

Лабораторный практикум
«Разработка распределенных приложений в среде .Net»
по дисциплине «Сетевые технологии»

Аннотация

Практикум предназначен для 4 курса студентов специальности «Информатика и вычислительная техника» для использования при изучении учебной дисциплины «Сетевые технологии». Практикум рассчитан на 20 академических часов.

Microsoft .Net находит все большее применение у системных разработчиков как очень удобное RAD-средство для разработки как Windows-приложений, так Web-приложений, распределенных систем.

В практикуме предоставлена информация об основных возможностях платформы .Net, предлагаются сведения об объектно-ориентированном языке программирования C#, возможностях по созданию пользовательского интерфейса Windows-приложений, работе с БД и созданию Web-приложения с дальнейшим его размещением в облачной платформе Microsoft Azure.

Содержание

| | |
|--|----|
| Введение..... | 4 |
| Лабораторная работа №1. Создание консольных приложений на С# | 23 |
| Лабораторная работа №2. Создание Windows-приложения на С#. Регистрация на портале Microsoft Azure..... | 27 |
| Лабораторная работа №3. Создание БД MySQL и использованием Microsoft Azure | 38 |
| Лабораторная работа №4. Создание Web-приложения. Размещение Web-приложения в Microsoft Azure | 42 |

Введение

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

В .NET обязательно использование пространства имен, как способа организации системы типов в единую группу.

Пространство имен задается следующим образом:

```
using <ИмяПространстваИмен>.
```

```
using System;  
using System.Drawing;
```

Пространство имён System содержит все базовые типы, пространство имён System.Collection содержит типы для управления коллекциями объектов, пространство имён System.Data содержит базовые типы для управления базами данных, пространство имён System.Net содержит типы, устанавливающие сетевые соединения WebRequest, WebResponse, TopClient, TopListener, UdpClient, Sockets.

Пространство имён System.Web.UI.WebControls содержит типы, создающие средства управления пользовательским интерфейсом для Web-приложений, пространство имён System.Windows.Forms содержит типы, создающие средства управления пользовательским интерфейсом для Windows-приложений.

Особенностью ASP.NET, является использование серверных элементов управления (Server Controls). Файлы страниц ASP.NET, могут иметь различные расширения. Файл стандартной ASP.NET - страницы имеет расширение .aspx. Файл Web-службы имеет расширение .asmx, а файл пользовательского элемента управления - расширение .ascx.

ASP.NET поддерживает создание веб - приложений и Web-сервисов. C# – объектно-ориентированный язык программирования специально разработан для платформы .NET.

Пример кода C#:

```
using System;  
namespace my  
{  
    class MainClass  
    {  
        public static long f(long n)  
        {  
            return n *2;  
        }  
    }  
}
```

}

Жизненный цикл ASP.NET страницы состоит из следующих событий: инициализация (Page_Init), восстановление состояния управляющих элементов, загрузка страницы (Page_Load), события страницы (изменения или действия), сохранения состояния, выгрузка страницы (Page_Load).

Среда разработки Visual Web Developer Express Edition содержит возможности визуального конструирования веб – приложений, редактирования их кода, отладки, предоставляет возможности разработки приложений с использованием доступа к удаленным данным, также обеспечивает защиту доступа при использовании сайта.

Особенность средства разработки состоит в том, что можно сразу анализировать работу приложения на сервере, средство включает средства отладки на сервере.

Web Developer позволяет осуществлять визуальное конструирование веб-приложений. Для создания макетов веб-страниц имеется графический конструктор. Технология базовых страниц (Master Pages) позволяет обеспечить единый стиль сайта и сокращает время разработки. Возможности IntelliSense предоставляет быстрый доступ к методам и библиотекам, необходимым для ваших приложений.

Продукт позволяет создать динамически обновляемый веб-сайт, использующий базу данных. Конструктор баз данных и запросов к ним поддерживает визуальные методы работы. Система снабжена развитой системой помощи.

Базовые возможности Web Developer состоят в визуальном конструировании сайтов, поддержки HTML, CSS и JavaScript, поддержки XML, RSS и веб-сервисов, наличии средств визуального конструирования данных, возможностях настройки страниц, наличии мастеров для создания сайтов и шаблоны страниц.

При создании веб-приложения в Visual Web Developer, можно сразу определить место их размещения на веб-сервере, FTP-узле или в файловой папке на диске.

Для конфигурирования веб-сайта применяется инструмент Web Site Administration Tool, который является административным веб-интерфейсом сайта. С помощью этого инструмента можно управлять пользователями, ролями, персональной информацией, провайдерами данных, параметрами приложений и др. Для администрирования пользователей и ролей служит вкладка Security.

В Visual Web Developer имеется встроенная поддержка FTP, позволяющая устанавливать веб-приложения на сервер компании-провайдера.

Среда разработки предлагает шаблон полнофункционального работоспособного личного сайта Personal Web Site Starter Kit.

Visual Web Developer позволяет легко размещать веб-сайты, просто копируя файлы на сервер. Вся информация о конфигурации сайта содержится в XML-файлах, входящих в проект.

Базы SQL Server Express Edition можно развертывать простым копированием файлов БД на сервер, если на нем уже настроен SQL Server.

Для реального размещения сайта нужны поставщики услуг хостинга с поддержкой ASP.NET.

В платформу .Net входят много языков(в целом 16), которые без особого труда интегрируются друг с другом. Язык C# был создан специально для данной платформы, прочие языки были лишь адаптированы для данной платформы. Язык C# схож с языками C++ и Java, но имеет и отличия.

Структура программы на C#

Рассмотрим структуру на самом классическом примере «Hello, world!».

```
class Hello
{
    public static void Main(string[] args)
```

```

    {
        System.Console.WriteLine("Hello, world!");
    }
}

```

Любая программа на языке C# - это набор типов. В одном из типов программы должна находиться так называемая «точка входа» - статический метод Main. Наличие или отсутствие этого метода определяет тип получаемого результата компиляции – сборки. Если метод присутствует – получаем исполняемую программу, в противном случае – библиотеку DLL. Типы могут быть вложены друг в друга.

Библиотека типов и пространства имен

Выше было сказано, что сборка в .Net – это набор типов. Также упоминали о том, что в .Net Framework входит большая библиотека типов, единая для всех языков этой платформы - Framework Class Library (FCL). В ней хранится множество типов. Библиотека классов имеет в себе части, посвященные работе с данными (ADO.NET), с веб приложениями(ASP.NET), с windows интерфейсом (Windows Forms), с XML (DOM, XSD, XSLT, сериализация/десериализация, интеграция с ADO.NET), с сетевыми приложениями (System.Net), с распределенными приложениями (remoting), с COM+ (Enterprise Services, Serviced components), с шифрованием данных (System.Security.Cryptography).

Для удобства ее использования и удобства работы со всеми создаваемыми типами сама библиотека имеет иерархическую структуру пространств имен, в которых и находятся типы. Одним из наиболее важных пространств имен верхнего уровня является пространство имен System. Рассмотрим внимательнее строку примера

```
System.Console.WriteLine("Hello, world!");
```

В этой строке для вывода заветного "Hello, world!" мы используем метод WriteLine класса Console, который находится в пространстве имен System. Что же делать в случае, если нам предстоит тридцать раз

использовать этот класс. Разумеется, сочетания клавиш <Ctrl+C> и <Ctrl+V> могут помочь. Но есть вариант получше. Рассмотрим следующий пример.

```
using System;  
class Hello  
{  
    public static void Main(string[] args)  
    {  
        Console.WriteLine("Hello, world!");  
    }  
}
```

Теперь мы вынесли пространство имен System в блок using, и можем далее нигде не указывать это пространство имен, если будем использовать типы из этого пространства. Следует заметить, что, если в этом пространстве имеются другие пространства имен, то указание этого пространства в блоке using не означает, что к классам вложенных пространств можно получить доступ без указания этого пространства.

Структура типов

Типы в C# имеют структуру, представленную на рис. 1.

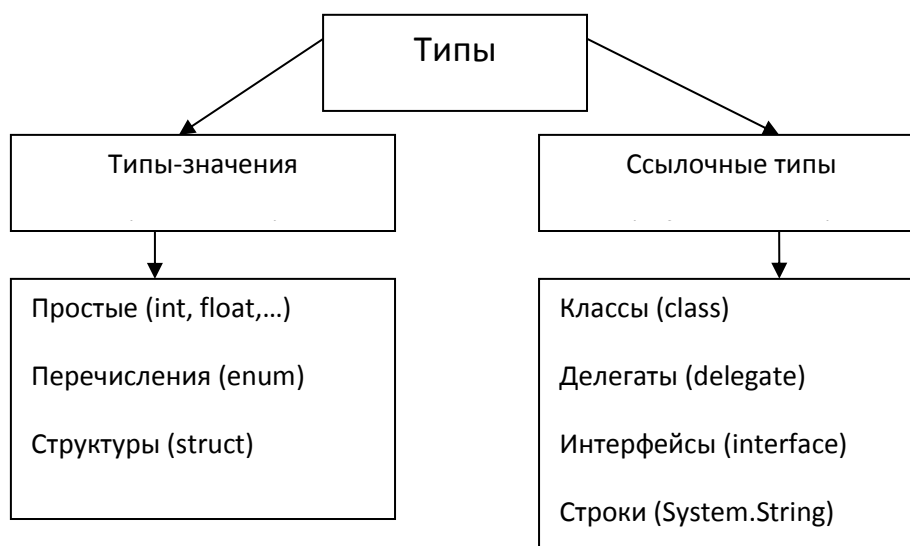


Рисунок 1 – Типы данных в C#

К простым типам относятся: sbyte, byte, char, short, ushort, int, uint, long, ulong, float, double, decimal.

Перечисления

Перечисление – это тип-значение, состоящий из набора поименованных констант. Укажем формат определения перечисления.

```
enum Имя [:базовый класс]
{
    Имя0 = Значение0,...,ИмяN = ЗначениеN
}
```

По умолчанию нумерация начинается с нуля и увеличивается на 1. В качестве примера опишем перечисление, указывающее день недели:

```
enum DayOfWeek {Monday=1, Tuesday, Wednesday, Thursday, Friday,
Saturday, Sunday};
```

Структуры

Структуры во многом схожи с классами. Основное их отличие в том, что структуры не являются ссылочным типом. Дополнительно, на структуры накладываются следующие ограничения:

- структура не может иметь деструктор;
- структура не может иметь конструктор без параметров;
- структура не может использоваться как базовый тип.

В качестве примера укажем структуру, описывающую окружность, которая в свою очередь использует структуру, описывающую точку.

```
struct Point
{
    public int x,y;
    public Point(int p1, int p2)
    {
        x=p1; y=p2;
    }
}
struct Circle
{
    public Point Centre;
```

```

        public int Radius;
        public Circle(int p1,int p2,int p3)
        {
            Centre=new Point(p1,p2);
            Radius=p3;
        }
    }

```

Ссылочные типы будут рассмотрены ниже.

Переменные

C# в отличие от Visual Basic является языком со строгой типизацией. В C# определено 7 видов переменных: статические, экземплярные, элементы массивов, входные параметры, выходные параметры, локальные переменные.

Статические переменные – это переменные класса, объявленные с указанием служебного слова `static`. Эти переменные одинаковы для всех экземпляров класса. Обращение к ним производится по имени класса, а не по имени экземпляра. Экземплярные переменные для каждого экземпляра могут хранить свое значение. Рассмотрим это на примере.

```

class Variables
{
    static int N1;
    int N2;
    public Variables(int p1, int p2)
    {
        N1=p1;
        N2=p2;
    }

    public static void Main(string[] args)
    {
        Variables var1 = new Variables(10,20);
        System.Console.WriteLine("Статическая пер.: "+Variables.N1.ToString());
        Variables var2 = new Variables(100,200);
        System.Console.WriteLine("Статическая пер.: " +Variables.N1.ToString());
        System.Console.WriteLine("Экземплярные пер.: "+var1.N2.ToString()+
"+var2.N2.ToString());
    }
}

```

После инициализации переменной var2 значение N1 изменилось, т.к. это общая переменная для всего класса.

Локальные переменные – это переменные, объявленные в теле блока.

Примеры

```
class Variables
{
    public static void Main(string[] args)
    {
        int a;
        {
            int b;
        }
        {
            int a; //компилятор сошлется на эту строку, она пересекается с
внешней переменной
            int b; // с этой переменной все в порядке, т.к. она находится в разных
блоках кода
        }
    }
}
```

Результат компиляции кода представлен на рис. 2.

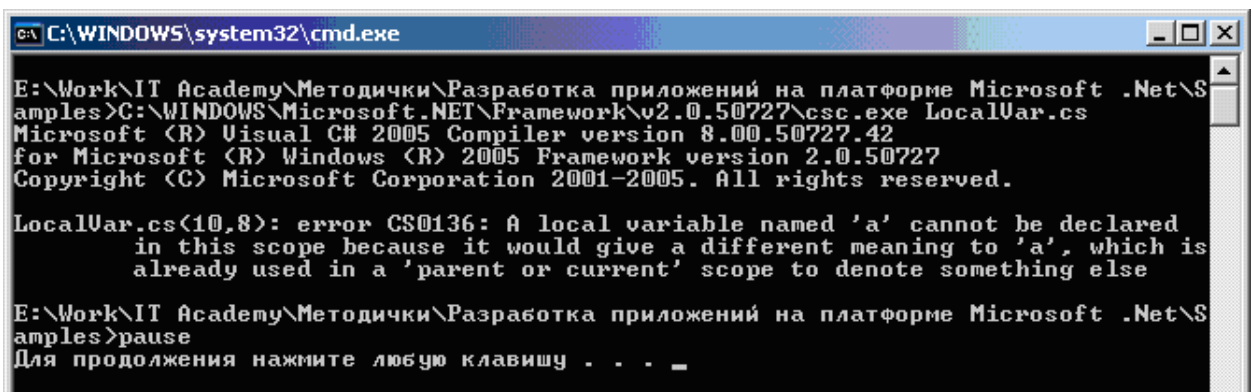


Рисунок 2 – Результат выполнения

Локальные переменные (без ошибки):

```
class Variables
{
    public static void Main(string[] args)
    {
        //
        int a;
        {
            int b;
```

```

    }
    {
        int a;
        int b;
    }
}

```

Компиляция этого кода хоть и проходит с предупреждениями о неиспользовании объявленных переменных, но без ошибок.

```

C:\WINDOWS\system32\cmd.exe
E:\Work\IT Academy\Методички\Разработка приложений на платформе Microsoft .Net\Samples>C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\csc.exe LocalVar.cs
Microsoft (R) Visual C# 2005 Compiler version 8.00.50727.42
for Microsoft (R) Windows (R) 2005 Framework version 2.0.50727
Copyright (C) Microsoft Corporation 2001-2005. All rights reserved.

LocalVar.cs(7,8): warning CS0168: The variable 'b' is declared but never used
LocalVar.cs(10,8): warning CS0168: The variable 'a' is declared but never used
LocalVar.cs(11,8): warning CS0168: The variable 'b' is declared but never used

E:\Work\IT Academy\Методички\Разработка приложений на платформе Microsoft .Net\Samples>pause
Для продолжения нажмите любую клавишу . . . _

```

Рисунок 3 – Результат выполнения

Средства ввода и вывода

Для организации консольного ввода и вывода используем статические методы класса System.Console.

```

Console.WriteLine("Hello, World!");
Console.Write("Hello, World!");

```

Это наиболее простой вариант вывода. Данные методы позволяют также осуществлять форматированный параметризованный вывод.

```

Console.WriteLine("{0}, {1} {2}", "Hello", "World", "!");

```

При помощи параметров и осуществляется форматирование строки.

Общий вид параметра {N,M:F<R>}, где:

N – номер параметра (начинаются с нуля);

M – ширина поля;

F – формат вывода;

R – количество выводимых разрядов.

Форматы вывода следующие:

C - форматирование числа как денежной суммы;

D - Целое число;

E - Вещественное число в виде 1e+3;

F - Вещественное число в виде 123.456;

G - Вещественное число в наиболее компактном формате;

N - Вещественное число в виде 123,456,789.5;

X - Целое число в шестнадцатеричном виде.

Средства ввода

C# позволяет осуществлять консольный ввод как одного символа, так и целой строки.

```
int i = Console.Read();
```

```
string str = Console.ReadLine();
```

После получения входной строки ее несложно преобразовать в необходимый тип.

```
string str = Console.ReadLine();
```

```
int i = Int32.Parse(str);
```

```
float f = Float.Parse(str);
```

```
double d = Double.Parse(str);
```

Массивы

Все массивы наследуются от базового класса System.Array.

Поддерживается три вида массивов:

- одномерные массивы;
- многомерные массивы;
- вложенные (jagged) массивы.

Рассмотрим работу с массивами на базе примера.

```
class Arrays  
{
```

```

static void Main(string[] args)
{
    int[] a = new int[3];
    int[] b = { 1,2,3 };
    int[] d; // d = null
    d = b; // d и c – это один и тот же массив!
    a = (int[])b.Clone(); //копируем массив
    int[,] e = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } }; //прямоугольный массив
    int[][] f = new int[2][]; //вложенный массив
    f[0] = new int[3] { 0, 1, 2 };
    f[1] = new int[2] { 3, 4 }; //строки вложенного массива могут быть разной
длины
}
}

```

Для получения размерностей массива можно использовать следующие методы:

Length – возвращает количество элементов в массиве;

Rank – возвращает количество измерений массива;

GetLength(int i) – возвращает размер i-го измерения.

Структура

```

public struct ArrayMas
{
    public string KBK;
    public double PaySum;
    public double PaySumUsed;
    public DateTime Date;
    public double Rate;
}

public struct UserMas
{
    public ArrayMas[] MAS1;
    public ArrayMas[] MAS2;
    public ArrayMas[] MAS3;
    public ArrayMas[] MAS4;
    public int NNN1;
    public int NNN2;
    public int NNN3;
}

```

```
public int NNN4;  
}
```

Условные операторы и операторы циклов

В С# имеются следующие условные операторы: if, switch; и операторы циклов: for, while, do, foreach. Оператор безусловного перехода goto не рассматривается как морально устаревший.

Условный оператор IF

```
if (a>b)  
{  
    Console.WriteLine("a>b");  
}  
else  
{  
    Console.WriteLine("a<=b");  
};  
if (a==b) { Console.WriteLine("a=b");}; //условие равенства в С# обозначается двумя  
знаками равенства
```

Условный оператор SWITCH

```
int a=10;  
switch (a)  
{  
    case 0: ...; break;  
    case 1: ...; break;  
    default: ...; break;  
}
```

Операторы, указанные после default, выполняются в том случае, если ни одно из вышестоящих условий не было выполнено.

Цикл FOR

```
int[] a={ 10, 20, 30, 40 };  
for (int i=0;i<a.Length;i++)  
{  
    System.Console.WriteLine(a[i].ToString());  
}
```

Переменная-счетчик, объявляемая в заголовке цикла, существует только в теле цикла и является локальной переменной

Цикл WHILE


```
int a=5678;
while (a>1)
{
    a /= 10;
}
```

Цикл DO

```
int a=5678;
do
{
    a /= 10;
} while (a>1);
```

Цикл FOREACH

```
int[] a = { 10, 20, 30, 40 };
foreach (int i in a) { System.Console.WriteLine(i); }
```

Цикл `foreach` предназначен для работы с коллекциями и является аналогом цикла `for`.

Классы

Классы в C# имеют тот же смысл, что и в языках C++ и Java. Формально класс описывается следующим образом:

```
модификатор-доступа class Имя-класса {
    ... описание данных и методов ...
}
```

Модификаторы доступа определяют поле видимости данного класса. Для классов предназначены два модификатора доступа:

`public` – класс доступен для других компонент;
`internal` – класс видим только внутри данной компоненты (приложения).

По умолчанию применяется модификатор `internal`.

Членами класса могут быть:

- Константы

- Поля (field)
- Конструкторы (в том числе без параметров)
- Деструктор
- Методы (статические и экземплярные)
- Свойства (property)
- Индексаторы (свойства с параметрами)
- События (event)
- Вложенные типы

Модификаторы доступа также указываются и перед полями и методами класса:

- private (умолчание)
- public
- protected
- internal - поле видимо только внутри компоненты
- protected internal.

Методы

Формальное описание метода:

модификатор-доступа другие-модификаторы

возвращаемый-тип имя-метода(описание-параметров)

{ тело-метода }

Модификаторы доступа:

public, private, protected - аналогично C++

internal – метод видим только внутри компоненты

Другие модификаторы:

static – метод класса (аналогично C++)

virtual, override, new – управление виртуальными методами.

Конструктор – это метод, которые выделяет память под объект и инициализирует выделенную область.

```
Variables var = new Variables();
```

new выделяет память, а Variables() инициализирует ее.

Конструктор может быть с параметрами и без. Конструктор без параметров называется конструктором по умолчанию. Если в классе нет ни одного конструктора, то компилятор сам создает конструктор по умолчанию.

В классе может быть определено несколько конструкторов с различными списками параметров. Если в классе определен хотя бы один конструктор, конструктор по умолчанию не создается. Конструктор по умолчанию для структуры всегда создается вне зависимости от наличия других конструкторов. При его использовании все поля структуры оказываются нулевыми. Конструктор для структуры должен явно инициализировать все поля данных и не может быть protected. Статический конструктор – это конструктор класса. Он инициализирует статические поля класса.

Свойство – пара методов со специальными именами. Метод set() вызывается при задании значения свойства, метод get() – при получении значения свойства. Обращение к свойству выглядит как обращение к полю данных, но транслируется в вызов одного из двух методов.

Определяя в свойстве только один из двух методов, получаем свойства только для чтения и для записи. Каждый из методов может иметь модификатор доступа.

Таким образом, .NET является удобной и развитой RAD-средой для разработки приложений различного типа, в том числе и распределенных систем.

Microsoft (Windows) Azure

Microsoft (Windows) Azure – название облачной платформы Microsoft. Предоставляет возможность разработки и выполнения приложений и хранения данных на серверах, расположенных в распределённых дата-

центрах. Первоначально называлась Windows Azure. В 2014 году платформа была переименована в Microsoft Azure.

Microsoft Azure полностью реализует две облачные модели — платформы как сервиса (Platform as a Service, PaaS) и инфраструктуры как сервиса (Infrastructure as a Service, IaaS). Работоспособность платформы Windows Azure обеспечивает сеть глобальных дата-центров Microsoft.

Основные особенности данной модели:

- оплата только потребленных ресурсов;
- общая, многопоточная структура вычислений;
- абстракция от инфраструктуры.

В основе работы Microsoft Azure лежит запуск виртуальной машины для каждого экземпляра приложения. Разработчик определяет необходимый объём для хранения данных и требуемые вычислительные мощности (количество виртуальных машин), после чего платформа предоставляет соответствующие ресурсы. Когда первоначальные потребности в ресурсах изменяются, в соответствии с новым запросом заказчика платформа выделяет под приложение дополнительные или сокращает неиспользуемые ресурсы дата-центра.

Microsoft Azure как PaaS обеспечит не только все базовые функции операционной системы, но и дополнительные: выделение ресурсов по требованию для неограниченного масштабирования, автоматическую синхронную репликацию данных для повышения отказоустойчивости, обработку отказов инфраструктуры для обеспечения постоянной доступности и многое другое.

Microsoft Azure также реализует другой тип сервиса — инфраструктуру как сервис. Модель предоставления инфраструктуры (аппаратных ресурсов) реализует возможность аренды таких ресурсов, как серверы, устройства хранения данных и сетевое оборудование. Управление всей инфраструктурой осуществляется поставщиком, потребитель управляет только операционной системой и установленными приложениями. Такие сервисы оплачиваются по

фактическому использованию и позволяют увеличивать или уменьшать объём инфраструктуры через специальный портал, предоставляемый поставщиками. В данной сервисной модели могут быть запущены практически любые приложения, установленные на стандартные образы ОС.

В доступной сразу галерее образов доступны образы следующих операционных систем: Windows Server (2008, 2012, Technical Preview), CoreOS, Ubuntu Server, CentOS, openSUSE, SUSE Linux Enterprise Server, Oracle Linux и др.

В 2013 году было представлено новое хранилище образцов виртуальных машин — VM Depot. VM Depot — это проект для сообщества Windows Azure, запущенный командой Microsoft Open Technologies, Inc, ответственной за открытые технологии внутри Microsoft. Содержимое портала, а также настроенные для разных задач виртуальные машины, будут создаваться и публиковаться силами сообщества.

Microsoft Azure состоит из:

Compute – компонент, реализующий вычисления на платформе Windows Azure.

Storage – компонент хранилища предоставляет масштабируемое хранилище. Хранилища не имеет возможности использовать реляционную модель и является альтернативной, "облачной" версией SQL Server.

Fabric – Windows Azure Fabric по своему назначению является «контролёром» и ядром платформы, выполняя функции мониторинга в реальном времени, обеспечения отказоустойчивости, выделения мощностей, развертывания серверов, виртуальных машин и приложений, балансировки нагрузки и управления оборудованием.

Практически все сервисы Microsoft Azure имеют API, построенное на REST, что позволяет разработчикам использовать «облачные» сервисы с любой операционной системы, устройства и платформы.

Microsoft Azure была признана Compuware самой быстрой «облачной» платформой. Также бенчмарк Microsoft Azure показал высокую

производительность для масштабных вычислений - с результатами в 151,3 ТФлопс на 8064 ядрах с 90,2 процентной эффективностью. Аналитическая компания Nasuni представила очередное исследование провайдеров облачных сервисов хранения данных. Согласно этому отчёту, платформа Microsoft Azure является лидером в тестах производительности при записи и чтении данных из облака, доступности данных и минимальному числу ошибок (0%). В 2014 году Nasuni выпускает очередной отчёт с результатами тестирования облачных хранилищ Amazon, Google, HP, Microsoft и Rackspace.

Лабораторная работа №1. Создание консольных приложений на C#

Цель: закрепление знаний по C#, получение навыков создания консольных приложений.

Постановка задачи: Требуется разработать приложение согласно варианту задания.

Вся информация должна храниться в массивах. Рекомендуется объекты реализовать в виде классов.

Исходные данные для выполнения задания.

Вариант 1

Создаваемое приложение должно для каждого клиента банка хранить следующие сведения: Ф.И.О.; Возраст; Место работы; Номера счетов.

На каждом счете хранится информация о текущем балансе и история прихода, расхода. Для каждого клиента может быть создано неограниченное количество счетов. С каждым счетом можно производить следующие действия: открытие, закрытие, вклад денег, снятие денег, просмотр баланса, просмотр истории.

Вариант 2

Создаваемое приложение должно по каждому налогоплательщику хранить следующие сведения: Ф.И.О.; ИНН; Начисленные и уплаченные суммы налога.

На каждом лицевом счете налогоплательщика хранится информация о типе налога, текущей суммы начислений, ее уплаты и история прихода и расхода. Для каждого налогоплательщика может быть создано неограниченное количество операций. С каждой начисленной суммы можно произвести следующие операции: создание, редактирование суммы, удаление ошибочных начислений, просмотр истории.

Вариант 3

Создаваемое приложение должно по каждому клиенту автосервиса хранить следующие сведения: Ф.И.О.; Марка и модель автомобиля; история обращения в автосервис.

При каждом обращении в автосервис вводятся следующие параметры: дата обращения, причина обращения, стоимость ремонта, дата передаче автомобиля. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории обращения в автосервис.

Вариант 4

Создаваемое приложение должно по каждому посетителю библиотеки хранить следующие сведения: Ф.И.О.; номер читательского билет; дата рождения; пол; история обращения в библиотеку.

При каждом обращении в библиотеку вводятся следующие параметры: дата обращения, название книги, уникальный код книги, автор книги, дата возврата. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории обращения в библиотеку как по каждому посетителю, так и по всем.

Вариант 5

Создаваемое приложение должно по каждому клиенту оптового магазина для индивидуальных предпринимателей и юридических лиц хранить следующие сведения: Ф.И.О. или название организации; ИНН; КПП; ОГРН; история покупок.

При каждой покупке вводятся следующие параметры: дата покупки, название приобретаемого товара, штрих-код товара, количество, сумма сделки. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории покупок как по каждому покупателю, так и по всем.

Вариант 6

Создаваемое приложение должно по каждому клиенту автостоянки хранить следующие сведения: Ф.И.О.; марка и модель автомобиля; государственный знак автотранспортного средства; история обращения.

При каждой использовании услуг автостоянки вводятся следующие параметры: дата парковки, место парковки, дата убытия, сумма за парковку. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории обращения клиента за услугами автостоянки как по каждому, так и по всем клиентам.

Вариант 7

Создаваемое приложение должно по каждому студенту хранить следующие сведения: Ф.И.О.; номер группы; номер студенческого билета; история обращения.

При каждой попытке сдаче курсового проекта вводятся следующие параметры: дата совершения попытки, процент выполнения, перечень замечаний, планируемая дата следующей попытки сдачи. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории обращения студента.

Вариант 8

Создаваемое приложение должно по каждому сотруднику организации хранить следующие сведения: Ф.И.О.; табельный номер; дата рождения, отдел, должность; история посещения.

При каждом входе в здание организации вводятся следующие параметры: дата и время входа, номер турникета, через который совершен вход, дата и время выхода. По каждому из параметров необходимо производить следующие операции: создание, редактирование, удаление. Предусмотреть просмотр всей истории посещения работы сотрудником.

Предусмотреть возможность выбора сотрудников, которые опаздывают на работу.

Продолжительность: 4 часа.

Методические указания для выполнения задания лабораторной работы

Для создания приложений можно использовать бесплатную версию Microsoft Visual Studio 2012.

Ход выполнения работы

1. Внимательное изучение теоретического материала.
2. Разработка приложения.
3. Тестирование.
4. Демонстрация преподавателю.
5. Подготовка краткого отчета с указанием основных этапов работы и результатов работы приложения.
6. Подготовка ответов на контрольные вопросы.
7. Защита лабораторной работы (включает объяснение основных этапов разработки приложения, ответы на контрольные вопросы).

Контрольные вопросы

1. Охарактеризовать элементы языка C#.
2. Пояснить этапы разработки консольного приложения.
3. Какие языки можно использовать при разработке консольного приложений в Microsoft Visual Studio 2012?

Лабораторная работа №2. Создание Windows-приложения на C#.

Регистрация на портале Microsoft Azure.

Цель: получение навыков создания интерфейса для приложения, созданного в лабораторной работе №1.

Постановка задачи: Организовать ввод сведений о новых клиентах и счетах реализовать через дополнительные формы.

Исходные данные для выполнения задания.

Приложение, созданное в предыдущей лабораторной работе.

Продолжительность: 4 часа.

Методические указания для выполнения задания лабораторной работы

Все типы WindowsForms находятся в пространстве имен

System.Windows.Forms.

Внешний вид Windows-приложения на стадии разработки в Microsoft Visual Studio 2012 представлен на рис. 4.

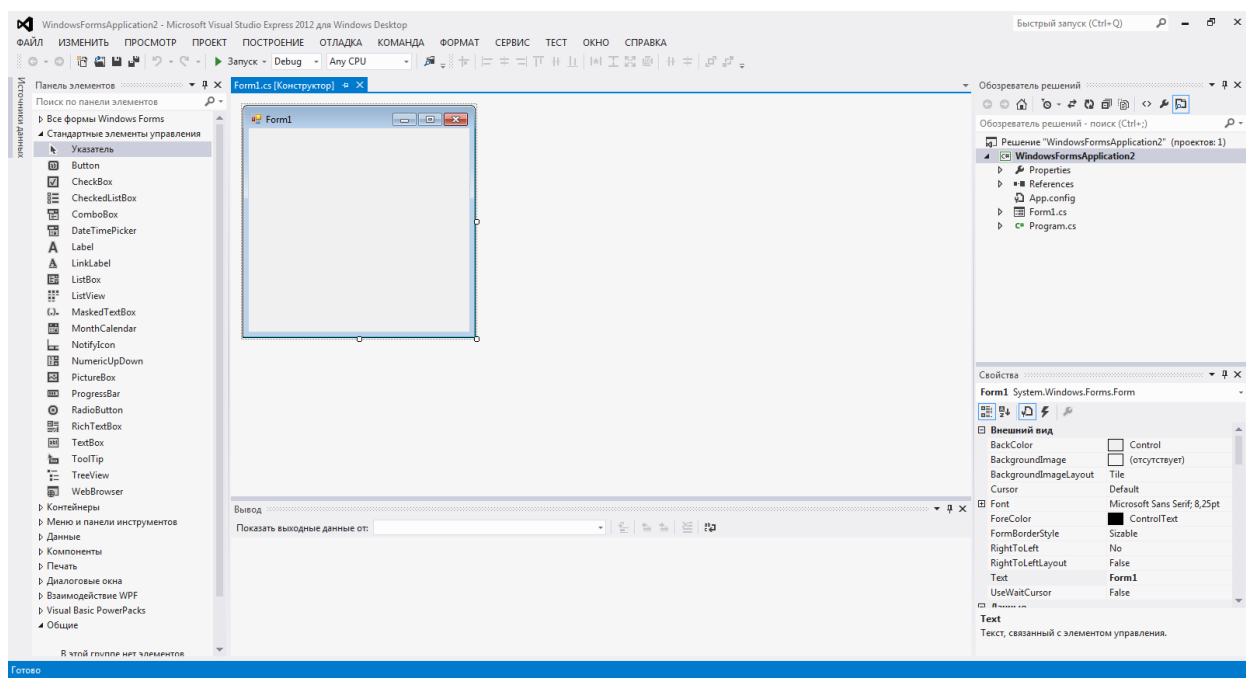


Рисунок 4 – Инструментальная среда

Основной класс в приложении – это класс **Form**. В последних версиях .Net появилась возможность разбивать класс по нескольким файлам. Visual Studio 2012 разбивает описание класса на два файла. Первый называется <имя формы>.cs. В нем располагается пользовательский код по обработке разных событий формы. Второй – < имя формы>.Designer.cs. В нем располагается код формы, сгенерированной самой Visual Studio. Механизм разбиения классов можно применять и вручную.

```
partial class PartCl
{
    int a;
    int b;
}
partial class PartCl
{
    public PartCl(int p1, int p2)
    {
        a=p1;
        b=p2;
    }
}
```

Каждую из частей можно расположить в отдельном файле.

Свойства форм и обработчики событий формы отображаются в окне «Свойства».

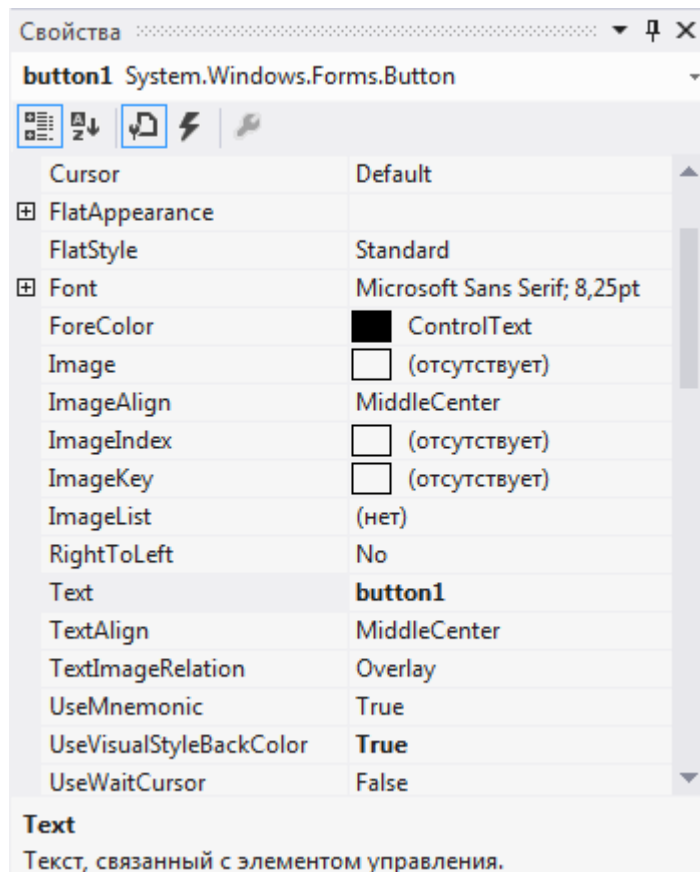


Рисунок 5 – Окно свойств

Для создания в проекте новой формы необходимо выполнить действия, согласно рис. 6.

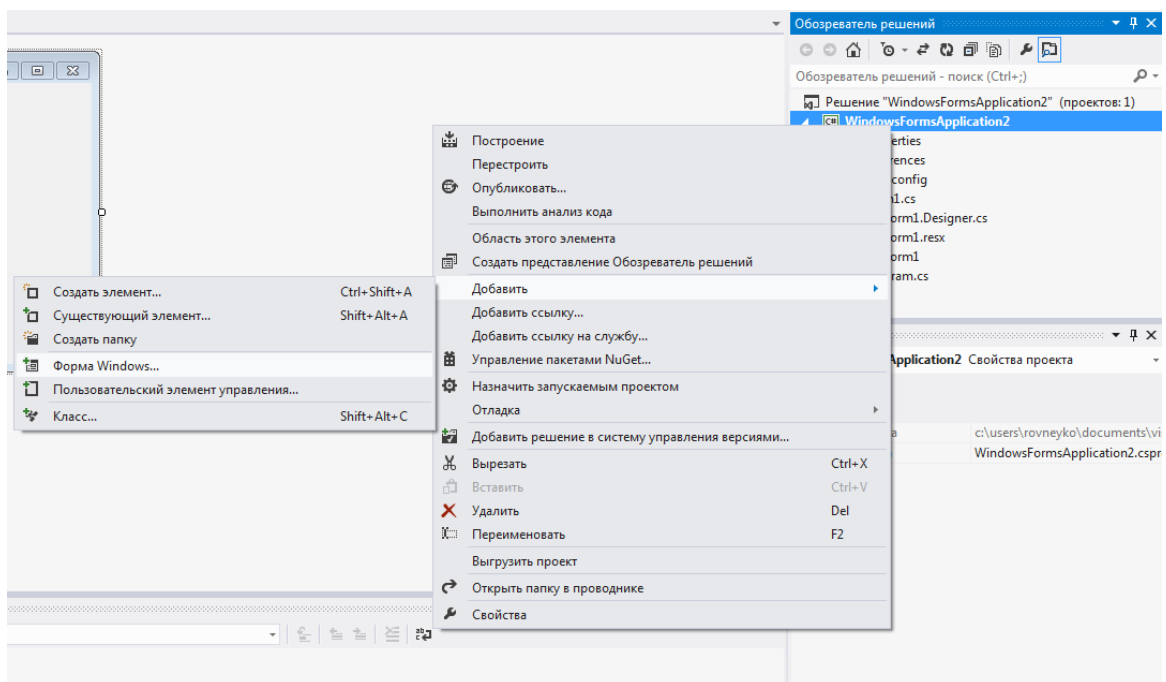


Рисунок 6 – Создание новой формы

За время жизненного цикла формы происходят следующие события с ней: `Load`; `GotFocus`; `Activated`; `Closing`; `Closed`; `Deactivate`; `LostFocus`; `Dispose`.

Отображение главной формы происходит при запуске программы. Остальные формы можно вызвать при помощи двух методов класса `Form`: `Show` и `ShowModal`. Метод `Show` отображает обычную форму, `ShowModal` отображает модальную форму.

Для вывода каких-либо сообщений можно использовать метод `Show` класса `MessageBox` из пространства имен `System.Windows.Forms`.
`MessageBox.Show("This is a test", "Title", MessageBoxButtons.OK);`

Используя этот класс, можно организовать простую интерактивность с пользователем.

```
if(MessageBox.Show("Press Yes or No?", "Title", MessageBoxButtons.YesNo)
== DialogResult.Yes) {...};
```

Компоненты панели управления для Windows-приложений представлены ниже на рис. 7. Имея опыт программирования в визуальных средах проектирования, можно из названий догадаться о предназначении того или иного компонента.

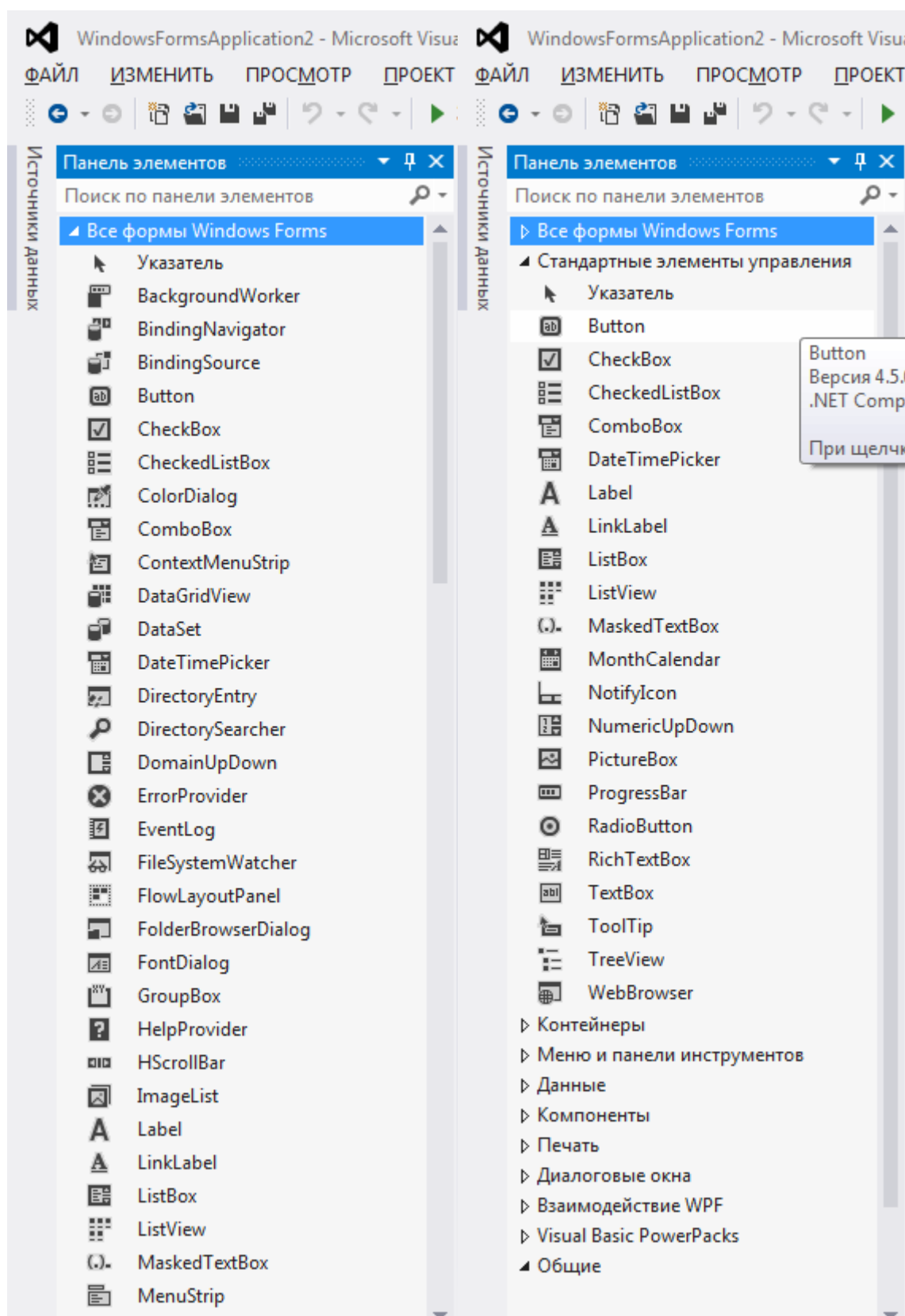


Рисунок 7 – Панели инструментов

Свойства полей данных формы могут быть считаны или изменены в процессе выполнения программы. Все свойства или установки обработчиков

событий, сделанные программистом в дизайнера формы или в окне Properties, записываются в код метода **private void InitializeComponent()**.

Все обработчики событий формы являются ее методами. Любому обработчику событий передаются два параметра: объект, вызвавший событие и параметры этого события.

```
private void button1_Click(object sender, EventArgs e)
```

Для создания обработчика выбора пункта меню необходимо дважды щелкнуть на этом пункте меню в дизайнера формы.

```
private void toolStripMenuItem2_Click(object sender, EventArgs e)
```

Регистрация на портале Microsoft Azure.

Для использования системы Microsoft Azure необходимо предварительно зарегистрироваться на сайте Майкрософт и получить доступ к **DreamSpark** (подписка на 1 год бесплатного использования Microsoft Azure).

В случае, если у Вас нет учетной записи на сайте Майкрософт, то для регистрации зайдите на сайт <https://www.dreamspark.com/> и перейдите в меню «Учащиеся» → «Войти в систему» → «Зарегистрироваться». Заполните все необходимые поля и нажмите «Создать учетную запись». После того, как регистрация пройдет успешно авторизуйтесь на сайте <https://www.dreamspark.com/> через меню «Учащиеся» → «Войти в систему».

Далее необходимо зайти на сайт <http://www.dreamspark.ru/> и нажать на кнопку «Получить доступ со сканом студбилета» (рис. 8).

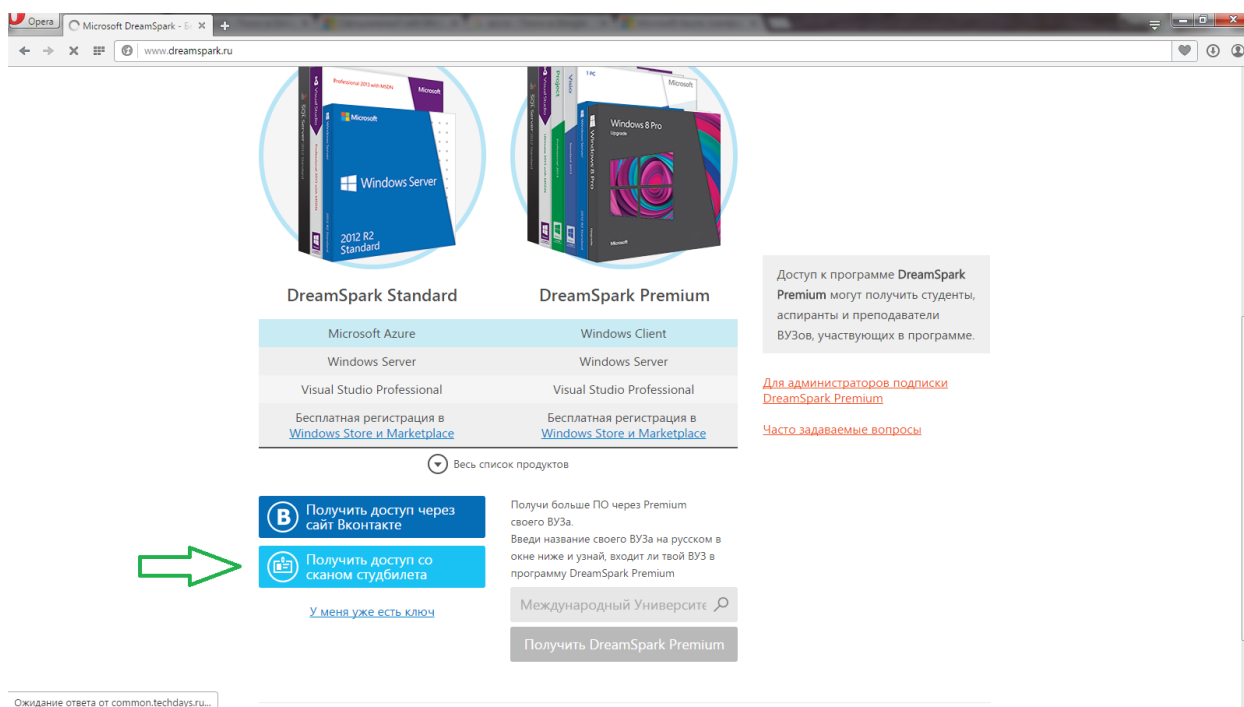


Рисунок 8 – Получение доступа к DreamSpark

Далее заполняете все требуемые поля и в качестве «Скан документа» прикладываете отсканированную копию **действующего** студенческого билета. После заполнения всех полей и нажатия на кнопку «Зарегистрироваться» Ваша заявка будет в течении 2-4 дней рассмотрена.

После того, как Ваша заявка будет рассмотрена, на указанный адрес электронной почты придет сообщение со следующим содержанием.

Спасибо за Ваш интерес к программе DreamSpark!

Вы успешно прошли процесс верификации и ниже вы найдете ссылку, по которой вам надо зайти для получения ПО в рамках программы DreamSpark. Напоминаем, что для получения ПО вам понадобится Microsoft Account (ранее Live ID):

<https://www.dreamspark.com/auth/CodeVerify.aspx?key=6MATC-F47RR-JR2D4-X6Y3K-WPDJZ>

Вы также можете скопировать ключ доступа в соответствующую форму: 6MATC-F47RR-JR2D4-X6Y3K-WPDJZ на сайте

После того, как вы зайдете по этой ссылке на сайт и зарегистрируетесь на нем с Microsoft Account, вы сможете скачивать лицензионное ПО используя только Microsoft Account. Повторного ввода кода не требуется.

Для вашего удобства мы подготовили список наиболее актуальных продуктов для разработчика и дизайнера (рабочий стол разработчика). Пройдя по ссылкам ниже вы можете сразу начать скачивать продукт, но предварительно необходимо пройти процесс регистрации описанный выше.

Visual Studio Professional 2013 Preview

Visual Studio Professional 2012

Kinect for Windows SDK

SQL Server 2012

Windows Phone 8 SDK

Windows Phone 7 SDK

Windows Azure Mobile Services SDK
Microsoft XNA Game Studio 4
Microsoft Visual Studio Express 2013 Preview for Windows

Также, имея подписку Dreamspark, вы можете получить бесплатный аккаунт разработчика в магазин приложений Windows и Windows Phone. Не упустите свой шанс заработать на создании приложений для ОС Windows и Windows Phone!

Чтобы своевременно получать информацию о всех предложениях Microsoft для студентов, мы рекомендуем подписаться на ежемесячную новостную рассылку «Студенческий вестник Microsoft» !

Напоминаем, что ПО, полученное вами в рамках DreamSpark, может использоваться исключительно в целях обучения.

Удачного программирования!

© Отдел академических программ, Microsoft Россия
Microsoft - студенческая территория: учись, общайся, развлекайся! www.ms-student.ru

После получения кода необходимо зайти на сайт <http://www.dreamspark.ru/> и нажать на ссылку «У меня уже есть ключ» (при необходимости авторизуемся). Далее следуем инструкции на сайте.

Внимание! Ввод данных по банковской карте НЕ ВЫПОЛНЯТЬ. В случае если система запросит информацию по банковской карте, то обратитесь за консультацией к преподавателю.

После того, как полученный код активирован зайдите в свой аккаунт DreamSpark на сайте <https://www.dreamspark.com> (рис. 9).

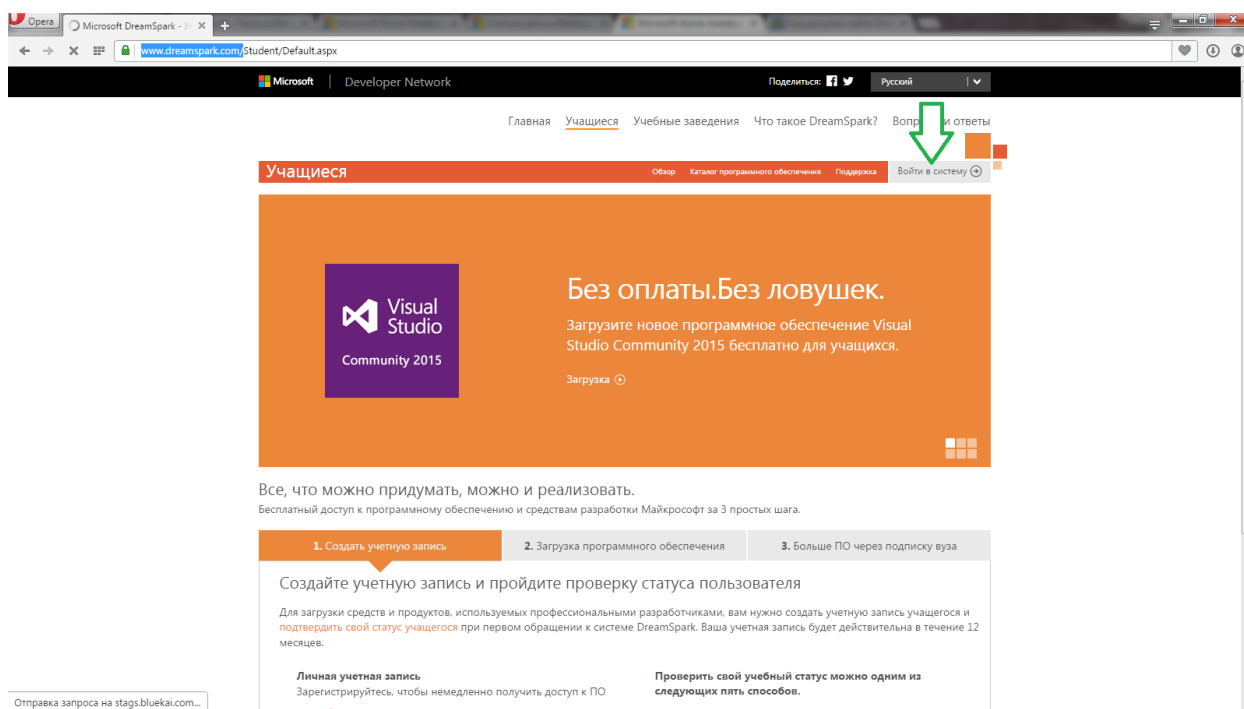


Рисунок 9 – Вход на аккаунт DreamSpark

После входа в свой аккаунт перейдем в меню «Каталог программного обеспечения», на полученной странице в разделе «Серверы и приложения» выберем продукт «Microsoft Azure для DreamSpark» (рис. 10).

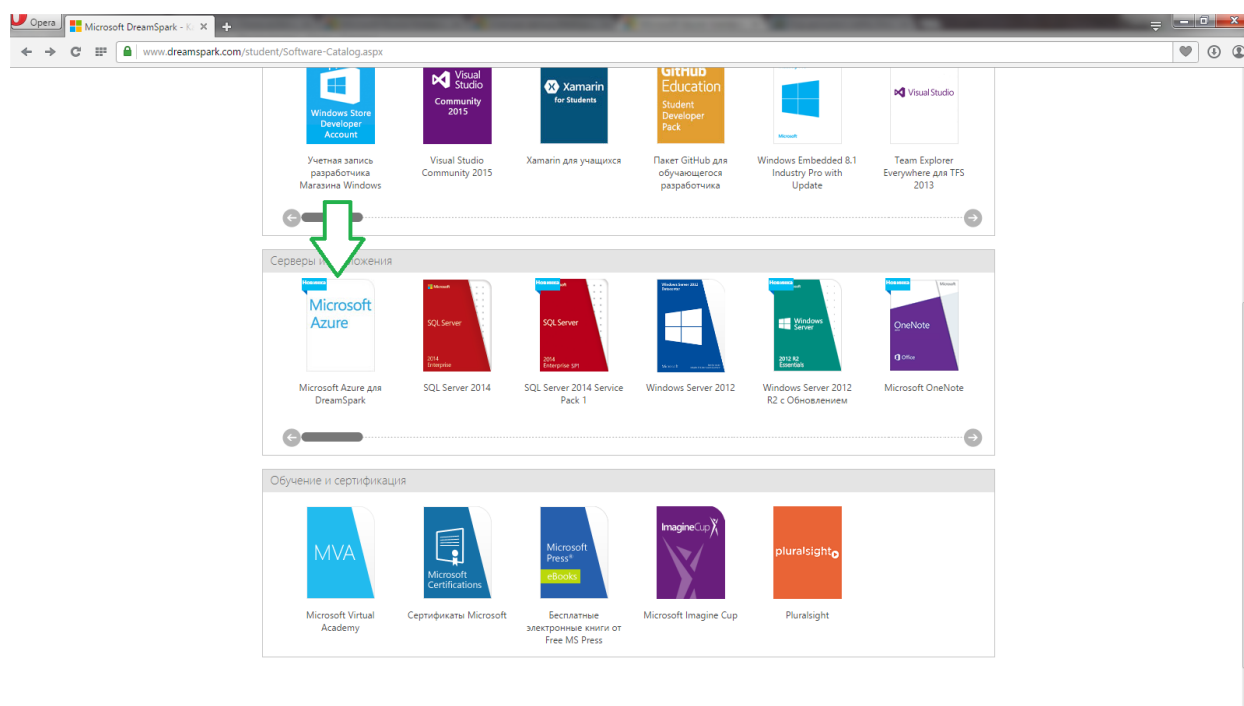
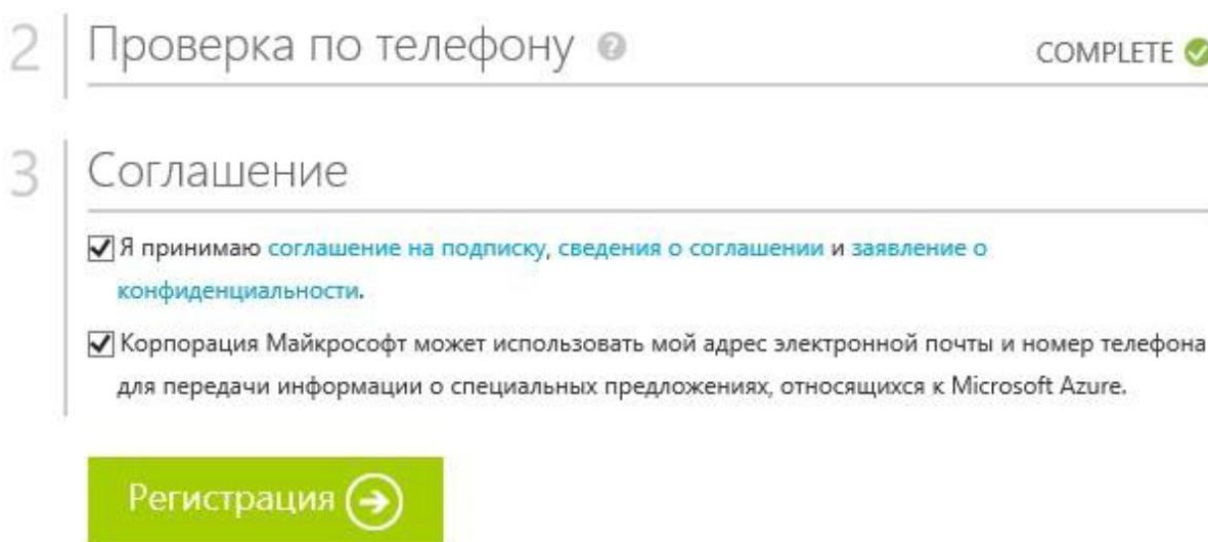


Рисунок 10 – Microsoft Azure для DreamSpark

При клике на Microsoft Azure для DreamSpark открывается следующая страница, на ней кликаем на кнопку «Зарегистрироваться сейчас». После нажатия на эту кнопку Вас перебросит на страницу, где необходимо проверить и дополнить свои личные данные: указать ФИО, страну (Россия), а также подтвердить номер телефона. На него вам придет код в SMS-сообщении, по которому и будет произведена проверка (рис. 11).



The screenshot shows a registration progress bar with two steps. Step 2, 'Проверка по телефону' (Phone verification), is marked as 'COMPLETE' with a green checkmark. Step 3, 'Соглашение' (Agreement), is the current active step. It contains two checked checkboxes: 'Я принимаю соглашение на подписку, сведения о соглашении и заявление о конфиденциальности.' and 'Корпорация Майкрософт может использовать мой адрес электронной почты и номер телефона для передачи информации о специальных предложениях, относящихся к Microsoft Azure.' Below the checkboxes is a green button labeled 'Регистрация' with a right-pointing arrow.

Рисунок 11 – Подтверждение личности

После нажатия на кнопку «Регистрация» откроется страница проверка введенных данных. Когда все будет подготовлено, получим данную кнопку (при условии, что не закрывали предыдущую страницу):

Ваша подписка готова.

Начать управлять моей службой >

После нажатия на «Начать управлять моей службой» перейдем на портал Microsoft Azure.

На этой процедуре регистрация завершена, далее входим на <https://portal.azure.com/>, используя указанный ранее Microsoft Account.

Ход выполнения работы

1. Внимательное изучение теоретического материала.
2. Разработка приложения.
3. Тестирование.
4. Регистрация на портале Microsoft Azure.
5. Демонстрация преподавателю.
6. Подготовка краткого отчета с указанием основных этапов работы и результатов работы приложения.
7. Подготовка ответов на контрольные вопросы.
8. Защита лабораторной работы (включает объяснение основных этапов разработки приложения, ответы на контрольные вопросы).

Контрольные вопросы

1. Охарактеризовать этапы жизненного цикла формы.
2. Пояснить основные элементы инструментальной среды разработки.
3. Какие языки можно использовать при разработке приложений в Microsoft Visual Studio 2012?
4. Как запрограммировать меню при разработке приложения?

Лабораторная работа №3. Создание БД MySQL и использованием Microsoft Azure

Цель: Получения навыков создание БД с использованием Microsoft Azure.

Постановка задачи: Требуется создать Бд MySQL в Microsoft Azure; добавить в нее таблицы согласно варианту задания и их наполнить данными. Написать SQL запросы согласно устному заданию преподавателя.

Исходные данные для выполнения задания.

Варианты заданий согласно выданному преподавателем варианту из Лабораторной работы №1.

Продолжительность: 4 часа.

Методические указания для выполнения задания лабораторной работы

Для того, чтобы создания БД MySQL необходимо перейти на портал Microsoft Azure. Далее в меню нажать «Создать» → «Данные+хранилища» → «База данных MySQL». Откроется окно создания базы данных (рис. 12).

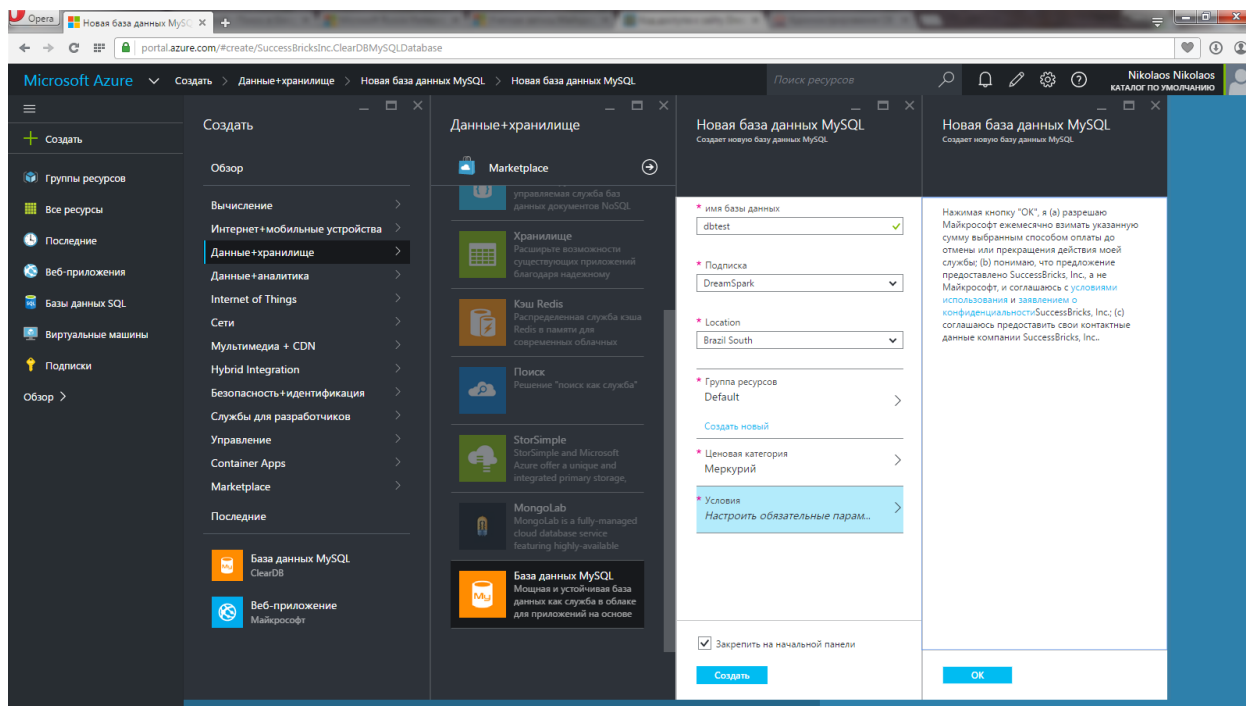


Рисунок 12 – Создание БД MySQL

После того, как БД создана, можно посмотреть ее свойства и параметры, необходимые для подключения из внешнего источника (рис. 13).

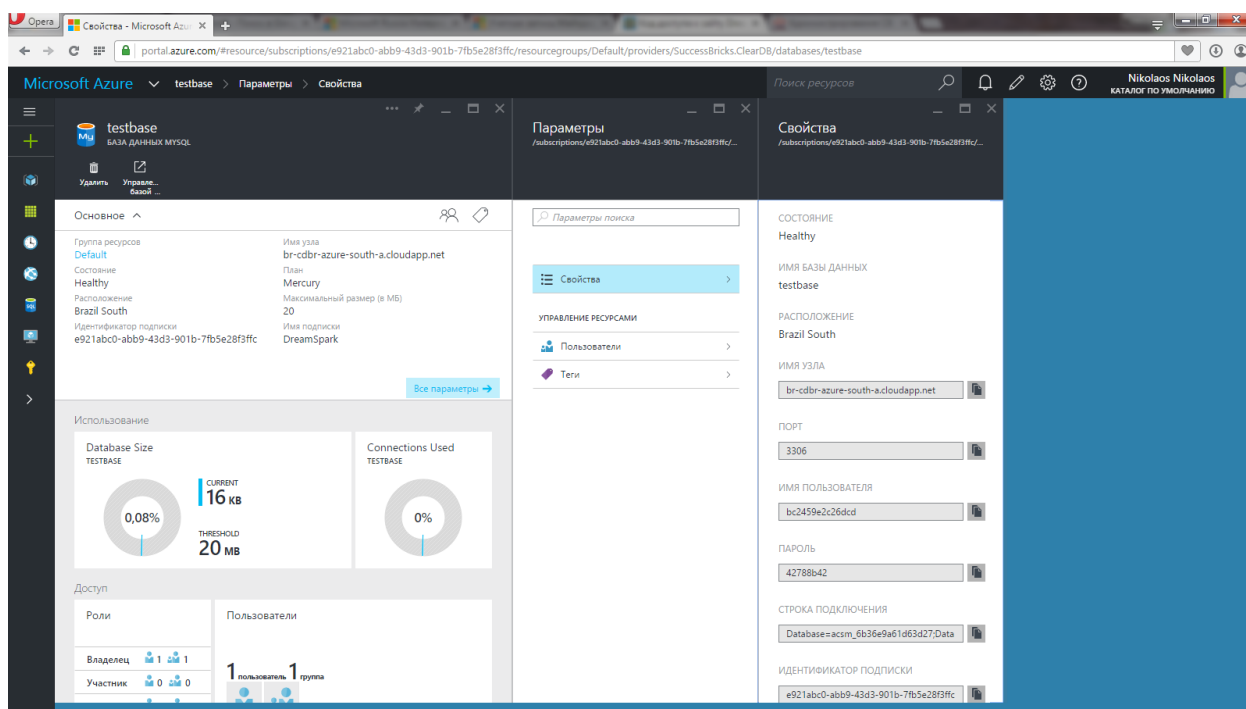


Рисунок 13 – Параметры БД

Для создания и наполнения таблиц БД необходимо использовать средства для администрирования БД MySQL. В лабораторной работе рассмотрим использование программного продукта MySQL Workbench 6.3 CE.

Для начала работы в MySQL Workbench 6.3 необходимо выполнить соединение с БД, расположенной в Microsoft Azure (рис. 14). Все необходимые параметры для настройки можно найти в свойствах БД (рис. 13).

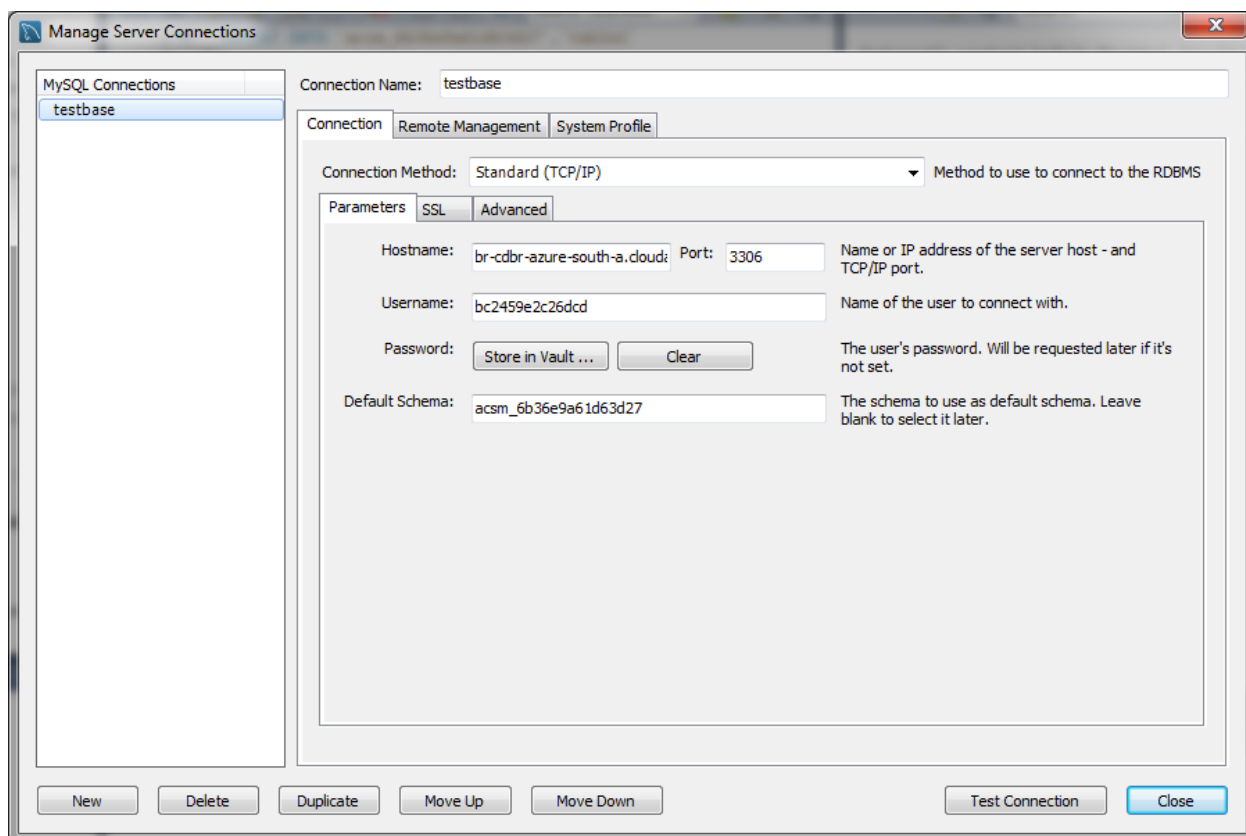


Рисунок 13 – Настройка соединения с БД, расположенной в Microsoft Azure

Ход выполнения работы

1. Внимательное изучение теоретического материала.
2. Разработка БД.
3. Написание SQL-запросов.
4. Демонстрация преподавателю.
5. Подготовка краткого отчета с указанием основных этапов работы и результатов работы.

6. Подготовка ответов на контрольные вопросы.
7. Защита лабораторной работы (включает объяснение основных этапов разработки БД, ответы на контрольные вопросы).

Контрольные вопросы

1. Что представляет из себя Microsoft Azure? Что он позволяет делать?
2. Пояснить этапы разработки БД.
3. Какие функции выполняет MySQL Workbench 6.3?

Лабораторная работа №4. Создание Web-приложения. Размещение Web-приложения в Microsoft Azure

Цель: Получения навыков Web-приложения в среде Microsoft Visual Studio.

Постановка задачи: Требуется создать Web-приложение с возможностью подключения к БД, созданной в Лабораторной работе №3. Разместить Web-приложение в Microsoft Azure.

Приложение должно выполнять функции добавления, удаления, редактирования данных и просмотра аналитических отчетов.

Исходные данные для выполнения задания.

Варианты заданий согласно выданному преподавателем варианту из Лабораторной работы №1.

Продолжительность: 8 часов.

Методические указания для выполнения задания лабораторной работы

Для выполнения лабораторной работы потребуется Microsoft Visual Studio 2012 Express for Web, MySQL Workbench 6.3, MySQL Connector Net 6.9.7 и Microsoft Azure.

Для создания Web-проекта необходимо в Microsoft Visual Studio 2012 Express for Web зайти в меню «FILE → New Project». В новом окне выбрать во вкладке «Templates» язык программирования «Visual c# → Web» и после «ASP.NET Web Forms Application». Указать путь сохранения проекта и нажать кнопку «ОК» (рис. 14).

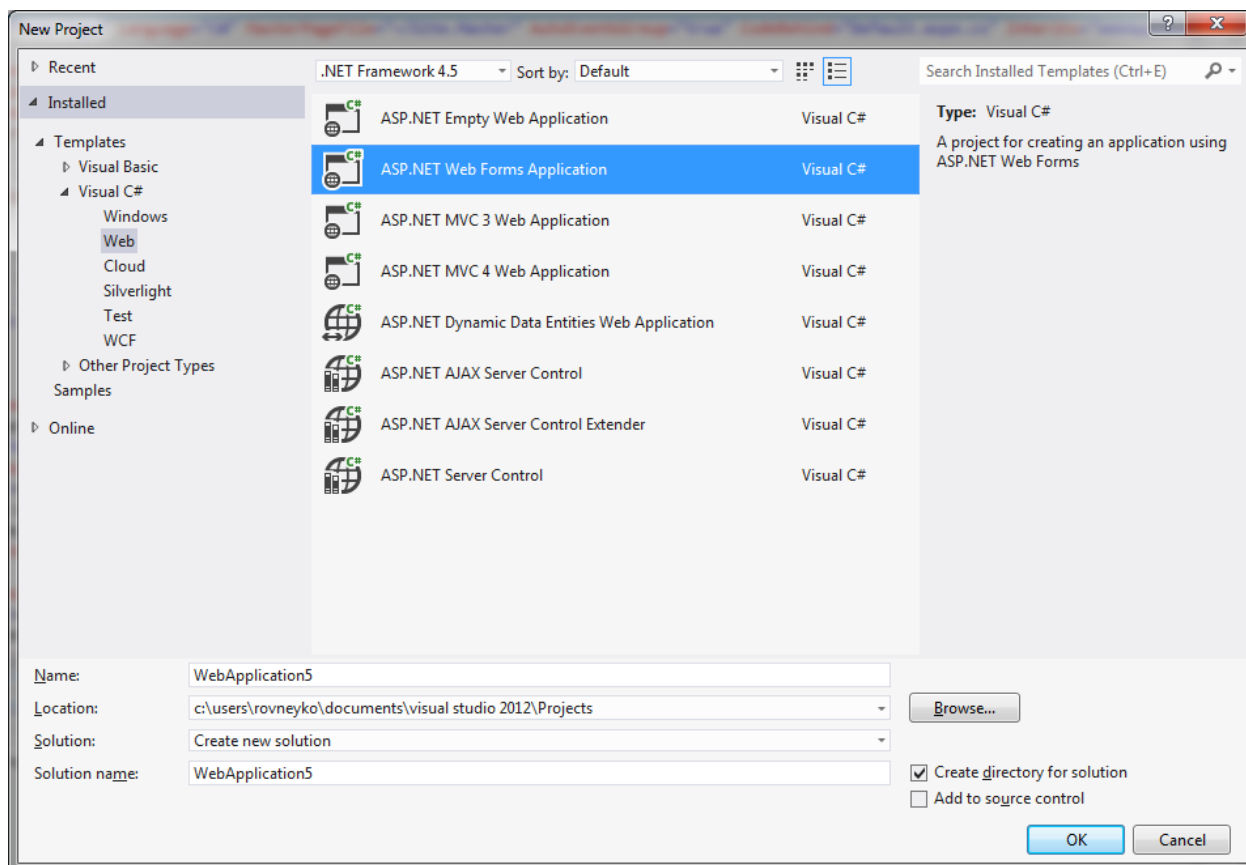


Рисунок 14 – Создание Web-сайта

В группе «Toolbox» представлены все доступные компоненты web-формы.

Файлы с расширением «*.aspx» отвечают за отображение информации на web-сайте, а в «*.aspx.cs» содержит программный код на C#.

Для подключения библиотеке по работе с БД MySQL необходимо щелкнуть правой кнопкой на имя проекта и в контекстном меню выбрать «Add Reference». В открывшемся окне в группе «Extensions» пометить «галочкой» библиотеку «MySQL.Data» (рис. 15). После выполнения данной процедуры Ваше web-приложение сможет подключиться к Бд MySQL.

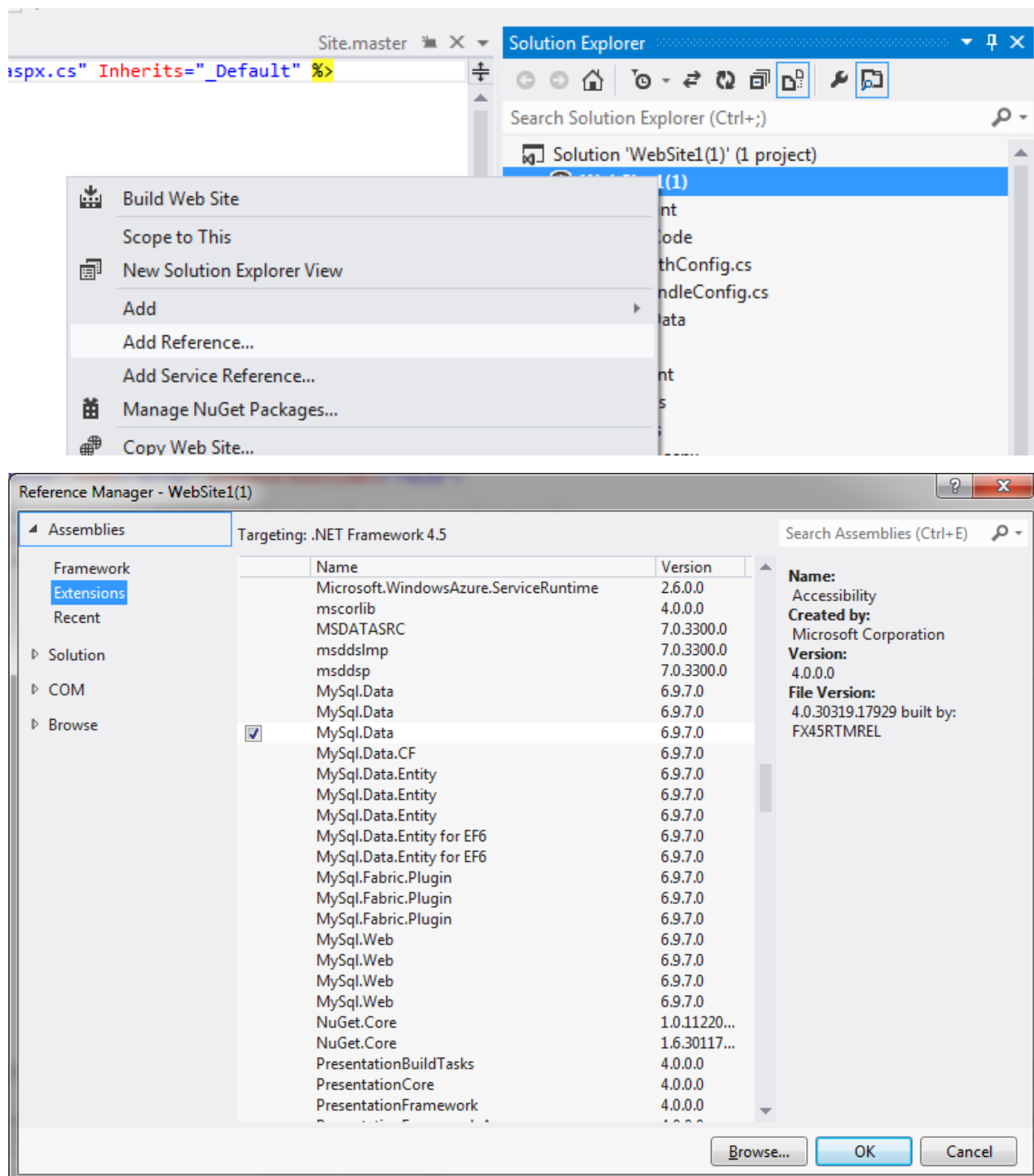


Рисунок 15 – Подключение библиотеки MySQL

Пример программы:

Содержимое файла Default.aspx:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebApplication4._Default" %>

<form id="form1" runat="server">
  <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False">
    <Columns>
      <asp:BoundField runat="server" DataField="Ид.номер" HeaderText="strFirst"/>
    </Columns>
  </asp:GridView>
</form>
```

```

        <asp:BoundField runat="server" DataField="Фамилия, Имя"
HeaderText="strLast"/>
        <asp:ButtonField Text="Button" />
    </Columns>
</asp:GridView>
</form>

```

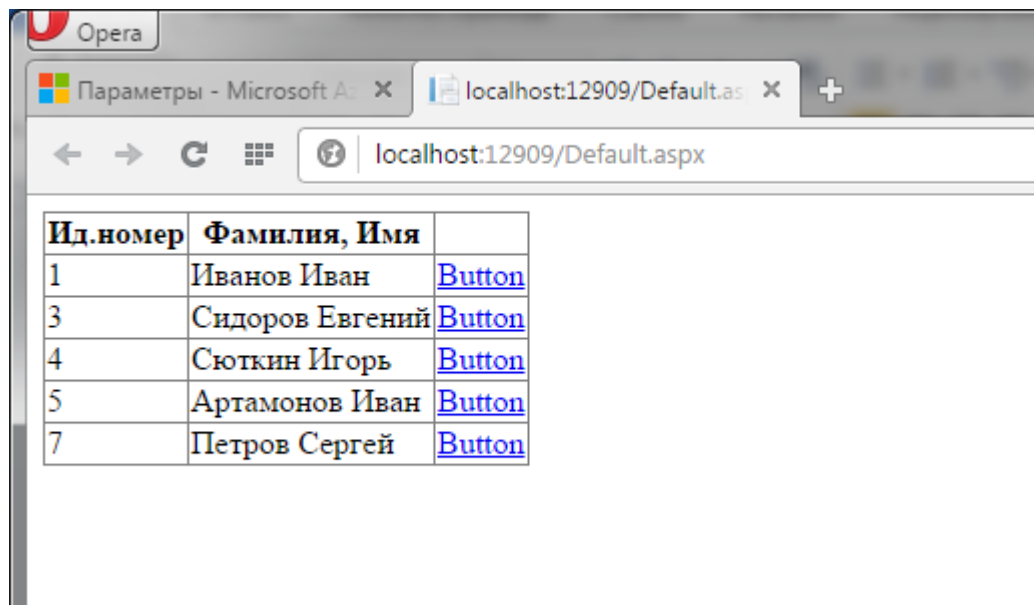
Содержимое файла Default.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;
namespace WebApplication4
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string strConnect = "server=br-cdbr-azure-south-
a.cloudapp.net;uid=bc2459e2c26dcd;pwd=42788b42;database=acsm_6b36e9a61d63d27";//строка
для подключения к БД
            MySqlConnection dbConn = new MySqlConnection(strConnect);
            dbConn.Open();
            DataSet ds = new DataSet();
            string sql = "SELECT * from table1";
            MySqlCommand cmd = new MySqlCommand(sql, dbConn);
            DataTable newTable = createDataTableTemplate(); //создаем таблицу, в которую
будем записывать результат запроса
            MySqlDataReader rdr = cmd.ExecuteReader();//выполняем запрос и записываем
результат в массив
            while (rdr.Read())
            {
                DataRow newRow = newTable.NewRow();
                newRow["id"] = rdr.GetString(0);
                newRow["name"] = rdr.GetString(1);
                newTable.Rows.Add(newRow);//добавление в таблицу строки с результатом
запроса
            }
            rdr.Close();
            dbConn.Close();//закрываем соединение
            GridView1.DataSource = newTable;//присваиваем источник данных элементу
отображения таблицы
            GridView1.DataBind();//применяем источник данных
        }
        private DataTable createDataTableTemplate()//конструктор таблицы
        {
            DataTable table = new DataTable("Table Title");
            DataColumn col1 = new DataColumn("id");
            col1.DataType = System.Type.GetType("System.String");
            DataColumn col2 = new DataColumn("name");
            col2.DataType = System.Type.GetType("System.String");
            table.Columns.Add(col1);
            table.Columns.Add(col2);
            return table;
        }
    }
}

```

Результат выполнения программы:



The screenshot shows a web browser window with the address bar displaying 'localhost:12909/Default.aspx'. The main content area contains a table with three columns: 'Ид.номер', 'Фамилия, Имя', and an empty column. The table has five rows of data, each with a 'Button' link in the third column.

| Ид.номер | Фамилия, Имя | |
|----------|-----------------|------------------------|
| 1 | Иванов Иван | Button |
| 3 | Сидоров Евгений | Button |
| 4 | Сюткин Игорь | Button |
| 5 | Артамонов Иван | Button |
| 7 | Петров Сергей | Button |

Размещение Web-приложения на портале Microsoft Azure

На портале Microsoft Azure в меню выбираем «Веб-приложения». В открывшемся окне нажимаем кнопку «Добавить». После задаем имя сайта и создаем его.

После создания сайта необходимо скачать «Профиль публикации» как показано на рисунке 16.

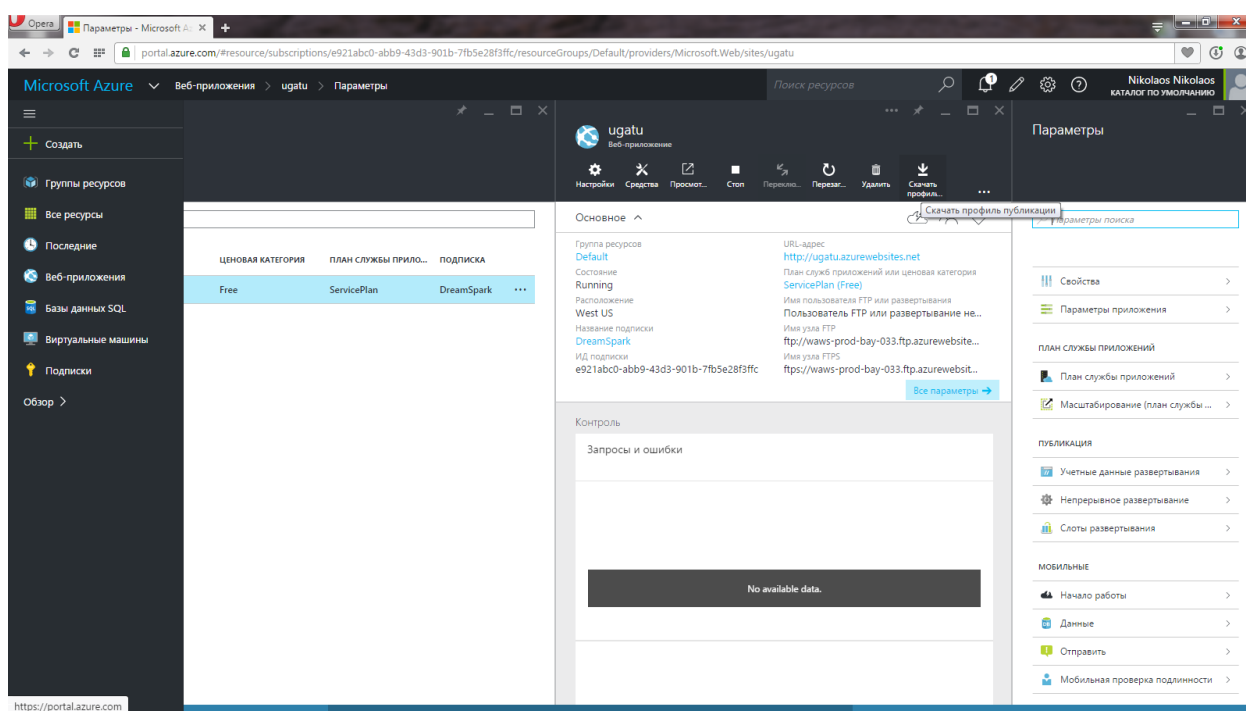


Рисунок 16 – Скачивание профиля публикации

Для работы библиотеке MySQL в Microsoft Azure необходимо добавить ее в проект. В «Solution Explorer» в группе «References» необходимо найти «Mysql.Data» и установить свойству «Copy Local» значение «True» (рис. 17).

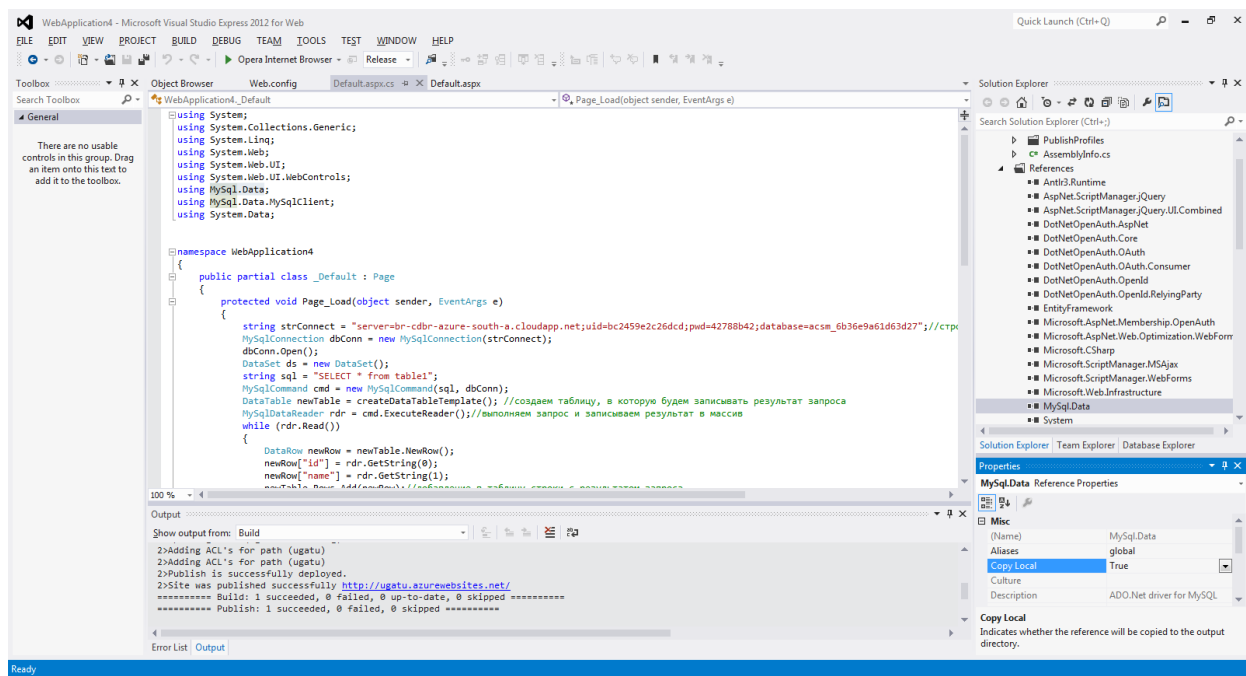


Рисунок 17 – Установка MySql.Data

После выполнения процедуры скачивания профиля в Microsoft Visual Studio 2012 Express for Web заходим в меню «BUILD → Publish Selection». В открывшемся окне импортируем скаченный файл с профилем. После выбора произойдёт заполнение полей группы «Connection» и в случае если все заполнилось корректно, то публикуем приложение нажатием на кнопку «Publish» (рис. 18).

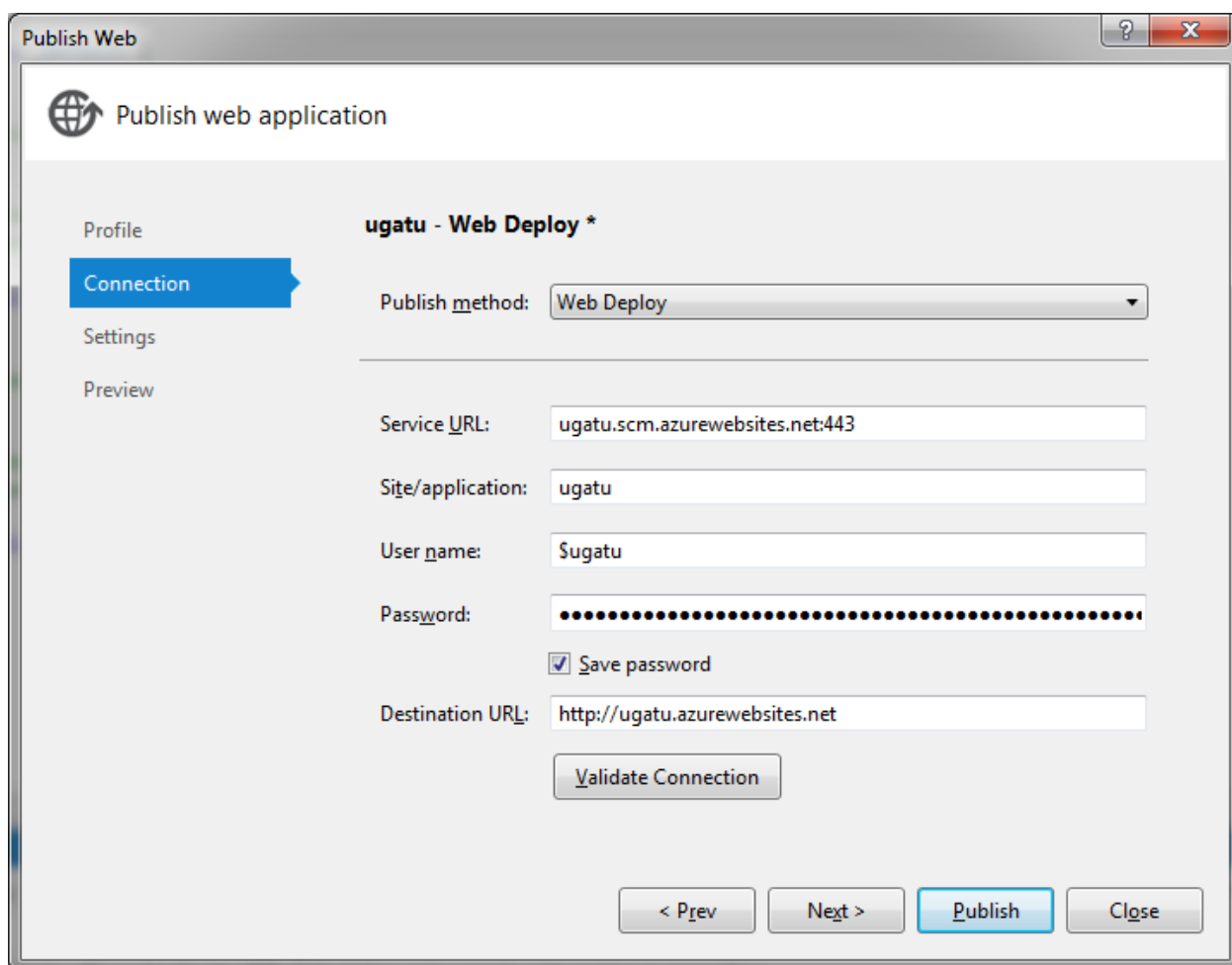


Рисунок 18 – Публикация приложения в Microsoft Azure

После публикации Ваше web-приложение доступно по Internet-ссылке, указанной в поле «Destination URL» на рис. 18 (рис. 19).

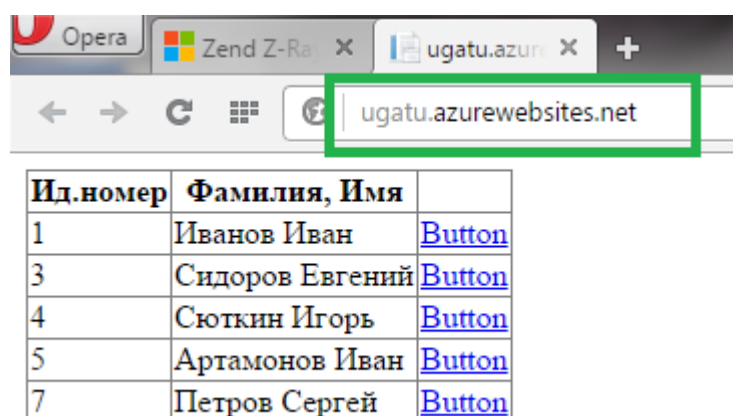


Рисунок 18 – Опубликованное приложение в Microsoft Azure

Ход выполнения работы

1. Внимательное изучение теоретического материала.
2. Разработка Web-приложение.
3. Разместить Web-приложение в Microsoft Azure.
4. Демонстрация преподавателю.
5. Подготовка краткого отчета с указанием основных этапов работы и результатов работы.
6. Подготовка ответов на контрольные вопросы.
7. Защита лабораторной работы (включает объяснение основных этапов разработки, ответы на контрольные вопросы).

Контрольные вопросы

1. Возможности Microsoft Visual Studio 2012 Express for Web
2. Пояснить этапы разработки Web-приложения.
3. Для чего необходим файл Web.config?