColor       if(z=="#ff00ff" ||
z=='rgb(255, 0, 255)'){z="#00ffff"}       else{z="ff00ff"}
      document.all.k2.style.backgroundColor=z
}*/
function D_2(thi)
```

```
{       //Изменение цвета внутреннего прямоугольника     var
z=thi.style.backgroundColor
       if(z=="#ff00ff" || z=='rgb(255, 0, 255)'){z="#00ffff"}
else{z="ff00ff"}
       thi.style.backgroundColor=z
}
function H_l()
{       //Изменение цвета слова "Цвет"     var
z=document.all.HH.style.color     //alert("1.
z="+document.all.HH.style.color)
   if(z=="#aa00aa" || z=='rgb(170, 0, 170)')
       {       z="#00ffff"
       //alert("2. z="+z)
       }
       else{z="#aa00aa"}         document.all.HH.style.color=z
}
</SCRIPT></HTML>
```

Скопируйте в свой каталог и просмотрите пример 4 в браузере Internet Explorer. Щёлкните по слову *ЦВЕТ*. Изменится не только цвет слова, но и цвет фона документа, так как после щелчка сначала выполнится функция ***H_1()***, а затем событие всплывёт к родительскому элементу ***BODY*** и выполнится функция ***rodE1().*** При щелчке по внутреннему прямоугольнику будут меняться цвета обоих прямоугольников и фона документа. Щелчок по внешнему прямоугольнику изменит цвет этого прямоугольника и цвет фона документа.

Задание 3

Чтобы предотвратить всплывание события в примере 4, вставьте в начало всех функций, кроме первой, оператор window.event.cancelBubble=true

Лабораторная работа 8. Программирование на JavaScript. Работа с формами

Цель работы – научиться использовать формы для ввода данных пользователем, проверки введенных данных и передачи их на web-сервер.

Форма служит для ввода пользователем через окно браузера данных и передачи их на web-сервер. Форма состоит из контейнера ***<FORM> …</FORM>*** и помещённых в него тегов (элементов) **<INPUT>, <SELECT>** *и* **<TEXTAREA>**.

Проверка данных перед отправкой на сервер

Для уменьшения нагрузки на сеть и web-сервер можно проверять введённые пользователем данные в браузере с помощью скрипта на JavaScript. Если в данных обнаружится ошибки, то пользователю предоставляется возможность их исправить. Введенные правильно данные отправляются на web-сервер. Использование сценария для управления формой демонстрируется в примере 1.

Пример 1

```
<HTML>
<HEAD><TITLE>Первая страница</title></head>
<H2>Представьтесь, пожалуйста</h2>
<FORM name=F1 METHOD="POST" ACTION="">
Имя....
<INPUT TYPE="text" NAME="name"><BR>
Возраст <INPUT TYPE="text" NAME="age">
<P> <INPUT TYPE="submit" VALUE="ВВОД" onclick="Proverka()">
</FORM> <SCRIPT>
function Proverka()
{ im=document.F1.name.value vozr=document.forms[0].elements[1].value st=""

if(im == "") st="имя\n" if(vozr == "") st+="возраст"
if(st == "") document.F1.action="Prim3_1.php" else
//отмена передачи на web-сервер
{   str="Введите:\n "+st     alert(str)
   return=false
}

}
</script> </html>

<HTML>
```

```
<HEAD><TITLE>Вторая страница</title></head>
<BODY>
<H3>П Р И В Е Т С Т В И Е</h3>
<?php
$imja=$_POST["name"]; //приём параметров из формы $age=$_POST["age"];
$x="Здравствуйте!
$imja.";
if($age>50) echo "$x Вы включены в старшую группу."; elseif($age>30)echo
"$x Вы включеныв группу среднего возраста."; else echo"$x Вы относитесь к
молодёжной группе.";
?>
<P>
<A href="Prim3_1.html">Возврат </a>
</body>
</html>
```

Пример 1 состоит из двух страниц. Первая страница служит для ввода пользователем данных и их проверки с помощью скрипта, написанного на JavaScript. Если данные введены правильно, то они отправляются на web-сервер. На web-сервере полученные данные обрабатываются скриптом, написанным на языке PHP, формируется и пересылается на браузер пользователя новая страница.

В скрипте, написанном на JavaScript, для доступа к данным в форме используются имена и индексы элементов формы. Для задания адреса (URL) страницы, содержащей PHP-скрипт, используется свойство *action* объекта *Form*.

Получение данных из всплывающего списка

Иногда можно полностью решать задачу ведения диалога с пользователем средствами JavaScript, не обращаясь к web-серверу. В примере 2 пользователь вводит код цвета в модели RGB и выбирает из списка название цвета. После нажатия кнопки ***Ввод*** на экран выводятся окрашенные в выбранные цвета код и название цвета.

Пример 2

```
<html>
<HEAD><TITLE>СКИПТ SELECT </title> </head>
<body>
<!--Пример выбора и выводана экранэлемента списка Select -->
<SCRIPT>
function select_()
```

```
{ a=document.all.Kod.value //код цвета b=
document.all.Gor.selectedIndex
//номервыбранного элемента// спискаselect c =
document.all.Gor.options[b].text//текст элементасписка d =
document.all.Gor.options[b].value//передаваемое из формы
//значение<option value= red>
e="<FONT size= 7 color="+a+">"+a+"</font>"//трансляция и
document.all.alfa.innerHTML= e// вывод на экран HTML-строки e=
"<FONT size = 7 color= "+d+">"+c+"</font>"
document.all.beta.innerHTML= e
}
</script>
<H2>Подбор оттенков цвета</h2>
В первое поле нужно ввести шестнадцатеричный код цвета.<BR>
Например,красный цвет имеет код FF0000.
<BR>Из списка во втором поле выбирается для сравнения
<BR> один из основных цветов(красный, зелёный,синий)
<P>Введитекод цвета
<input TYPE= "text"name="Kod"> <P>Выберите цвет
<SELECT NAME= "Gor">
<option value="red">Красный </option>
<option value="yellow">Жёлтый </option>
<option value="maroon">каштановый</option>
<option value=green">Зеленый</option>
<option value="blue">Синий
</select>
<P> <BUTTON onclick="select_()">Выполнить </button>
<P><B ID="alfa"></b>
<br> <B ID = "beta"></b>
</body>
</html>
```

В примере 2 нет необходимости использовать контейнер ***<FORM> …</form>***, так как на web-сервер ничего не передаётся. Для вывода на экран кода и названия цвета используется свойство *innerHTML*. Строго говоря, *innerHTML* следовало бы называть не свойством, а методом. Рассмотрим применение этого свойства на примере вывода на экран введённого пользователем кода цвета.

Пусть пользователь ввёл код красного цвета *a=FF0000*. В результате выполнения оператора e="<FONT size= 7 color="+a+">"+a+"" сформируется строка e="<FONT size=7 color=FF0000>FF0000"

В результате выполнения оператора document.all.alfa.innerHTML= e

в элемент

<B ID="alfa"> вставится

значение переменной **e**:

<B ID="alfa"><FONT size=7 color=FF0000>FF0000

Браузер выполнит преобразованный оператор языка *HTML*, т. е. выведет на экран надпись **FF0000** красного цвета.

Таким образом, с помощью свойства ***innerHTML*** можно на стороне браузера программным путём вносить изменения в HTMLдокумент (страницу сайта).

Задание 1

Создайте сайт из двух страниц.

Первая страница имеет заголовок «*Заказ мебели».*

На странице расположены два поля со списками (теги ***<SELECT>***), поле *(**<INPUT>**)* и кнопка *(**<SUBMIT>**)*. Из первого поля со списком пользователь выбирает изделие (шкаф, стол, сервант и т. д.). Из второго поля со списком пользователь выбирает материал (дуб, орех, бук). В третье поле нужно ввести количество заказываемых изделий.

После ввода данных необходимо проверить, все ли данные введены. Если обнаружена ошибка, то нужно вывести сообщение и предложить её исправить. Правильно введённые данные нужно отправить на web-сервер. Вторая страница содержит написанный на PHP скрипт, с помощью которого формируется следующее сообщение:

Ваш заказ принят

Заказано изделие – *название заказанного изделия*

Материал – *заказанный материал*

Количество – *заказанное количество*

Если в форме нужно заполнить много полей, то пользователю удобнее получать сообщения об ошибках сразу после окончания ввода данных в очередное поле, т. е. после нажатия клавиши **Tab** или клавиши со стрелкой. Для немедленной проверки введённых данных используйте событие ***onchange*:**

<INPUT TYPE ="text" SIZE=6 onchange="arg(this)">

Функция, вызываемая событием **onchange**, имеет примерно такую структуру: function arg(fld)

```
{ x=fld.value //введённое значение
if(x. . .) //условия проверки   {
alert("Сообщение об ошибке");
    fld.focus();
fld.select()
  }
}
```

Методы **focus()** и **select()** служат для возвращения курсора мышки в поле ввода и выделения ошибочных данных. Эти методы без использования специальных приёмов правильно работают только в браузере Mozilla.

Задание 2

Создайте web-страницу (рис. 2) для вычисления тригонометрических функций. Вводимые пользователем данные должны проверяться немедленно после ввода и после нажатия кнопки ***Вычислить***.

Рис. 2. Web-страница для задания 2

Не забудьте перевести градусы в радианы.

Название тригонометрической функции можно передавать как параметр тега:

<option value="sin". . . >

Сформируйте текстовую строку вида

"Math." + имя_ф + "(" + знач_аргумета + ")" // имя_ф – sin,cos или tan

Затем воспользуйтесь функцией *eval(строка),* которая выполняет выражение, хранящееся в строке.

Лабораторная работа 9. Разработка сайта на языке PHP. Создание нового пользователя приложения

Цель работы – научиться создавать на PHP функции для проверки допустимости данных и добавления их в базу данных.

В этой лабораторной работе рассматривается расширение функционала приложения добавлением функции *Create a New Wisher*.

Реализация затрагивает файл *index.php*, при этом будут созданы два новых файла с именами ***createNewWisher.php*** и ***editWishList.php***.

Данный пример состоит из трех действий:

1. Пользователь открывает файл *index.php* титульной страницы и щелкает ссылку для регистрации.
2. Пользователь переходит на страницу *createNewWisher.php* для создания нового автора желания.
3. После создания нового автора желания пользователь переключается на *editWishList.php*, где для него можно создать список желаний.

Добавление ссылки для начала создания нового автора желания

Откройте файл index.php. Добавьте пустую строку под закрывающим тегом **</*form*>**. Введите в эту пустую строку следующий блок кода:

```
<br>Still don't have a wish list?!
<a href="createNewWisher.php">Create now</a>
```

Создание новых web-страниц PHP

Создайте в исходных файлах проекта две новые web-страницы PHP: ***createNewWisher.php*** и ***editWishList.php***.

В ***editWishList.php*** добавьте текст *Hello!* к тексту в формате HTML, а в остальном оставьте всё как было. Этот файл необходим в качестве объекта ссылки для ***createNewWisher.php.***

Добавление формы HTML для ввода данных нового автора желания

Введите или вставьте следующий блок *HTML* в строку ***createNewWisher.php*** под блоком *PHP*:

```
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title></title>
   </head>
   <body>
     Welcome!<br>
  <form action="createNewWisher.php" method="POST">
Your name: <input type="text" name="user"/><br/>
         Password: <input type="password" name="password"/><br/> Please confirm your password: <input type="password" name="password2"/>
<br/>          <input type="submit" value="Register"/>        </form>
    </body>
</html>
```

Тип ***password*** – это специальный тип текстового поля, в котором символы заменяются звездочками. Код представляет собой форму HTML, позволяющую ввести имя и пароль нового автора пожелания в текстовые поля. При нажатии кнопки *Register* введенные данные передаются для проверки допустимости на ту же страницу – *createNewWisher.php*. Предупреждения от средства проверки *HTML* можно проигнорировать.

Проверка допустимости данных и добавление их в базу данных

В этом разделе мы добавим код PHP к *createNewWisher.php*. Добавьте этот код к блоку *PHP* на верху файла. Блок *PHP* должен находиться над кодом *HTML* ***all***, пустыми строками или пробелами. Расположение блока кода *PHP* является важным для правильного функционирования оператора переадресации. Внутри блока *PHP* введите или вставьте в указанном порядке блоки кода, описанные ниже в данном разделе.

Необходимо инициализировать переменные. Первая группа переменных осуществляет передачу параметров доступа к базе данных, а другая группа переменных используется в работе кода PHP.

Добавьте следующий код для проверки допустимости данных:

```
3. /** database connection credentials */
   $dbHost="localhost"; //on MySql
```

4. $dbXeHost="localhost/XE";
   $dbUsername="phpuser";
   $dbPassword="phpuserpw";

   /** other variables */
   $userNameIsUnique = true;
   $passwordIsValid = true;
   $userIsEmpty = false;
   $passwordIsEmpty = false;
   $password2IsEmpty = false;

Под переменными следует добавить блок ***if***. Параметр блока ***if*** выполняет проверку того, что страница была запрошена из нее самой посредством метода POST. Если это не так, дальнейшие проверки допустимости не выполняются, и на экран выводится страница с пустыми полями, как описано выше.

/** Check that the page was requested from itself via the POST method. */ if ($_SERVER["REQUEST_METHOD"] == "POST")
{

Внутри фигурных скобок блока ***if*** добавьте другой блок ***if*** , позволяющий проверить, ввел ли пользователь имя автора желания в поле. Если текстовое поле ***user*** является пустым, значение ***$userIsEmpty*** меняется на ***true***.

/** Check that the page was requested from itself via the POST method. */ if ($_SERVER["REQUEST_METHOD"] == "POST") {

** Check whether the user has filled in the wisher's name in the text field "user" */ if
($_POST["user"]=="")
{
$userIsEmpty = true;
}
}

Добавьте код, устанавливающий подключение к базе данных. Если установить подключение невозможно, то выводится ошибка MySQL:

/** Check that the page was requested from itself via the POST method. */

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
/** Check whether the user has filled in the wisher's name in the text field
"user" */       if ($_POST["user"]=="") {
       $userIsEmpty = true;
   }

/** Create database connection */

   $con = mysqli_connect($dbHost, $dbUsername, $dbPassword); if (!$con)
 {       exit('Connect Error (' . mysqli_connect_errno() . ')
 'mysqli_connect_error());
   }
//set the default client character set     mysqli_set_charset($con, 'utf8');
}
```

Добавьте код, позволяющий проверить, существует ли пользователь, имя которого соответствует указанному в поле ***user.*** Эта задача выполняется путем поиска идентификационного номера автора пожелания в соответствии с именем, указанным в поле ***user***. Если такой номер существует, значение ***$userNameIsUnique*** меняется на ***false***.

```
/** Check that the page was requested from itself via the POST method.
*/ if ($_SERVER["REQUEST_METHOD"] == "POST") {

/** Check whether the user has filled in the wisher's name in the text field "user"
*/

   if ($_POST["user"]=="")
{
       $userIsEmpty = true;
   }
/** Create database connection */
   $con = mysqli_connect($dbHost, $dbUsername, $dbPassword);     if
(!$con)
  { exit('Connect Error (' . mysqli_connect_errno() . ') '                .
mysqli_connect_error());
   }
  /**set the default client character set */     mysqli_set_charset($con, 'utf-
8');
  /** Check whether a user whose name matches the "user" field already exists */
```

```
   mysqli_select_db($con, "wishlist");
   $user = mysqli_real_escape_string($con, $_POST["user"]);
   $wisher = mysqli_query($con, "SELECT id FROM wishers WHERE
name='".$user."'");
   $wisherIDnum=mysqli_num_rows($wisher);
   if ($wisherIDnum) {
      $userNameIsUnique = false;
   }
}
```

После кода, проверяющего уникальность пользователя, добавьте серию блоков ***if*** , проверяющих, правильно ли пользователь ввел и подтвердил пароль. Код выполняет проверку того, что поля *Password* (переменная ***password***) и *Confirm Password* (***переменная password2***) заполнены и идентичны друг другу. В противном случае значения соответствующих логических переменных также изменяются:

```
if ($_POST["password"]=="") {
$passwordIsEmpty = true;
}
if ($_POST["password2"]=="")  {
$password2IsEmpty = true;
}
if ($_POST["password"]!=$_POST["password2"]) {    $passwordIsValid = false;
}
```

Завершите блок if ($_SERVER['REQUEST_METHOD'] == *POST*), добавив код, вставляющий новую запись в базу данных *Wishers*:

```
/** Check that the page was requested from itself via the POST method. */
if ($_SERVER['REQUEST_METHOD'] == "POST") {
   /** Check whether the user has filled in the wisher's name in the text field
"user" */
   if ($_POST['user'] == "") {
      $userIsEmpty = true;
   }

   /** Create database connection */
   $con = mysqli_connect($dbHost, $dbUsername, $dbPassword);     if
(!$con) {
      exit('Connect Error (' . mysqli_connect_errno() . ') '
            . mysqli_connect_error());
```

```
    }
    //set the default client character set     mysqli_set_charset($con, 'utf-8');

    /** Check whether a user whose name matches the "user" field already exists */
    mysqli_select_db($con, "wishlist");
    $user = mysqli_real_escape_string($con, $_POST['user']);
    $wisher = mysqli_query($con, "SELECT id FROM wishers WHERE
name='".$user."'");
    $wisherIDnum=mysqli_num_rows($wisher);
    if ($wisherIDnum) {
        $userNameIsUnique = false;
    }

    /** Check whether a password was entered and confirmed correctly
*/
    if ($_POST['password'] == "") {
        $passwordIsEmpty = true;
    }
    if ($_POST['password2'] == "") {
        $password2IsEmpty = true;
    }
    if ($_POST['password'] != $_POST['password2']) {
        $passwordIsValid = false;
    }
    /** Check whether the boolean values show that the input data was validated
successfully.
*       If the data was validated successfully, add it as a new entry in the "wishers"
database.
*       After adding the new entry, close the connection and redirect the application
to editWishList.php.
     */
    if (!$userIsEmpty && $userNameIsUnique && !$passwordIsEmpty
&& !$password2IsEmpty && $passwordIsValid) {
        $password = mysqli_real_escape_string($con, $_POST['password']);
        mysqli_select_db($con, "wishlist");
        mysqli_query($con, "INSERT wishers (name, password) VALUES
('" . $user . "', '" . $password . "')");
mysqli_free_result($wisher);        mysqli_close($con);
        header('Location: editWishList.php');        exit;
    }
```

```
}
```

В соответствии с кодом выполняется проверка того, что указано уникальное имя пользователя и что пароль введен и подтвержден правильно. Если эти условия выполнены, код извлекает значения *user* и *password* из формы HTML и вставляет их соответственно в столбцы ***Name*** и ***Password***, относящиеся к новой строке в базе данных *Wishers*. После создания строки код закрывает подключение к базе данных и переадресует приложение на страницу *editWishList.php*.

Отображение сообщений об ошибках в форме ввода

Перейдем к реализации вывода сообщений об ошибках при неверно введенных данных. Реализация основывается на проверках допустимости и изменении значений логических переменных.

Введите следующий блок кода PHP в форме ввода HTML непосредственно под именем пользователя:

```
Welcome!<br>
        <form action="createNewWisher.php" method="POST">
           Your name: <input type="text" name="user"/><br/>
<?php
   if ($userIsEmpty) { echo ("Enter your
        name, please!"); echo ("<br/>");
        }
        if (!$userNameIsUnique) {
        echo ("The person already exists. Please check the spelling and try again");
        echo ("<br/>");
        }
        ?>
```

Введите следующий блок кода PHP в форме ввода HTML под кодом для ввода пароля:

```
Password: <input type="password" name="password"/><br/>


   <?php
     if ($passwordIsEmpty) { echo ("Enter the
          password, please!"); echo ("<br/>");
          }
          ?>
```

Введите следующий блок кода PHP в форме ввода HTML под кодом для подтверждения пароля:

```
Please confirm your password: <input type="password"
name="password2"/><br/>
<?php
        if ($password2IsEmpty) {
        echo ("Confirm your password, please");
        echo ("<br/>");
        }
        if (!$password2IsEmpty && !$passwordIsValid) { echo
        ("The passwords do not match!");
        echo ("<br/>");
        }
        ?>
```

Тестирование функциональных возможностей по созданию нового пользователя Create New Wisher

Запустите приложение. Откроется страница-указатель.

На странице-указателе щелкните ссылку рядом с текстом *Still don't have a wish list?* Откроется форма.

Оставьте поля пустыми и нажмите кнопку *Register* («Зарегистрировать»). На экране появится сообщение об ошибке.

Введите имя зарегистрированного пользователя, например, *Tom* в поле *Your name*, внимательно заполните другие поля и нажмите кнопку *Register*. На экране появится сообщение об ошибке.

Заполните поля *Password* и *Please confirm your password* различными значениями и нажмите кнопку *Register*. На экране появится сообщение об ошибке.

Введите *Bob* в поле *Your name*, укажите в полях пароля один и тот же пароль и нажмите кнопку *Register*. Откроется пустая страница, однако переадресация осуществляется правильно, поскольку URL-адрес заканчивается текстом *editWishList.php*:

Проверьте, что данные сохранены в базе данных, путем перехода к разделу *Wishers* в окне *Services*, расположенном под узлом *wislist1*, и выбора *View Data* в контекстном меню.

Лабораторная работа 10. Разработка сайта на языке PHP. Оптимизация кода путем добавления классов и объектов

Цель работы – научиться оптимизировать программу на PHP путем *Создания классов*.

В этой лабораторной работе рассматриваются способы оптимизации кода, позволяющие упростить поддержку кода в дальнейшем. Данная процедура затрагивает файлы ***createNewWisher.php*** и ***wishlist.php***. Кроме того, создается новый файл под названием ***db.php***.

Код приложения содержит несколько похожих блоков с запросами к базе данных. Для упрощения чтения и поддержки кода в будущем можно извлечь эти блоки, реализовать их в качестве функций отдельного класса ***WishDB*** и поместить текст ***WishDB*** в файл ***db.php***. Впоследствии можно будет включить файл ***db.php*** в любой файл PHP и использовать любую функцию класса ***WishDB*** без дублирования кода. Такой подход гарантирует, что любые изменения в запросах или функциях будут выполнены в одном местоположении, и анализировать весь код приложения не потребуется.

При использовании функции класса ***WishDB*** не следует изменять значения каких-либо переменных в классе ***WishDB***. Вместо этого необходимо использовать класс в качестве концептуального проекта для создания объекта ***WishDB*** и изменять значения переменных в этом объекте. Объект уничтожается при завершении работы. Поскольку значения непосредственно класса ***WishDB*** никогда не изменяются, данный класс можно использовать повторно неограниченное число раз. В некоторых случаях может потребоваться одновременно несколько экземпляров класса, а в других случаях будет предпочтителен «одноэкземплярный» класс, имеющий только один экземпляр в любой момент времени. ***WishDB*** в данной лабораторной работе представлен как одноэкземплярный класс.

Следует отметить, что создание объекта класса обозначается термином «создание экземпляра» этого класса и что объект в данном случае называется экземпляром класса. Общий термин, обозначающий программирование с использованием классов и объектов, – «объектно ориентированное программирование» (ООП). Ваша цель – переместить функциональность вызова базы данных из отдельных файлов PHP в класс ***WishDB***. Это соответствует новой объектно ориентированной структуре приложения.

Создание файла db.php

Создайте новую подпапку в папке «Исходные файлы». Дайте папке имя ***Includes***.

Создайте новый файл под именем ***db.php*** и поместите его в папку Includes. Позже вы сможете добавлять в эту папку файлы, которые будут включаться в другие PHP-файлы.

Чтобы создать файл *db.php* в новой папке, сделайте следующее:

1. Щелкните правой кнопкой мыши узел *Source Files* и выберите *New > Folder* в контекстном меню. Откроется диалоговое окно «Новая папка».
2. В поле «Имя папки» введите *Includes*. Затем нажмите кнопку «Готово».
3. Щелкните правой кнопкой мыши узел *Includes* и выберите *New > PHP File* в контекстном меню. Откроется диалоговое окно «Новый файл PHP».
4. В поле «Имя файла» введите ***db***. Затем нажмите кнопку «Готово».

Создание класса WishDB

Для создания класса ***WishDB*** необходимо инициализировать переменные класса и реализовать конструктор класса. Пользователи MySQL, обратите внимание, что класс ***WishDB*** *расширяет mysqli*. Это означает, что класс ***WishDB*** *наследует* функции и другие характеристики класса PHP ***mysqli***. Вы убедитесь в важности этого при добавлении функций ***mysqli*** к классу.

Откройте файл ***db.php*** и создайте класс ***WishDB***. В данном классе объявите переменные настройки базы данных для хранения имени и пароля собственника базы данных (пользователя), имени и машины размещения базы данных. Все объявления переменных являются закрытыми: это означает, что начальные значения в этих объявлениях недоступны вне класса WishDB. Вы также объявляете закрытую *статическую переменную* ***$instance***, которая хранит экземпляр ***WishDB***. Ключевое слово «статический» означает, что функции в классе имеют доступ к переменной даже при отсутствии экземпляра класса.

Наберите следующий код:

```
class WishDB extends mysqli {

    // single instance of self shared among all instances     private static
    $instance = null;
    // db connection config vars     private
    $user = "phpuser";     private $pass =
    "phpuserpw";     private $dbName =
    "wishlist";
```

```
        private $dbHost = "localhost";
}
```

Создание экземпляров класса WishDB

При использовании функций класса **WishDB** в других файлах PHP должна быть вызвана функция, позволяющая создать объект («создать экземпляр») класса **WishDB**. **WishDB** разработан в качестве *одноэкземплярного класса*; это означает, что в любой определенный момент времени может существовать только один экземпляр класса. Поэтому рекомендуется предотвращать создание экземпляра WishDB, которое осуществляется извне и способствует появлению дублирующихся экземпляров.

Внутри класса WishDB введите следующий код:

```
//This method must be static, and must return an instance of the object if the object
//does not already exist.
public static function getInstance() {    if
(!self::$instance instanceof self) {
self::$instance = new self;
  }
  return self::$instance; }

// The clone and wakeup methods prevents external instantiation of copies of the
Singleton class,
// thus eliminating the possibility of duplicate objects. public
function __clone() {
  trigger_error('Clone is not allowed.', E_USER_ERROR);
}
public function __wakeup() {
  trigger_error('Deserializing is not allowed.', E_USER_ERROR);
}
```

Функция ***getInstance*** является общедоступной и статической. Общедоступность означает возможность свободного доступа извне класса. Статическая функция доступна даже в том случае, если для класса не было создано экземпляров. Поскольку функция getInstance вызывается для создания экземпляров класса, она является статической. Обратите внимание, что эта функция имеет доступ к статической переменной $instance и устанавливает ее значение как экземпляр класса.

Двойное двоеточие (::), или «оператор разрешения диапазона» (*Scope Resolution Operator*), и ключевое слово ***self*** используются для получения доступа к статическим функциям. ***Self*** в рамках определения класса

используется в качестве ссылки на данный класс. Если двойное двоеточие находится вне определения класса, вместо ***self*** используется имя класса.

Добавление конструктора к классу WishDB

Класс может содержать в себе специальный метод, известный как «конструктор», который выполняется автоматически каждый раз при создании экземпляра этого класса. Рассматрим добавление к классу ***WishDB*** конструктора, который подключается к базе данных каждый раз при создании экземпляра ***WishDB***. Добавьте к ***WishDB*** следующий код:

```
// private constructor private
function __construct()
{    parent::__construct($this->dbHost, $this->user, $this->pass, $this-
>dbName);
   if (mysqli_connect_error()) {
      exit('Connect Error (' . mysqli_connect_errno() . ') '
            . mysqli_connect_error());
                                    }                          parent::set_charset('utf-8');
}
```

Следует учитывать, что вместо переменных $con, $dbHost, $user или $pass используется псевдопеременная $this. Псевдопеременная $this используется при вызове метода внутри контекста объекта. Она ссылается на значение переменной внутри этого объекта.

Функции класса WishDB

Рассматрим реализацию следующих функций класса ***WishDB***:

– *get_wisher_id_by_name* – для извлечения идентификатора пользователя на основе имени;

– *get_wishes_by_wisher_id* – для извлечения списка пожеланий *Wish list*, принадлежащего определенному пользователю с соответствующим идентификатором;

– create_wisher – для добавления нового пользователя в таблицу *Wishers*.

Функция get_wisher_id_by_name

Эта функция возвращает идентификатор пользователя, а в качестве входного параметра для ее выполнения требуется имя пользователя.

После функции WishDB введите следующую функцию в класс WishDB:

```
public function get_wisher_id_by_name($name) {
```

```
   $name = $this->real_escape_string($name);

   $wisher = $this->query("SELECT id FROM wishers WHERE name =
'"

          . $name . "'");
   if ($wisher->num_rows > 0){
$row = $wisher->fetch_row();
      return $row[0];
   } else
      return null; }
```

Блок кода выполняет запрос SELECT ID FROM wishers WHERE name = [переменная для имени пожелания]. Результат запроса – массив идентификаторов из записей, соответствующих запросу. Если массив не пустой, это по умолчанию означает, что он содержит один элемент, поскольку при создании таблицы имя поля было определено как UNIQUE. В этом случае функция возвращает первый элемент массива *$result* (элемент под номером ноль). Если массив пуст, функция возвращает значение *null*.

Для базы данных MySQL строка ***$name*** используется с с escape-символом для предотвращения атак SQL-инъекций. Рекомендуется исключить из участия в запросах MySQL такие строки, которые могли бы быть подвержены подобной атаке.

Функция get_wishes_by_wisher_id

Эта функция возвращает зарегистрированные пожелания пользователя, и для ее выполнения в качестве входного параметра требуется идентификатор пользователя. Введите следующий блок кода:

```
public function get_wishes_by_wisher_id($wisherID) {
   return $this->query("SELECT id, description, due_date FROM wishes
WHERE wisher_id=" . $wisherID);
}
```

Блок кода выполняет запрос

"SELECT id, description, due_date FROM wishes WHERE wisherID=" . $wisherID

и возвращает набор результатов, который является массивом записей, соответствующих запросу.

Функция create_wisher

Функция создает новую запись в таблице *Wishers*. Эта функция не возвращает каких-либо данных, и в качестве входных параметров для ее выполнения требуется имя и пароль нового пользователя. Введите следующий блок кода:

```
public function create_wisher ($name, $password){
   $name = $this->real_escape_string($name);
   $password = $this->real_escape_string($password);
   $this->query("INSERT INTO wishers (name, password)
VALUES ('" . $name . "', '" . $password . "')");
}
```

Блок кода выполняет запрос INSERT *wishers* (*Name*, *Password*) VALUES ([переменные представляющие имя и пароль нового пожелания]). При выполнении запроса добавляется новая запись в таблицу *Wishers* с полями *name* и *password*, заполненными значениями *$name* и *$password* соответственно.

Реорганизация кода приложения

Теперь при наличии отдельного класса для работы с базой данных дублированные блоки можно заменить вызовами соответствующих функций из этого класса. Это помогает в дальнейшем избежать ошибок и противоречий в написании кода. Усовершенствование кода, не оказывающее влияния на функциональные возможности, называется реорганизацией.

Начнем с файла *wishlist.php*, поскольку он небольшой и дает возможность представить оптимизацию более иллюстративно.

1.В верхней части блока <? php? > введите следующую строку, делающую возможным использование файла db.php:

```
require_once("Includes/db.php");
```

*2.*Замените код, который подключается к базе данных и получает идентификатор пожелания, вызовом функции *get_wisher_id_by_name.*

Новый код сначала вызывает функцию *getInstance* в *WishDB*. Функция *getInstance* возвращает экземпляр W*ishDB*, а код вызывает функцию *get_wisher_id_by_name* в пределах данного экземпляра. Если требуемое пожелание в базе данных не найдено, код завершает процесс и отображает сообщение об ошибке.

Для открытия подключения к базе данных наличие кода не является необходимым. Открытие подключения выполняется конструктором класса *WishDB*. Если имя и/или пароль изменяются, необходимо обновить только соответствующие переменные класса *WishDB*.

3.Замените код, который получает пожелания для автора пожеланий, идентифицированного с помощью кода, кодом, который вызывает функцию *get_wishes_by_wisher_id*.

4.Удалите строку, которая закрывает подключение к базе данных.

Код не нужен, потому что подключение к базе данных автоматически закрывается при уничтожении объекта *WishDB*. Однако рекомендуем сохранять код, освобождающий ресурс. Вам необходимо освободить все ресурсы, которые используют подключение, чтобы убедиться в том, что оно закрыто, даже при вызове функции *close* или уничтожении экземпляра с подключением к базе данных.

Реорганизация файла createNewWisher.php не оказывает воздействия на форму ввода HTML или код для вывода на экран соответствующих сообщений об ошибках.

1. В верхней части блока *<? php? >* введите следующий код, делающий возможным использование файла *db.php*:

```
require_once("Includes/db.php");
```

2. Удалите подтверждения подключения к базе данных ($dbHost и пр.). Теперь они находятся вdb.php.

3. Замените код, который подключается к базе данных и получает идентификатор пожелания, вызовом функции get_wisher_id_by_name.

Объект ***WishDB*** существует до тех пор, пока обрабатывается текущая страница. Если обработка завершена или прервана, этот объект уничтожается. Код для открытия подключения к базе данных не является необходимым, поскольку подключение выполняется посредством функции *WishDB*. Код для закрытия подключения также не является необходимым, поскольку подключение будет закрыто сразу же после уничтожения объекта ***WishDB***.

4. Замените код, который вставляет новых авторов пожеланий в базу данных, кодом, который вызывает функцию *create_wisher*.

ЗАКЛЮЧЕНИЕ

Вы закончили изучение основ web-программирования по нашему учебному пособию. Надеемся, что не только изучили теоретический материал, но и выполнили цикл лабораторных работ, следовательно, теперь имеете представление об общих принципах построения интернет-приложений. Если решили серьезно заниматься интернет-разработкой, то для вас это только начало пути.

В рамках одного учебного пособия невозможно продемонстрировать весь арсенал современных средств для проектирования и создания интернет-приложений на стороне клиента и на стороне сервера.

Вы должны понимать, что языки и технологии web-программирования постоянно развиваются и меняются, поэтому придется всё время продолжать осваивать их самостоятельно. Существует огромное количество профессиональных сайтов и форумов, на которых размещены справочные материалы по различным языкам webпрограммирования и технологиям разработки web-приложений, которыми активно пользуются и студенты, и профессиональные разработчики при решении конкретных задач.

Мы желаем Вам успехов в создании собственных интернетприложений!

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шабашов, В. Я. Организация доступа к данным из PHP-приложений для различных СУБД: учеб. пособие по дисциплине «Webпрограммирование» / В. Я. Шабашов. – М.; Берлин: Директ-Медиа, 2019. – 121 с.

2. Основы работы в Web-среде: лабораторный практикум / авт.-сост. С. В. Говорова; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». – Ставрополь: СКФУ, 2017. – 160 с.

3.Аникина Е.И. ПРОФЕССИОНАЛЬНАЯ ЭТИКА IT-СПЕЦИАЛИСТОВ.-Курск.- 2018.- С.176

4. Аникина Е.И. Компьютерная и информационная этика.- Saarbrücken.- 2015.-С. 68.

5. Аникина Е.И. Информационные технологии: этические аспекты.- Saarbrücken.- 2016.-С. 220.

6. Anikina E.I. DEVELOPMENT OF DATABASE DRIVEN WEB-APPLICATION.-Saarbrücken, 2017.-С.152.

7. Аникина Е.И. МОБИЛЬНЫЕ ТЕХНОЛОГИИ BYOD: ТЕНДЕНЦИИ РАЗВИТИЯ И ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ В ВЫСШЕМ ОБРАЗОВАНИИ// Известия Юго-Западного
государственного университета. Серия: Лингвистика и педагогика.2019.-Т.9.-№3.- С.132-141.

8. Аникина Е.И., Бабков А.С., Малышев А.В.АВТОМАТИЗАЦИЯ ФУНКЦИЙ ДЕКАНАТА В ЭЛЕКТРОННОЙ ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНОЙ СРЕДЕ ЮЗГУ//
Известия Юго-Западного государственного университета.-2017.-№ 6 (75).-С.44-50.

9. Диков, А. В. Веб-технологии HTML и CSS: учеб. пособие / А. В. Диков. – 2-е изд. – М.: Директ-Медиа, 2012. – 78 с.

Редактор *С. П. Тарасова*