МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)**

**ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО**

**ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ**

**ПИ (филиал) ДГТУ в г. Таганроге**

УТВЕРЖДАЮ

Директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.К. Исаев

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г

Рег. № \_\_\_\_\_\_\_\_\_\_\_\_

**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ**

к практическим занятиям

по учебной дисциплине ОП.19 Web-программирование

 по специальности 09.02.07 «Информационные системы и программирование»

Таганрог

2020

**Лист согласования**

Учебно-методическое пособие по учебные дисциплины Web-программирование разработана на основе Федерального государственного образовательного стандарта (далее – ФГОС) для специальности среднего профессионального образования (далее – СПО) 09.02.07 «Информационные системы и программирование».

**Разработчик(и):**

Преподаватель Т.М. Марданова

Методические указания рассмотрены и одобрены на заседании цикловой методической комиссии специальности 09.02.05Прикладная информатика (по отраслям)

Протокол № 7 от «04» февраля 2020г

Председатель цикловой методической комиссии О.В. Андриян

**Рецензенты:**

ЧОУ ВО «ТИУиЭ» начальник информационно-аналитического управления, к.т.н., доцент О.И. Овчаренко

АО «Красный гидропресс»зам. начальника отдела ИТ С.С. Пирожков

**Согласовано:**

Заведующий УМО

Т. В. Воловская

**Введение**

В учебно-методическом пособии к практикуму по курсу «Web-программирование» изложены сведения, необходимые для успешного выполнения практических занятий по данному курсу. Описан процесс работы с инструментарием, применяемым на практических занятиях, представлен ряд типичных задач и подходы к их решению. Практические занятия посвящены углубленному знакомству обучающихся с умением разрабатывать программный код клиентской и серверной части веб-приложений и проектированию информационную систему.

Цель настоящего пособия – помочь обучающимся при выполнении практических работ, выполняемых для закрепления знаний по теоретическим основам и получения практических навыков работы.

Обучающийся должен знать:

* языки программирования и разметки для разработки клиентской и серверной части веб-приложений;
* принципы проектирования и разработки информационных систем

Обучающийся должен уметь:

* разрабатывать программный код клиентской и серверной части веб-приложений;
* разрабатывать и проектировать информационные системы

 Данное учебно-методическое пособие предназначено для обучающихся 2 курса.

**Правила выполнения практических занятий**

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний.

Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет).

Обучающийся должен на уровне понимания и воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию.

Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

**Практическое занятие №1 Создать HTML-документ со вставками сценариев на РНР на обработку переменных различных типов**

**Цель:**научиться обрабатывать формы на Web-страницах с помощью сценариев на языке PHP.

**Задания для выполнения:**

1. Реализовать на практике пример обработчика формы для ввода логина и пароля.

2. Разработать сценарий для вывода приветствия пользователя, который вводит свое имя с помощью формы. Для передачи данных использовать метод GET.

3. Разработать сценарий вычисления суммы, разности, произведения, частного и остатка для введенных двух чисел. Проверить его выполнения. Для передачи данных использовать метод POST.

4. Разработать сценарий для проведения тестирования (4-5 вопросов) и вывод результатов анкетирования пользователя.

**Выполнение работы**

**Обработка запросов с помощью PHP**

Внутри PHP-скрипта существует несколько способов получения доступа к данным, переданных клиентом по протоколу HTTP. До версии PHP 4.1.0 доступ к таким данным осуществлялся по именам переданных переменных. Таким образом, если, например, было передано first\_name = Nina, то внутри скрипта появлялась переменная $ first\_name со значением Nina. Если нужно было различать, каким методом были переданы данные, то использовались ассоциативные массивы

\ $ HTTP\_POST\_VARS и \ $ HTTP\_GET\_VARS, ключами которых были имена переданных переменных, а значениями - соответственно значения этих переменных. Таким образом, если пара first\_name = Nina передана методом GET, то $HTTP\_GET\_VARS["first\_name"]="Nina"".

Использовать в программе имена переданных переменных непосредственно опасно. Поэтому было решено начиная с PHP 4.1.0 задействовать для обращения к переменным, переданных с помощью HTTP-запросов, специальный массив - $ \_REQUEST. Этот массив содержит данные, переданные методами POST и GET, а также с помощью HTTP cookies. Это глобальный ассоциативный массив, то есть его значений можно приобрести в любом месте программы, используя в качестве ключа имя соответственной переменной (элемента формы).

**Пример**

|  |
| --- |
|  |
| Рис. 11.2. Пример внешнего вида формы |

Созданная форма для регистрации участников заочной школы программирования. Тогда в файле action.php, что обрабатывает эту форму, можно написать следующее:

".$\_REQUEST["first\_name"]. "

".$\_REQUEST["last\_name"]."!
";

$str .="Вы вибрали для изучения курс по ".

$\_REQUEST["kurs"];

echo $str; ?

Тогда, если в форму мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, на экране браузера получим следующее сообщение:

Здравствуйте, Вася Петров!

Вы выбрали для изучения курс по PHP

После введения массива $ \_REQUEST массивы $ HTTP\_POST\_VARS и $ HTTP\_GET\_VARS для однородности были переименованы в $ \_POST и $ \_GET соответственно, но сами они из употребления не исчезли из соображений совместимости с предыдущими версиями PHP. В отличие от своих предшественников, массивы $ \_POST и $ \_GET стали суперглобальные, то есть доступными непосредственно и внутри функций и методов.

Приведем пример использования этих массивов. Допустим, нам нужно обработать форму, содержащую элементы ввода с именами first\_name, last\_name, kurs (например, форму, приведенную выше). Данные были переданы методом POST, и данные, передаваемые другими методами, мы обрабатывать не хотим. Это можно сделать следующим образом:

".$\_POST ["first\_name"]."

".$\_POST ["last\_name"] ."!
";

$str .= "Вы выбрали для изучения курс по ".

$\_POST["kurs"];

echo $str; ?

Тогда на экране браузера, если мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, увидим сообщение, как в предыдущем примере:

Здравствуйте, Вася Петров!

Вы выбрали для изучения курс по PHP

Для того, чтобы сохранить возможность обработки скриптов предыдущих версий, чем PHP 4.1.0, была введена директива register\_globals, что разрешает или запрещает доступ к переменным непосредственно по их именам. Если в файле настроек PHP параметр register\_globals = On, то к переменным, переданным сервера методами GET и POST, можно обращаться просто по их именам (можно писать $ first\_name). Если же register\_globals = Off, то нужно писать

$\_REQUEST["first\_name"]

або

$\_POST["first\_name"],

$\_GET["first\_name"],

$HTTP\_POST\_VARS["first\_name"],

$HTTP\_GET\_VARS["first\_name"].

С точки зрения безопасности эту директиву лучше отключать (register\_globals = Off). При включенной директиве register\_globals перечисленные выше массивы также будут содержать данные, переданные клиентом.

Иногда возникает необходимость узнать значение какой-либо переменной окружения, например метод, использовавшийся при передаче запроса или IP-адрес компьютера, отправившего запрос. Получить такую информацию можно с помощью функции getenv (). Она возвращает значение переменной окружения, имя которой передано ей в качестве параметра.

// возвратит использованный метод

echo getenv ("REMOTE\_ADDR");

// выведет IP-адрес пользователя, который послал запрос

?

Как мы уже говорили, если используется метод GET, то данные передаются добавлением строка запроса в виде пар «имя\_переменной = значение URL-адреса ресурса». Все, что записано в URL после знака, можно получить с помощью команды

getenv("QUERY\_STRING");

Благодаря этому можно по методу GET передавать данные в каком-либо другом виде. Например, указывать только значения нескольких параметров через знак плюс, а в скрипте разбирать строку запроса на части или можно передавать значение всего одного параметра. В этом случае в массиве $ \_GET появится пустой элемент с ключом, равным этому значению (всей строке запроса), причем символ «+», что встретился в строке запроса, будет заменен на подчеркивание «\_».

Методом POST данные передаются только с помощью форм, и пользователь (клиент) не видит, какие именно данные отправляются серверу. Чтобы их увидеть, хакер должен подменить нашу форму своей. Тогда сервер отправит результаты обработки неправильной формы не туда, куда нужно. Чтобы этого избежать, можно проверять адрес страницы, с которой были посланы данные. Это можно сделать опять же с помощью функции getenv ():

getenv("HTTP\_REFERER");

**Пример обработки запроса с помощью PHP**

Нужно написать обработчик формы для ввода логина и пароля и вывода их на экран.

Форма должна иметь следующий вид:


После ввода данных и нажатия кнопки «ОК» на странице должна появиться следующая информация:

В случае, когда форма и обработчик находятся в одном файле, код страницы будет следующим:

В случае, когда форма и обработчик находятся в разных файлах, то страницы и их код будут выглядеть следующим образом:

**Практическое занятие №2**

**Создание сценариев с применением констант.**

По сути своей любой PHP-скрипт ‑ это обычный текстовый файл, который, помимо простого текста (или HTML-кода), содержит в себе команды PHP.

Когда пользователь обращается к такому скрипту, запрос передаётся серверу, который действует следующим образом: всё, что не является PHP-кодом, просто передаётся пользователю без изменений, но как только обработчик встречает метку начала PHP-кода, он выполняет этот код и «отдаёт» пользователю результат обработки.

PHP-код может находиться в любой части документа и может быть включен в документ одним из двух способов:

|  |  |  |
| --- | --- | --- |
| **<?php****PHP-код****?>** | или | **<?****PHP-код****?>** |

Основные правила написания программы на РНР:

* в именах переменных и функций большие и малые буквы *различаются*;
* в конце каждого оператора (команды) указывается точка с запятой (;);
* специального символа переноса оператора на другую строчку нет – переносить можно с любой позиции в которой можно указать пробел (слова не разрываются при переносе оператора).

**ЗАДАЧА №1**

Создайте php-скрипт **lab1-5.php**, в котором вычисляется значение заданного выражения для случайных аргументов из интервала [-20, 20], а затем выводится на экран сама формула (с заменой имен переменных на их значения) и результат вычислений. Протестируйте работу скрипта и объясните полученный результат.

Распределение заданий по вариантам:

| ***Вариант*** | ***Функция*** |  | ***Вариант*** | ***Функция*** |  | ***Вариант*** | ***Функция*** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1** |  |  | **6** |  |  | **11** |  |
| **2** |  |  | **7** |  |  | **12** |  |
| **3** |  |  | **8** |  |  | **13** |  |
| **4** |  |  | **9** |  |  | **14** |  |
| **5** |  |  | **10** |  |  | **15** |  |

**Практическое занятие № 3 Создание сценариев с применением условных конструкций.**

**ЦЕЛЬ РАБОТЫ:** Научится использовать циклические операторы

Используя циклы while, do - while и for, написать три варианта программы получения на экране таблицы синусов для значений аргумента в диапазоне от 0 до π/2 с заданным числом шагов.

Вычислить и вывести все члены числового ряда

значение которых превышает 10 в -5.

Напечатать в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр.

##  **Порядок выполнения работы:**

По предложенной литературе изучить необходимый материал;

Выполнить задания практического занятия и записать листинг программы;

Дать ответы на контрольные вопросы;

 **СОДЕРЖАНИЕ ОТЧЕТА**:

Наименование и цель работы

Код программы

Записать результат (листинг) выполненых заданий.

Ответы на контрольные вопросы

Вывод о проделанной работе

**Практическое занятие № 4 Создание сценариев с применением циклов.**

**Цель**: Научиться применять операторы циклов в сценариях JavaScript

 Создайте документ HTML с заголовком «Сюрприз». Фон документа должен быть белым. Потом с помощью JavaScript досчитайте до 5, на этом этапе цвет фона меняется на желтый и появляется текст: «Скоро будет еще один цветной сюрприз…». Снова досчитайте до 5, и измените цвет фона.

Запросите у пользователя запрос: «Сколько раз пожелать вам доброго дня?». Ограничимся предельным значением – 25. Пожелайте доброго дня столько количество раз, сколько запросил пользователь.

 **Порядок выполнения работы:**

 Для выполнения первого задания воспользуйтесь оператором **for**.

 Для выполнения второго задания воспользуйтесь оператором **while**.

 **Содержание отчета:**

Наименование и цель работы

Задание, под каждым поместить код программы и полученные результаты в виде скринов.

Ответы на контрольные вопросы

Вывод о проделанной работе

**Практическое занятие №5 Создать сценарий со встроенными функциями; Создать сценарий с пользовательской функцией.**

**Цель**: Научиться применять обработчики событий в сценариях JavaScript

**Задание**:

Создайте сценарий, в котором присутствует обработчик событий **onClick** (щелчок мышью). При нажатии на кнопку должно появиться окно с сообщением **«Сценарий заработал!!!»**

Создайте сценарий, в котором текст, при наведении на него курсором мыши будет окрашиваться в отличный от черного цвет.

**Порядок выполнения работы:**

**Содержание отчета:**

Наименование и цель работы

Код программы

Ответы на контрольные вопросы

Вывод о проделанной работе

**Практическое занятие № 6 Создать сценарий по обработке массивов.**

ПОИСК МАКСИМАЛЬНОГО ЭЛЕМЕНТА В МАССИВЕ

С ЗАПОМИНАНИЕМ ЕГО ПОЛОЖЕНИЯ (ИНДЕКСА)

ПОСТАНОВКА ЗАДАЧИ: Найти в массиве A, состоящем из n элементов, наибольший

элемент и запомнить его индекс (K).

Program MaxElMassiva; uses crt;

{ описание массива и переменных }

{ процедура ввода массива }

{ процедура вывода массива }

procedure MaxEl;

begin

max:=a[1]; k:=1;

for i:=1 to n do if max<a[i] then begin max:=a[i]; k:=i; end;

writeln('Максимальный элемент массива А равен ',max);

writeln('Максимальный элемент массива стоит на ' ,k, ' месте);

end;

Begin

ClrScr; n:=20;

Input;

ClrScr;

Print; writeln;

MaxEl;

Readkey;;

End.

З А Д А Н И Я:

1. Набрать и исполнить программу.

2. Найти минимальный элемент и его индекс.

3. Найти минимальный и максимальный элементы, и их индексы.

4. Заменить найденный минимальный элемент числом 100 и показать

на экране массив с этой заменой.

**Практическое занятие № 7 Создать сценарии, обрабатывающие строки.**

ПОСТАНОВКА ЗАДАЧИ: Определить, есть ли в заданном массиве элемент равный L.

Имеется массив А: 3 1 2 5 3 4 8 L=5

В массиве А имеется элемент равный 5

В данной программе можно использовать "флажок" - специальную переменную, которая принимает значения "истина" или "ложь" в зависимости от того найден элемент или нет.

**Program Poisk\_El\_Massiva; uses crt;**

. . .

**procedure Poisk(L: integer);**

**var fl: boolean;**

**begin**

**fl:=false;**

**for i:=1 to n do if a[i]=L then fl:=true;**

## **If fl then write(' Элемент равный ',l,' имеется ')**

**else write(' Элемент равный ',L,' отсутствует');**

**end;**

**Begin**

**ClrScr; n:=20;**

**Input;**

**ClrScr;**

**Print; writeln; writeln;**

**write('Введите значение искомого элемента: '); readln(L);**

**Poisk(L);**

**Readkey;**

**End.**

**З А Д А Н И Я:**

1. Набрать и исполнить программу, т. е. найти 3-4 элемента.

2. Если в массиве несколько одинаковых элементов, то найти первый из них и указать его индекс.

3. Подсчитать количество одинаковых элементов в массиве.

**Практическое занятие №8 Создать страницу с элементами формы**

Цель работы: Научиться. создавать формы.
Задачи работы:

Овладеть методикой работы по созданию форм с помощью команды FORM.

Получить навыки по созданию форм, кнопок командами: INPUT, SELECT, TYPE.

Обеспечивающие средства: Сборник описаний практических работ; операционная система Windows XP, программа Internet Explorer; программа «Блокнот»; персональный компьютер.
Требования к отчету: Итоги практической работы представить в виде файла lab10.html на диске.
персональный компьютер.
Технология работы:
1. Форма — это инструмент, с помощью которого HTML-документ может послать некоторую информацию в некоторую заранее определенную точку внешнего мира, где информация будет некоторым образом обработана.
Форма открывается меткой <FORM> и заканчивается меткой </FORM>. Создайте в окне «содержание » практической работы №8 следующие формы:

<HTML>

<HEAD>

<TITLE>Пример создания форм</TITLE>

</HEAD>

<H1>Несколько более сложная форма </H1>

<FORM ACTION="http://206.31.82.215/hp/nc/fd-win.pht" METHOD=post>

<H2>Расскажите немного о себе...</H2>

<P>Указывать подлинные данные совсем не обязательно.

Для целей демонстрации вполне подойдут и вымышленные. </P>

<P>Имя: <INPUT TYPE=text SIZE=40 NAME=fn><BR>

Фамилия: <INPUT TYPE=text SIZE=40 NAME=ln><BR>

Пол: <INPUT TYPE=radio NAME=gender VALUE="male" checked>мужской

<INPUT TYPE=radio NAME=gender VALUE="female">женский<BR>

Возраст: <INPUT TYPE=text SIZE=5 NAME=age> лет<BR>

<INPUT TYPE=submit VALUE="Запустить обработчик"></P>

<SELECT NAME="selection">
<OPTION VALUE="option1" checked>Вариант 1
<OPTION VALUE="option2">Вариант 2
<OPTION VALUE="option3">Вариант 3
</SELECT>

</FORM>

</BODY>

</HTML>

Метка <OPTION> определяет элемент меню. Обязательный атрибут VALUE устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Метка <OPTION> может включать атрибут checked, показывающий, что данный элемент отмечен по умолчанию.
Оформите: окна практической работы №8 в единой цветовой гамме BGCOLOR="FFF0FF", окно "main" на тон темнее; правой верхней части окна «head» вставьте «логотип»-HТML-прямоугольник 10х100, другие оформите по своему усмотрению.
2.Создайте все формы (Пример использования), приведенные в теоретической части. Покажите выполненное преподавателю.
Теоретическая часть:
Типы элементов формы
TYPE="TEXT"
Представляет собой одностроковое текстовое поле (рис. 10.1), физический размер которого можно установить с использованием атрибута SIZE, а первоначально введенную в него символьную последовательность — с помощью атрибута VALUE.

Рис. 10.1. Элемент формы TEXT
Примером использования данного элемента в коде HTML может служить следующая строка:
<INPUT TYPE='TEXT" SIZE="40" NAME="user\_name" VALUE-''Введите ваше имя">TYPE="PASSWORD"
Тип, полностью аналогичный предыдущему, за исключением того, что вводимая пользователем в текстовое поле информация будет отображаться на экране символами «\*», чтобы скрыть текст от постороннего зрителя.

TYPE="CHECKBOX"
Элемент, представляющий собой простую форму выбора, принимающую одно из двух устойчивых состояний: «отмечено» — «не отмечено» (рис. 10.2).

Рис. 10.2. Элемент формы CHECKBOX
Данный элемент оперирует так называемыми булевыми переменными, то есть переменными, каждая из которых может принимать значение «ИСТИНА» или «ЛОЖЬ». Каждый элемент формы CHECKBOX создает логическую пару значений вида «имя\_элемента—состояние» и передает их на сервер. Для установки первоначального состояния элемента (отмечен, либо нет) используют атрибут CHECKED.
Пример использования:
<INPUT TYPE="CHECKBOX" CHECKED NAME="C01" VALUE="yes">

TYPE="RADIO"
Так называемая радиокнопка (рис. 10.3) применяется в случае, когда какая-либо логическая переменная может принимать только одно значение из множества возможных.

Рис. 10.3. Элемент формы RADIO
Все элементы RADIO одной формы обозначаются одним и тем же значением атрибута NAME. Использование радиокнопок требует явного указания значений атрибута VALUE, одна из кнопок должна быть обязательно выделена атрибутом CHECKED.
Пример использования:
<Р АLIGN="СЕМТЕR">Пожалуйста, укажите ваш возраст</Р>
<CENTER>
<INPUT TYPE="RADIO" NAME= "user-age" VALUE="0-12">
<INPUT TYPE="RADIO" NAME= "user-age" VALUE="13-17">
<INPUT TYPE="RADIO" NAME= "user-age" VALUE="18-25">
<INPUT TYPE="RADIO" NAME= "user-age" VALUE="26-35" CHECKED>
<INPUT TYPE="RADIO" NAME= "user-age" VALUE="36-">
</CENTER>

TYPE="BUTTON"
Отображает обыкновенную кнопку (рис. 10.4), нажатие на которую приводит к какому-либо действию сервера.

Рис. 10.4. Элемент формы BUTTON
С помощью атрибута NAME данному элементу присваивается уникальное имя, атрибут VALUE позволяет изменять надпись, отображаемую на кнопке.
Пример использования:
<INPUT TYPE="BUTTON" VALUE="Button" NAME="B1">TYPE="SUBMIT"
Определяет кнопку, по нажатии которой данные из формы передаются серверу. Так же как и в предыдущем случае, надпись, отображаемая на кнопке, задается атрибутом VALUE.
Пример использования:
<INPUT TYPE="SUBMIT" VALUE="Отправить!">

TYPE="RESET"
Создает кнопку, которая очищает неправильно заполненную текстовую форму. Параметры и значения этой кнопки не передаются на сервер вместе с другими данными формы. Пример использования:
<INPUT TYPE="RESET" VALUE="Очистить форму">TYPE="FILE"
Генерирует на экране кнопку, по нажатии на которую на экране появляется Проводник Windows, позволяющий присоединить к отсылаемым на сервер данным любой файл с локального компьютера пользователя. Данный элемент применяется в основном в формах отправки с сервера сообщений электронной почты для организации вложений, а также для загрузки изображений на сервер. Обычно рядом с кнопкой отображается небольшое текстовое поле, куда автоматически заносится имя отсылаемого файла и путь к нему на локальном диске.
Пример использования:
<INPUT TYPE="FILE" NAME="picture">TYPE="IMAGE"
Создает кнопку отсылки, аналогичную элементу SUBMIT, но с использованием графического изображения, созданного пользователем. Обычно применяется вслучаях, когда стандартная серая прямоугольная кнопка «не вписывается» в дизайн сайта. URL к изображению, играющему роль кнопки, указывается атрибутом SRC, причем сам элемент может содержать собственные атрибуты, аналогичные применяемым в теге <IMG> (см. урок 5), в том числе ALIGN, ALT и др. Атрибуты NAME и VALUE трактуются так же, как аналогичные атрибуты элемента SUBMIT.
Пример использования:
<INPUT TYPE="IMAGE" SRC="/images/button.gif ALIGN="BOTTOM" NAME="submit" VALUE="Отправить!">
TYPE="HIDDEN"
Данный элемент является скрытым и не отображается на экране монитора пользователя. Применяется он для хранения и передачи на сервер информации о текущем состоянии формы: при нажатии кнопки SUBMIT элемент HIDDEN сформирует логическую пару переменных типа «имя—значение», которые будут отосланы серверу с помощью протокола HTTP. Элементы HIDDEN служат доступной альтернативой файлам cookies — специальным файлам, в которых сохраняются индивидуальные настройки пользователя и позволяющим, например, восстановить последнее состояние формы при повторном посещении пользователем содержащей эту форму страницы.
Пример использования:
<INPUT TYPE="HIDDEN" NAME="forml" VALUE="c3576-236-2113">
Однако с помощью тега <INPUT> можно передать далеко не все доступные web-дизайнеру элементы форм. Например, текстовые поля, задаваемые атрибутами TEXT и PASSWORD, весьма ограничены по количеству символов, которые можно ввести в эти поля. Если перед web-мастером встала задача создать такие текстовое поля, чтобы пользователь мог вводить большое количество символов, используется самостоятельный тег <TEXTAREA>, формирующий на экране одноименный элемент формы (рис. 10.5).

Рис. 10.5. Элемент формы TEXTAREA
Синтаксис записи данного тега выглядит следующим образом:
<TEXTAREA МАМЕ="имя элемента" ROWS-''целое число" СOLS="целое число">
Текст, выводимый в текстовом поле по умолчанию</TEXTAREA>
С атрибутом NAME, определяющим индивидуальное имя каждого элемента формы, вы уже знакомы. Атрибуты ROWS и COLS указывают соответственно максимально допустимое количество строк вводимого текста и символов в строке. В случае если набираемый пользователем текст не умещается в видимую часть текстового контейнера, по краям поля появляются вертикальные и горизонтальные полосы прокрутки для скроллинга содержимого элемента TEXTAREA.
Пример использования:
<TEXTAREA NAME="message" ROWS="25" COLS="40">
Введите сюда текст сообщения
</TEXTAREA>
Для создания меню выбора в форме (рис. 10.6) web-мастер может воспользоваться тегом <SELECT>.

Рис. 10.6. Элемент формы SELECT
Синтаксис записи данного тега в общем виде выглядит так:
<SELECT NAME="имя" MULTIPLE SIZE="целое число">
<OPTION VALUE="Пункт 1">Пункт 1</OPTION>
<OPTION VALUE-''Пункт 2">Пункт 2</OPTION>
<OPTION VALUE="Пункт 3">Пункт 3</OPTION>
OPTION VALUE="Пункт N">Пункт N</OPTION>
</SELECT>
Атрибут MULTIPLE определяет для пользователя возможность отметить не одну, а сразу несколько позиций из предложенного списка. Если данный атрибут задан, можно воспользоваться атрибутом SIZE, определяющим количество позиций, видимых на экране одновременно.
Пример использования:
<Р АLIGN ="CENTER">Пожалуйста, укажите ваш город</Р>
<CENTER>
<SELECT NAME="City" MULTIPLE SIZE="3">
OPTION VALUE="a">Санкт-Петербург</OPTION>
OPTION VALUE="b">Mocква</OPTION>
<OPTION VALUE="c">Екатеринбург</OPTION>
</SELECT>
</CENTER>
3. Создайте все формы из теоретической части и сохраните их в файле form\_10.txt, form\_10.html
4. Сохранить файл как lab10.txt в блокноте и как lab10.html для просмотра в браузере.

**Практическое занятие № 9 Создание сценариев с изображением и обработкой текста на нем.**

Задание № 1. Создание простейшего файла HTML 1. Создайте личную папку, куда вы будете сохранять все файлы своего сайта. 2. Запустите программу Блокнот (Notepad). 3. Наберите в окне программы простейший файл HTML.

Расписание занятий на вторник

4. Сохраните файл под именем RASP.HTML (обязательно укажите тип файла HTML при сохранении) в личной папке. 5. Для просмотра Web-страницы используйте любую программу браузера (Internet Explorer, Opera, Mozilla Firefox или другую). Для этого, не покидая программу Блокнот (сверните окно на панель задач), откройте личную папку и двойным кликом по файлу RASP.HTML откройте окно браузера.

На экране вы увидите результат работы, изображенный на рисунке 1. Задание № 2. Управление расположением текста на экране 1. При необходимости откройте текст Web-страницы в Блокноте (1 щелчок правой клавишей мыши по файлу RASP.HTML, в контекстном меню выбрать команду Открыть с помощью... и выбрать программу Блокнот). При необходимости открыть файл в браузере – двойной клик по значку файла левой клавишей мыши. 2. Внести изменения в файл RASP.HTML, расположив слова Расписание, занятий, на вторник на разных строках.

Расписание занятий на вторник

3. Сохраните текст с внесенными изменениями в файле RASP.HTML (меню Файл | Сохранить). Если у вас уже отображается Web-страница, то вам достаточно переключиться на панели задач на программу браузера и обновить эту страницу (кнопка ). Изменилось ли отображение текста на экране? Не удивляйтесь тому, что внешний вид вашей Web-страницы не изменился. Не забывайте каждый раз сохранять текст Web-страницы при ее корректировке в программе Блокнот и обновлять страницу при ее просмотре в программе браузера. Задание № 3. Некоторые специальные команды форматирования текста Существуют специальные команды, выполняющие перевод строки и задающие начало нового абзаца. Кроме того существует команда, запрещающая программе браузера изменять каким-либо образом изменять форматирование текста и позволяет точно воспроизвести на экране заданный фрагмент текстового файла. Тег перевода строки
отделяет строку от последующего текста или графики. Тег абзаца

тоже отделяет строку, но еще добавляет пустую строку, которая зрительно выделяет абзац. Оба тега являются одноэлементными, тег

– двойной, т.е. требуется закрывающий тег.

1. Внесите изменения в файл RASP.HTML

Расписание занятий
на вторник

2. Сохраните внесенные изменения, переключитесь на панели задач на программу браузера, обновите Web-страницу. Как изменилось отображение текста на экране?

**Практическое занятие № 10 Создать сценарии, содержащие функции обработки файлов, каталогов и БД.**

Средства защиты БД, реализованные в ACCESS, позволяют предотвратить умышленные или случайные просмотр, изменение и удаление информации лицами, которые не имеют соответствующих прав доступа. Эти средства особенно важны при функционировании БД в сети.

В ACCESS предусмотрены различные уровни защиты данных и администрирования доступа к ним. Возможности ACCESS позволяют обеспечить безопасность как самого приложения, так и файла БД. Простейшим средством защиты БД от несанкционированного доступа является *пароль*.

После того как пароль установлен, при каждом открытии БД будет появляться диалоговое окно, в которое требуется ввести пароль. Пользователи смогут открыть БД, только после ввода правильного пароля. Этот способ достаточно надежен, поскольку ACCESS шифрует пароль, так что к нему нет прямого доступа при чтении файла БД. Недостаток такого способа защиты в том, что он применяется только при открытии БД. После открытия БД все объекты становятся доступными для пользователя (если не определена защита на уровне пользователей). Для БД, которая совместно используется небольшой группой пользователей или на автономном компьютере, установка пароля обычно оказывается достаточной.

1) Установите пароль на любую из созданных вами баз данных. Для этого закройте свою БД, не выходя из Access, затем снова откройте ее, щелкнув в окне открытия БД справа от кнопки Открыть и выбрав «Монопольно». Открыв БД, в меню Сервис выберите Задать пароль и следуйте дальнейшим указаниям Access.

2) При следующем открытии БД система запросит установленный вами пароль. Снять пароль можно, снова открыв БД монопольно и выбрав в меню Сервис Удалить пароль. Естественно, перед удалением пароля система потребует ввести его, чтобы никто, кроме вас, не смог произвести эту операцию. Удалите пароль со своей базы данных.

3) Усовершенствовать защиту позволяют *средства поддержки рабочих групп*, *ведения учетных (регистрационных) записей*, *задания прав владения и прав доступа*. С помощью средств защиты можно указать, какие операции по обработке объектов БД разрешается выполнять пользователю или группе пользователей. О каждом пользователе или группе ведутся учетные записи с указанием прав доступа.

Рабочей называется группа пользователей, работающих с одной БД и имеющих общий файл рабочей группы. Файл рабочей группы - это системный файл с информацией о группе пользователей, работающих в многопользовательской среде с БД. В файлах рабочих групп хранятся учетные записи, пароли, а также данные о правах доступа к объектам БД.

При инсталляции ACCESS по умолчанию создается стандартный файл рабочей группы SYSTEM.MDW, сохраняемый в папке, в которой установлен ACCESS. Он используется программой до появления нового файла рабочей группы. Изменение стандартного или создание нового файла рабочей группы выполняет программа Администратор рабочих групп (Wrkgadm), находящаяся в подпапке System папки Windows. По умолчанию все файлы рабочей группы ACCESS имеют расширение .MDW.

Текущая рабочая группа указывается в файле рабочей группы, имя которого отображается в диалоговом окне программы «Администратор рабочих групп». Она находится среди системных файлов Windows, найти ее можно с помощью поиска в программе Проводник. Найдите и запустите Администратор рабочих групп.

Для изменения файла рабочей группы нажать кнопку Связать (Связь), вследствие чего на экране появится диалоговое окно Файл рабочей группы.

В поле База данных ввести полное имя нового файла рабочей группы. Этот файл можно найти, используя кнопку Обзор... После нажатия кнопки ОК программа сообщит об установлении связи с новой рабочей группой.

Для создания нового файла рабочей группы необходимо нажать кнопку Создать в окне Администратор рабочих групп, при этом откроется диалоговое окно Сведения о владельце рабочей группы (Имя, Организацию и Код группы – любую комбинацию из букв и цифр, не превышающую 20 символов), ввести соответствующие данные. По умолчанию отображается информация о пользователе, указанном при установке Microsoft Office 97.

После нажатия ОК на экране отобразится диалоговое окно Файл рабочей группы, поле База данных которого содержит полное имя существующего файла рабочей группы. При необходимости ввести новое имя и нажать кнопку ОК. Новый системный файл рабочей группы используется только после следующего запуска ACCESS.

Аналогичным образом для каждой группы пользователей можно создать файлы рабочих групп, которые рекомендуется поместить в одну папку. Файл рабочей группы не обязательно помещать в папку с файлом БД.

Сведения об имени, организации и коде рабочей группы необходимо записать и сохранить, т.к. эти данные могут понадобиться при восстановлении системного файла рабочей группы. Если по каким-либо причинам эти данные будут утеряны, восстановить БД не удастся.

После установки ACCESS пользователь получает право доступа ко всем объектам БД (становится членом группы Admins с именем Admin). Поскольку по умолчанию пароль в этой учетной записи не указывается, его не нужно вводить для входа в систему.

Члены группы Admins (администраторы) имеют право на модификацию БД. Пока не будет задана регистрация входа в систему всех членов рабочей группы, ACCESS будет разрешать вход в систему используя предопределенную регистрационную запись Admin. Чтобы задать регистрацию входа в систему и тем самым устранить произвольный доступ к ней, следует установить пароль для администратора в регистрационной записи Admin. В противном случае при каждом запуске ACCESS администратор будет регистрироваться как пользователь Admin без указания пароля.

Чтобы регистрировался вход в систему необходимо выполнить последовательность действий:

q Запустить ACCESS, указав рабочую группу, и открыть БД.

q В подменю Защита меню Сервис активизировать команду Пользователи и группы.

q В открывшемся диалоговом окне перейти на вкладку Изменение пароля, ввести в поле Новый пароль задуманный пароль и подтвердить его в поле Подтверждение. Поле Текущий должно остаться пустым. Затем - кнопка ОК.

При вводе пароля вместо введенных символов отображаются звездочки (\*). Пароль может иметь длину от 1 до 14 знаков и включать любые символы, кроме 0 кода ASCII. Следует помнить, что при вводе пароля различаются строчные и прописные символы.

При следующем запуске ACCESS на экране появится диалоговое окно Вход для ввода пароля. В поле Имя следует ввести имя пользователя, а в поле Пароль - пароль. После нажатия кнопки ОК программа продолжит процесс запуска, если пароль введен правильно.

Удалить пароль можно, выполнив последовательность действий:

q В подменю Защита меню Сервис активизировать команду Пользователи и группы.

q В открывшемся диалоговом окне перейти на вкладку Пользователи и, удостоверившись, что в поле Имя указано правильное имя пользователя, нажать кнопку Снять пароль.

Таким же образом устанавливается и изменяется пароль для остальных пользователей системы. Сначала они входят в систему под своим именем, не вводя пароль, затем указывают его.

4) *Учетные записи.*После создания рабочей группы можно приступить к внесению учетных (регистрационных) записей. По умолчанию создается учетная запись Admin, а также учетные записи групп Admins (Администраторы), Users (Пользователи) и предоставляются права доступа ко всем объектам.

Учетная запись администратора включена в рабочую группу Admins. Администратор имеет право доступа ко всем объектам, созданным в этой группе. Кроме администратора может быть указан владелец БД. В системе обеспечения безопасности ACCESS 97 владельцы объектов имеют особый статус. По умолчанию пользователь, создавший объект, становится владельцем объекта и прав на работу с ним.

Администраторы и владельцы наделены особыми правами:

q Администратор БД всегда может получить права доступа ко всем объектам, созданным членами данной рабочей группы;

q Владелец БД всегда может открыть БД;

q Владелец объекта наделен полными правами доступа к этому объекту.

q Так как регистрационная запись Admin одинакова для всех копий ACCESS, для защиты БД необходимо включить нового пользователя в группу Admins или заменить администратора Admin.

q Чтобы включить нового пользователя в группу Admins, необходимо выполнить действия:

o Запустить ACCESS, выбрав рабочую группу и открыть любую БД

o В подменю Защита меню Сервис активизировать команду Пользователи и группы.

o В открывшемся диалоговом окне перейти на вкладку Пользователи и нажать кнопку Создать.

o В появившемся диалоговом окне в поле Имя ввести имя учетной записи (пользователя), а в поле Код - уникальный код записи. Затем - кнопка ОК.

o Имя пользователя может иметь длину от 1 до 20 символов и включать буквы, цифры, пробелы и другие символы за исключением символов: " \ [] ^| <> + = ; , ? \* и управляющих символов (с кодами ASCII от 00 до 31). Кроме того, имя записи не может начинаться с пробела.

o После возвращения в окно Пользователи и группы выбрать в списке Имеющиеся группы группу Admins и нажать кнопку Добавить. Группа Admins будет помещена в список Участие в группе, что обеспечит ввод учетной записи в группу Admins. В завершение нажать ОК.

Такая же последовательность операций выполняется при создании учетной записи владельца, которая, в зависимости от требования защиты, может быть внесена в группу Admins или в другую группу.

Кроме учетных записей администратора и владельца в ACCESS 97 создаются записи для пользователей и групп пользователей, которые могут иметь одинаковые права.

Примечание:

Учетные записи пользователей должны создаваться в рабочей группе, которая будет применяться при запуске приложения.

Для создания новой группы пользователей и соответствующей учетной записи надлежит активизировать команду Пользователи и группы подменю Защита меню Сервис, в открывшемся диалоговом окне выбрать вкладку Группы и нажать кнопку Создать.

В открывшемся диалоговом окне в поле Имя указывается имя группы, а в поле Код - уникальный код (любая комбинация символов длиной от 4 до 20 знаков).

На имя группы накладываются те же ограничения, что и на имя пользователя.

Удаление учетной записи или группы из рабочей группы осуществляется следующим образом:

q Запустить ACCESS с выбранной рабочей группой и открыть БД.

q В подменю Защита меню Сервис активизировать команду Пользователи и группы.

q В открывшемся диаловом окне перейти на вкладку Пользователи (при удалении учетной записи пользователя) или Группы (при удалении учетной записи группы).

q В списке Имя выбрать выбрать имя пользователя или группы для удаления, нажать кнопку Удалить, затем подтвердить удаление, нажав кнопку Да.

q Закрыть окно Пользователи и группы.

После создания учетных записей пользователей и групп взаимосвязь между ними можно проследить, нажав кнопку Отчет о защите. При этом появляется диалоговое окно, в котором можно выбрать необходимые сведения и вывести их на печать.

5) *Изменение прав владения.*После инсталляции ACCESS пользователь Admin является владельцем любой БД и всех объектов. Поскольку для него не устанавливается пароль, для защиты БД от несанкционированного доступа необходимо изменить права владения базой и всеми объектами.

Существует несколько способов смены владельца объектов БД:

q импортирование всех объектов БД в новый файл;

q использование вкладки Смена владельца диалогового окна Разрешения.

Для первого способа последовательность действий такая:

q Запустить ACCESS с выбранной рабочей группой.

q Войти в систему под именем нового владельца БД

q Создать новую (пустую) БД.

q В подменю Внешние данные меню Файл активизировать команду Импорт.

q В окне Импорт указать БД, владельца которой необходимо сменить, и скопировать все ее объекты в новую БД.

Более простой способ смены владельца БД состоит в следующем:

q Открыть БД и выбрать в меню Сервис команду Защита.

q В открывшемся подменю активизировать команду Разрешения.

q В появившемся диалоговом окне открыть вкладку Смена владельца.

q Выбрать тип объекта в списке Тип объекта. В области Объект отобразится список всех объектов выбранного типа, а в области Текущий владелец - имена владельцев.

q Выделить в списке Объекты те объекты, владельца которых необходимо сменить.

q В поле Новый владелец указать учетную запись пользователя, которому предоставляются права владельца этих объектов. Если в качестве владельца будет выступать группа, следует предварительно активизировать переключатель Группы.

q Нажать кнопку Сменить владельца.

Присвоение и удаление прав доступа. Администратор БД предоставляет всем или некоторым членам рабочей группы права доступа к различным объектам базы. Права доступа хранятся в файле БД и характеризуют ее объекты.

К разграничению прав доступа пользователей и групп можно приступить после создания рабочей группы, определения администратора и владельца БД, а также создания учетных записей пользователей и групп. Пользователь наследует права той группы, к которой принадлежит.

Перечень прав доступа, определенных в ACCESS приведен в следующей таблице:

|  |  |  |
| --- | --- | --- |
| Право доступа | Действие | Объекты доступа |
| Открытие/Запуск | Открытие БД, формы, отчета или запуск макроса | БД, формы, отчеты и макросы |
| Чтение макета | Просмотр объектов в режиме конструктора | Таблицы, запросы, формы, отчеты, макросы, модули |
| Изменение макета | Просмотр, изменение и удаление объектов в режиме конструктора | Таблицы, запросы, формы, отчеты, макросы и модули |
| Администратора | Полный доступ к объектам и данным, включая возможность присвоения прав доступа | БД, таблицы, запросы, формы, отчеты, макросы и модули |
| Чтение данных | Просмотр данных | Таблицы и запросы |
| Обновление данных | Просмотр и изменение данных без вставки и удаления | Таблицы и запросы |
| Вставка данных | Просмотр и вставка данных без изменения и удаления | Таблицы и запросы |
| Удаление данных | Просмотр и удаление данных | Таблицы и запросы |
| Монопольный доступ | Открытие БД в монопольном режиме | БД |

Наряду с правами доступа к существующим объектам предоставляются и права доступа к новым объектам. Из таблицы видно, что некоторые права доступа обуславливают наличие других прав. Так таблица, в которой разрешено обновление данных, доступна для чтения данных и макета. При наличии прав администратора пользователю доступны все перечисленные выше права.

Определить права доступа к некоторому объекту может владелец этого объекта, администратор в рабочей группе Admins или пользователь, с правами администратора для данного объекта.

Все права доступа к объекту сохраняются при его изменении только в том случае, если не применялся буфер обмена или не выполнялся импорт/экспорт объекта. Кроме того, все связанные с объектом права доступа могут быть утеряны при сохранении объекта под новым именем посредством команды Сохранить как/экспорт.

Чтобы предоставить права доступа к объектам БД, необходимо выполнить действия:

q Открыть БД. При этом пользователь, вошедший в систему, должен иметь право доступа к объектам.

q Выбрать в меню Сервис команду Защита. В открывшемся подменю активизировать команду Разрешения.

q В появившемся диалоговом окне перейти на вкладку Разрешения.

q В поле Тип объекта выбрать тип объекта, на который распространяются новые права.

q В списке Имя объекта отметить объекты, доступ к которым будет разрешен пользователю или группе, указанным в списке Пользователи и группы.

q Выбрать право доступа к отмеченным объектам.

q Нажать ОК.

Шифрование БД. Шифрование - это защита БД от несанкционированного доступа с помощью текстового редактора или средств работы с файлами, например, входящих в состав Windows или Norton Utilites. Информация в зашифрованной БД недоступна для чтения. Шифрование несколько замедляет работу ACCESS, т.к. расходуется время на шифрование и дешифрование файлов.

Шифрование и дешифрование БД могут производить только члены группы Admins. Эти операции выполняются следующим образом:

q Запустить ACCESS с выбранной рабочей группой.

q В подменю Защита меню Сервис активизировать команду Шифровать/дешифровать, вследствие чего на экране появится окно выбора БД для шифрования.

q Выбрать БД, которую необходимо зашифровать или дешифровать, и нажать ОК.

q Если выбранная БД не зашифрована, откроется диалоговое окно Шифрование базы данных под именем, в котором программа предложит новое имя для зашифрованной БД. Если же выбранная БД зашифрована, появится окно Дешифрование базы данных под именем.

q Выбрать имя файла, и нажать кнопку Сохранить.

**Практическое занятие № 11. Создать сценарии, содержащие функции обработки файлов, каталогов и БД.**

**ЗАДАНИЕ**

Создайте базу данных «Библиотека**»**. База данных должна содержать 4 взаимосвязанных таблицы: **Книги, Авторы, Издательства, Книги-Авторы** (для связи таблиц **Книги** и **Авторы**)*.*

**ХОД РАБОТЫ**

1. Запустите MS Access через меню **ПУСК**.
2. При первом запуске программы Access отображается окно **Приступая к работе с Microsoft Office Access**.

1. На странице **Приступая к работе с Microsoft Office Access** в разделе **Новая пустая база данных** выберите команду **Новая база данных**..

1. В области **Новая база данных**. Если имя файла указано без расширения, расширение будет добавлено автоматически. Чтобы сохранить файл в другой папке, отличной от используемой по умолчанию, нажмите кнопку **Открыть**(рядом с полем **Имя файла**), перейдите к нужной папке и нажмите кнопку **ОК**.

1. В окне *Новая база данных* укажите место размещения файла, для этого нажмите кнопку **Открыть** перейдите к нужной папке и нажмите кнопку **ОК**

1. В поле **Имя файла** введите имя файла **БД\_Библиотека\_Фамилия студента**(например, БД\_Библиотека\_Иванов И.И.)***.***Если имя файла указано без расширения, расширение будет добавлено автоматически . Щелкните по кнопке *Создать*.
2. Приложение Access создаст базу данных с пустой таблицей с именем «Таблица1» и откроет эту таблицу в режиме таблицы. Курсор находится в первой пустой ячейке столбца **Добавить поле**.

**Создание таблиц в режиме Конструктора**

В режиме конструктора сначала создается структура новой таблицы. Затем можно переключиться в режим таблицы для ввода данных или ввести данные, используя другой метод, например вставку или импорт.

1. На вкладке **Создание** в группе **Таблицы** щелкните **Конструктор таблиц**.

1. В режиме конструктора необходимо задать **поля таблицы**, определить их **типы данных** и **другие свойства**. Для каждого поля в таблице введите имя в столбце **Имя поля**, а затем в списке **Тип данных** выберите тип данных.

**СПРАВОЧНАЯ ИНФОРМАЦИЯ**

**Таблица может содержать следующие типы полей:**

Свойство **Тип данных** (DataType) определяет тип данных, сохраняемых в поле таблицы. В каждое поле допускается ввод данных только одного типа.

|  |  |  |
| --- | --- | --- |
| **Тип данных** | **Хранит** | **Размер** |
| Текстовый | Алфавитно-цифровые знакиИспользуется для текста или текста и чисел, не применяемых в расчетах. | До 255 знаков. |
| Поле МЕМО | Алфавитно-цифровые знаки (более 255 знаков) или форматированный текст. | До 65 535 знаков. |
| Числовой | Числовые значения (целые или дробные).Используется для хранения числовых данных, используемых в вычислениях, за исключением денежных значений. | 1, 2, 4 и 8 байт или 16 байт (если используется для кода репликации). |
| Дата/время | Даты и время.Используется для хранения значений даты и времени. | 8 байт. |
| Денежный | Денежные значения.Используется для хранения денежных значений (валюты). | 8 байт. |
| Счетчик | Уникальное числовое значение, которое автоматически вводит Office Access 2007 при добавлении записи.Используется для создания уникальных значений, например для первичного ключа. В поле с типом данных «Счетчик» числа могут последовательно увеличиваться на указанное приращение или выбираться случайно. | 4 байта или 16 байт, если используется для кода репликации. |
| Логический | Логические значения.Используется для полей, которые могут содержать одно из двух значений, например «Да» и «Нет» или True и False. | 1 бит (8 бит = 1 байт). |
| Поле объекта OLE | OLE-объекты или другие двоичные данные.Используется для хранения OLE-объектов других приложений Microsoft Windows. | До 1 Гбайт. |
| Вложение | Рисунки, изображения, двоичные файлы, файлы Microsoft Office.Стандартный тип данных для сохранения цифровых изображений и любого типа двоичных файлов. | Для сжатых вложений — 2 гигабайта. Для несжатых вложений примерно 700 Кбайт в зависимости от степени возможного сжатия вложения. |
| Гиперссылка | Гиперссылки.Используется для хранения гиперссылок вызова веб-страниц одним щелчком с помощью URL-адреса или файлов с помощью формата универсального имени UNC. Кроме того, можно использовать ссылку на объекты Access, хранящиеся в базе данных. | До 1 гигабайта знаков или 2 гигабайт памяти (2 байта на знак), из которых в элементе управления можно отобразить 65 535 знаков. |
| Мастер подстановок | Фактически типом данных не является, а вызывает мастер подстановок.Используется для запуска мастера подстановок, с помощью которого можно создать поле, позволяющее выбрать значение из другой таблицы, запроса или списка значений, используя поле со списком. | На основе таблицы или запроса — размер привязанного столбца.На основе значения — размер текстового поля, содержащего значение. |

1. При желании можно ввести описание для каждого поля в столбце **Описание**. Это описание будет отображаться в строке состояния, когда в режиме таблицы курсор будет находиться в данном поле.
2. Задайте поля таблицы **Книги** в соответствии со столбцами **Поле** и **Тип данных** таблицы 1. Тип данных указывается выбором элемента из раскрывающегося списка.

Таблица 1. – Структура таблицы: Книги

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип данных** | **Размер поля** | **Маска ввода** |
| Код книги | Числовой | целое |  |
| Название книги | Текстовый | 30 |  |
| Количество страниц | Текстовый |  | ###”стр.” |
| Количество экземпляров | Числовой |  |  |
| Стоимость книги | денежный |  |  |
| Год издания | Дата\Время |  |  |
| Код издательства | Числовой | целое |  |
| Предметная область | Текстовый | 20 |  |
| Вид печатной продукции | Текстовый | 10 |  |

1. Сделайте поле **Код книги**ключевым

- вкладка **Конструктор**/ группа **Сервис**/команда **Ключевое** **поле**

1. Щелкая курсором на полях, задайте для них свойство **Размер поля**(выбирая из списка или вводя значение с клавиатуры)в соответствии с таблицей 1.
2. Для полей **Вид печатной продукции**, **Предметная область**определите список фиксированных значений.

**Вид печатной продукции:** газета, журнал, книга, брошюра.

**Предметная область –**информатика, экономика, психология.

Для этого в окне свойств полей выберете вкладку **Подстановка**, затем **Тип элемента управления** установить **Поле со списком (или Список)**, далее **Тип источника строк – Список значений,**и в свойстве**Источник строк**ввести список элементов разделяемых точкой с запятой**(;).**

Например для поля**Предметная область –**информатика; экономика; психология

1. Задайте для поля Количество страниц **Маску ввода**в соответствии с таблицей 1.

**СПРАВОЧНАЯ ИНФОРМАЦИЯ**

**Свойство полей «Маска ввода».**

Свойство Маска ввода (InputMask) задает маску ввода, облегчающую ввод данных в элемент управления- поле. Например, удобно создать следующую маску ввода для поля <Телефон>, позволяющую вводить только цифры и автоматически добавляющую промежуточные символы: (\_\_\_) \_\_\_-\_\_\_\_. Значение данного свойства определяется автоматически при использовании мастера по созданию масок ввода.

При создании маски ввода пользователь имеет возможность указать, что часть данных следует вводить обязательно (например, региональный код для телефонных номеров), а другие данные являются необязательными (например, добавочный номер телефона). Эти символы определяют тип данных, например номер символа, который необходимо ввести для каждого символа маски ввода.

Символы, которые следует вводить в маску ввода определяются следующими специальными символами.

|  |  |
| --- | --- |
| **Символ** | **Описание** |
| **0** | Цифра (обязательный символ; знаки (+) и (-) не разрешены). |
| **9** | Цифра или пробел (необязательный символ; знаки (+) и (-) не разрешены). |
| **#** | Цифра или пробел (необязательный символ; незаполненные позиции выводятся как пробелы в режиме редактирования, но удаляются при сохранении данных; знаки (+) и (-) не разрешены). |
| **L** | Буква (обязательный символ). |
| **?** | Буква (необязательный символ). |
| **A** | Буква или цифра (обязательный символ). |
| **a** | Буква или цифра (необязательный символ). |
| **&** | Любой символ или пробел (обязательный символ). |
| **C** | Любой символ или пробел (необязательный символ). |
| **. , : ; - /** | Десятичный разделитель, разделители групп разрядов, времени или даты. |
| **<** | Преобразует все символы к нижнему регистру. |
| **>** | Преобразует все символы к верхнему регистру. |
| **!** | Указывает, что маска ввода заполняется справа налево; этот символ следует использовать, если в левой части маски находятся позиции, заполнять которые не обязательно. Маски ввода обычно заполняются слева направо. Символ восклицательного знака можно помещать в произвольную позицию в маске ввода. |
| **\** | Указывает, что следующий символ следует воспринимать как постоянный (а не специальный) символ (например, \A представляет символ A). |

1. Для поля **Код книги** задайте свойство **Подпись** – **Регистрационный номер** и поле **Обязательное поле – Да**.

**СПРАВОЧНАЯ ИНФОРМАЦИЯ**

**Свойство "Подпись".**

Свойство Подпись (Caption) определяет текст, который выводится в подписях объектов в различных режимах. Подпись поля указывает текст, который выводится в подписях, присоединенных к элементам управления, создаваемых путем переноса с помощью мыши из списка полей. Этот текст также выводится для таблицы или запроса в заголовке столбца в режиме таблицы.

**Свойство "Обязательное поле".**

Свойство Обязательное поле (Required) указывает, требует ли поле обязательного ввода значения. Если это свойство имеет значение "Да", то при вводе новой записи необходимо ввести значение в это поле или в любой присоединенный к нему элемент управления. Пустые (Null) значения в этом поле не допускаются. Например, можно потребовать, чтобы в элементе управления "Фамилия" в каждой записи обязательно выводилась какая-либо фамилия. Чтобы позволить ввод в поле пустых значений, недостаточно указать для свойства Обязательное поле значение "Нет".

1. Когда все необходимые поля будут добавлены, сохраните таблицу:

- Щелкните значок **Кнопка Microsoft Office** , а затем выберите команду **Сохранить**.

1. Перейдите в режим заполнения таблицы, щелкнув по кнопке «Вид» инструментальной панели.
2. Заполните таблицу 10-ю записями о книгах из несколько предметных областей.
3. Аналогично таблице **Книги** создайте и сохраните еще 3 таблицы: **Авторы**, **Издательства** и **Книги-Авторы**, в соответствии с таблицей 2. В столбце **Поле** названияполей**,**которые нужно сделать **ключевыми** – выделены жирным шрифтом.

Таблица 2. – Структура таблиц: Авторы, Издательства и Книги-Авторы

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Тип данных** | **Размер поля** |
| Авторы | **Код автора** | Числовой | Целое |
| Фамилия | Текстовый | 30 |
| Имя | Текстовый | 20 |
| Отчество | Текстовый | 15 |
| Страна | Текстовый | 20 |
| Книги-Авторы | **Код автора** | Числовой | Целое |
| **Код книги** | Числовой | целое |
| Издательства | **Код издательства** | Числовой | целое |
| Название | Текстовый | 40 |
| Город | Текстовый |  |
| Примечание | Текстовый |  |

1. Задайте схему данных таблиц: вкладка **Работа с базами данных**/группа **Показать или скрыть**/команда **Схемы данных .**

1. В появившемся диалоге **Добавление таблицы**, добавьте все таблицы (Авторы, Издательства, Книги, Книги-Авторы), последовательно указывая на названия таблиц и щелкая кнопку **Добавить**. Затем нажмите кнопку **Закрыть**.
2. Таблицы в схеме данных можно перемещать и менять размер. Сделайте все поля таблиц видимыми и расположите таблицы последовательно.
3. Свяжите первичный ключ таблицы **Книги (Код книги)** с внешним ключом таблицы **Книги-Авторы (Код книги)**связью «один–ко–многим». Для этого позиционируемся курсором на поле первичного ключа, щелкаем левой кнопкой мыши, не отпуская, перетаскиваем появившийся значок на поле внешнего ключа.
4. В появившемся диалоге **Изменение связей**установите **обеспечение целостности данных, каскадное обновление и удаление данных**. Нажмите кнопку **Создать**.

1. Аналогичным образом установите связи между таблицами **Авторы (Код автора)** и **Книги-Авторы (Код автора)**, **Книги (Код издательства)**и**Издательства (Код издательства)**.
2. Сохраните и закройте окно **Схема данных**.
3. Заполните таблицы **Авторы, Издательства** и **Книги-Авторы** записями: для каждой таблицы – 10 записей. При этом значения внешних ключей не должны отличаться от значений первичных.

**Список литературы:**

1. Зудилова Т.В., Бурков М.Л. – Web-программирование HTML – М. : ИД «ФОРУМ» : ИНФРА-М 2014 - https://e.lanbook.com/book/40724
2. Зудилова Т.В., Буркова М.Л. Web-программирование JavaScript М. : ИД «ФОРУМ» : ИНФРА-М 2014 - - https://e.lanbook.com/book/43561
3. Одиночкина С.В. Web-программирование PHP М. : ФОРУМ : ИНФРА-М 2014 - - https://e.lanbook.com/book/43562
4. Никулова Г.А. WEB-программирование. Клиентские технологии: SVG: Учебно-методическое пособие – М.:КУРС, НИЦ ИНФРА-М 2017 - - https://e.lanbook.com/book/111987
5. Никулова Г.А., Субботин В.Р. – WEB-программирование. Серверные технологии: РНР.Часть 1: Учебно-методическое пособие – Мн.:РИПО 2017 – k https://e.lanbook.com/book/111934