Московский государственный технический университет
имени Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

профессор Э.Н. Самохвалов
доцент Г.И. Ревунков
доцент Ю.Е. Гапанюк

Методические указания к лабораторным работам по курсу «Разработка интернет - приложений» (5 семестр)

Москва
2014

СОДЕРЖАНИЕ

1 Введение

Дисциплина «Разработка интернет - приложений» предназначена для обучения студентов основам разработки веб-ориентированных информационных систем.

Лабораторный практикум по курсу «Разработка интернет - приложений» предназначен для формирования у студентов компетенций, связанных с разработкой клиентской и серверной частей веб-приложений, обработки данных в веб-приложениях.

Лабораторный практикум содержит 6 лабораторных работ:

1. **Разработка макета сайта на языке разметки HTML (2 часа)**

Лабораторная работа предназначена для практического освоения методов разработки клиентской части веб-приложений с использованием языка разметки HTML.

2. **Разработка макета сайта с использованием библиотеки Twitter Bootstrap (2 часа)**

Лабораторная работа предназначена для практического освоения методов разработки клиентской части веб-приложений с использованием библиотеки Twitter Bootstrap.

3. **Разработка программы на языке C#, реализующей работу с LINQ to Objects (2 часа)**

Лабораторная работа предназначена для практического освоения методов обработки данных в веб-приложениях без использования СУБД.

4. **Разработка программы на языке C#, реализующей работу с LINQ to Entities (3 часа)**

Лабораторная работа предназначена для практического освоения методов обработки данных в веб-приложениях с использованием СУБД.

5. **Создание проекта ASP.NET MVC с использованием механизма «scaffolding» (4 часа)**

Лабораторная работа предназначена для практического освоения методов быстрого построения серверной части веб-приложения.

6. **Обработка динамических данных с использованием технологии ASP.NET Web API (4 часа)**

Лабораторная работа предназначена для практического освоения методов создания клиентской и серверной частей веб-приложения на основе обработки динамических данных.

2 Цель лабораторного практикума

Целью лабораторного практикума является содействие в формировании следующих компетенций:

Общепрофессиональные компетенции (ОП):

- способен собирать, анализировать научно-техническую информацию и учитывать её в профессиональной деятельности (ОП-2);
- способен работать с информацией в глобальных компьютерных сетях (ОП-3).

Компетенции в проектной деятельности (ПР):

- способен разрабатывать интерфейсы конечных пользователей (ПР-3);
- способен разрабатывать спецификации прикладных программ и реализовывать их на языках высокого уровня (ПР-5).

Компетенции в производственно-технологической деятельности (ПТ):

- способен работать с современными инструментальными средствами проектирования и разработки баз данных, прикладных программ, аппаратного обеспечения автоматизированных информационных систем (ПТ-1);
- умеет использовать современные технологии отладки прикладных программ (ПТ-2).

В результате выполнения лабораторного практикума студент должен уметь:

- разрабатывать модели данных на основе технологии ADO.NET Entity Framework и запросы с использованием языка запросов LINQ to Entities (ПР-5, ПТ-1, ПТ-2);

- разрабатывать веб-приложения на основе технологии ASP.NET MVC с использованием «scaffolding» (ПР-3, ПР-5, ПТ-1, ПТ-2);
- разрабатывать контроллеры и виды ASP.NET MVC для реализации CRUD-функциональности (ПР-3, ПР-5, ПТ-1, ПТ-2);
- разрабатывать контроллеры и виды ASP.NET MVC для реализации отчетов (ПР-3, ПР-5, ПТ-1, ПТ-2).

3 Краткая характеристика объекта изучения, исследования

Объектом изучения лабораторного практикума является разработка веб-приложений.

В частности, в рамках лабораторного практикума изучаются такие аспекты разработки веб-приложений, как:

- разработка клиентской части веб-приложений с использованием языка разметки HTML и библиотеки Twitter Bootstrap;
- разработка серверной части веб-приложений с использованием технологии ASP.NET MVC;
- разработка моделей данных с использованием технологии ADO.NET Entity Framework;
- обработка данных в веб-приложениях с использованием технологий LINQ to Objects и LINQ to Entities.

Разработка веб-приложений является развитой областью профессиональных знаний, описание которой в силу большого объема не может быть представлено в рамках настоящих методических указаний.

Фрагменты программ, которые могут быть использованы для самостоятельного анализа студентами и применены в лабораторных работах, представлены в разделе «Вспомогательные материалы для выполнения лабораторных работ».

Для изучения основ разработки веб-приложений и выполнения лабораторных работ можно рекомендовать источники [1-4].

4 Схема и описание лабораторной установки

В качестве лабораторной установки используется компьютер со следующим программным обеспечением:

- операционная система Windows 7 и выше;
- среда разработки Visual Studio 2010 и выше;
- СУБД SQL Server 2008 и выше.

Все программное обеспечение является лицензионным и предоставляется компанией Microsoft в рамках академической программы сотрудничества с МГТУ им. Н.Э. Баумана.

5 Содержание отчета по лабораторным работам

Отчеты разрабатываются отдельно по каждой лабораторной работе. Отчет по каждой лабораторной работе должен включать:

- титульный лист;
- описание задания лабораторной работы;
- тексты программ;
- диаграмму классов (в случае использования C#);
- результаты выполнения программы, экранные формы.

6 Задачи и порядок выполнения работ

6.1 Лабораторная работа 1

Разработать макет сайта на языке разметки HTML. Макет сайта должен включать следующие элементы:

- Списки.
- Изображения.
- Таблицы.
- Фреймы, для создания меню используются гиперссылки.
- Плавающие фреймы.
- Элементы HTML-форм.

6.2 Лабораторная работа 2

Разработать макет сайта с использованием библиотеки Twitter Bootstrap. Макет сайта должен включать следующие элементы:

- Таблицы.
- Элементы HTML-форм.
- Панель навигации (в верхней части страницы).
- Выпадающие списки кнопок (могут быть использованы в панели навигации).
- Индикаторы прогресса.

6.3 Лабораторная работа 3

Разработать программу, реализующую обработку данных с использованием библиотеки LINQ to Objects.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Создайте класс «Сотрудник», содержащий поля:
    - ID записи о сотруднике;
    - Фамилия сотрудника;
    - ID записи об отделе.
3. Создайте класс «Отдел», содержащий поля:
    - ID записи об отделе;
    - Наименование отдела.
4. Предполагая, что «Отдел» и «Сотрудник» связаны соотношением один-ко-многим, разработайте следующие запросы:
    - Выведите список всех сотрудников и отделов, отсортированный по отделам.
    - Выведите список всех сотрудников, у которых фамилия начинается с буквы «А».

- Выведите список всех отделов и количество сотрудников в каждом отделе.
- Выведите список отделов, в которых у всех сотрудников фамилия начинается с буквы «А».
- Выведите список отделов, в которых хотя бы у одного сотрудника фамилия начинается с буквы «А».

5. Создайте класс «Сотрудники отдела», содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.
6. Предполагая, что «Отдел» и «Сотрудник» связаны соотношением много-ко-многим с использованием класса «Сотрудники отдела» разработайте запросы, указанные в пункте 4.

6.4 Лабораторная работа 4

Разработать программу, реализующую обработку данных с использованием библиотеки LINQ to Entities.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Используя условия лабораторной работы №3, создайте модель Entity Framework (на основе SQL Server) и разработайте запросы, указанные в лабораторной работе №3.
3. Рекомендуется разработать две различные модели для случаев один-ко-многим и много-ко-многим.

6.5 Лабораторная работа 5

Создание проекта ASP.NET MVC с использованием механизма «scaffolding».

1. Создайте модель Entity Framework, содержащую две сущности, связанные соотношением один-ко-многим.

2. Заполните модель тестовыми данными с использованием программы на языке C# (возможно использованием отдельного проекта C# или контроллера ASP.NET MVC).
3. С использованием стандартного механизма «scaffolding» сгенерируйте по модели макет приложения ASP.NET MVC, позволяющий добавлять, редактировать и удалять данные.
4. Создайте контроллер и вид, формирующий выборку данных из модели данных Entity Framework в виде HTML-таблицы с использованием технологии LINQ to Entities.

6.6 Лабораторная работа 6

Обработка динамических данных с использованием технологии ASP.NET Web API.

1. С использованием контроллера ASP.NET Web API реализуйте работающий индикатор процесса Twitter Bootstrap, который отрабатывает за заданное число секунд (может быть задано в виде константы или вводиться пользователем).
2. Создайте контроллер ASP.NET Web API, генерирующий случайные числа (данные графика). С использованием JavaScript реализуйте динамическое обновление данных графика в окне браузера.
   - Для реализации AJAX-запросов к серверу может быть использована библиотека jQuery.
   - Для отображения графика может быть использована произвольная библиотека на JavaScript, например http://dygraphs.com/

7 Вспомогательные материалы для выполнения лабораторных работ

7.1 Лабораторная работа 1

При разработке макета сайта можно использовать любые справочные материалы по языку разметки HTML. В качестве справочника может быть использован сайт http://htmlbook.ru/

7.2 Лабораторная работа 2

Библиотека Twitter Bootstrap является библиотекой с открытым исходным кодом. Официальный сайт Twitter Bootstrap http://getbootstrap.com/

Следует учитывать, что библиотека довольно быстро изменяется. Многие примеры, которые приведены на русскоязычных сайтах, в том числе http://mybootstrap.ru/ к сожалению, не работают в последних версиях библиотеки. Все примеры рекомендуется брать именно с официального сайта.

Также может представлять интерес сайт с расширениями (дополнительными стилями, компонентами и т.д.) для Twitter Bootstrap http://bootsnipp.com/resources

7.3 Лабораторная работа 3

7.3.1 Фрагмент программы, реализующей обработку данных с использованием библиотеки LINQ to Objects

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SimpleLINQ
{
    class Program
    {

        /// <summary>
        /// Класс данных
        /// </summary>
        public class Data
        {
            /// <summary>
            /// Ключ
            /// </summary>
```

```
            public int id;

            /// <summary>
            /// Для группировки
            /// </summary>
            public string grp;

            /// <summary>
            /// Значение
            /// </summary>
            public string value;

            /// <summary>
            /// Конструктор
            /// </summary>
            public Data(int i, string g, string v)
            {
                this.id = i;
                this.grp = g;
                this.value = v;
            }

            /// <summary>
            /// Приведение к строке
            /// </summary>
            public override string ToString()
            {
                return "(id=" + this.id.ToString() + "; grp=" + this.grp + "; value=" +
this.value + ")";
            }
        }

        /// <summary>
        /// Класс для сравнения данных
        /// </summary>
        public class DataEqualityComparer : IEqualityComparer<Data>
        {

            public bool Equals(Data x, Data y)
            {
                bool Result = false;
                if (x.id == y.id && x.grp == y.grp && x.value == y.value) Result = true;
                return Result;
            }

            public int GetHashCode(Data obj)
            {
                return obj.id;
            }
        }

        /// <summary>
        /// Связь между списками
        /// </summary>
        public class DataLink
        {
            public int d1;
            public int d2;

            public DataLink(int i1, int i2)
            {
                this.d1 = i1;
                this.d2 = i2;
            }
        }
```

```
//Пример данных
static List<Data> d1 = new List<Data>()
    {
        new Data(1, "group1", "11"),
        new Data(2, "group1", "12"),
        new Data(3, "group2", "13"),
        new Data(5, "group2", "15")
    };

static List<Data> d2 = new List<Data>()
    {
        new Data(1, "group2", "21"),
        new Data(2, "group3", "221"),
        new Data(2, "group3", "222"),
        new Data(4, "group3", "24")
    };

static List<Data> d1_for_distinct = new List<Data>()
    {
        new Data(1, "group1", "11"),
        new Data(1, "group1", "11"),
        new Data(1, "group1", "11"),
        new Data(2, "group1", "12"),
        new Data(2, "group1", "12")
    };

static List<DataLink> lnk = new List<DataLink>()
{
    new DataLink(1,1),
    new DataLink(1,2),
    new DataLink(1,4),
    new DataLink(2,1),
    new DataLink(2,2),
    new DataLink(2,4),
    new DataLink(5,1),
    new DataLink(5,2)
};

static void Main(string[] args)
{

    Console.WriteLine("Простая выборка элементов");
    var q1 = from x in d1 select x;
    foreach (var x in q1) Console.WriteLine(x);

    //++++++++++++++++++++++++++++++++++++++++++++++++++++

    Console.WriteLine("Выборка отдельного поля (проекция)");
    var q2 = from x in d1 select x.value;
    foreach (var x in q2) Console.WriteLine(x);

    //++++++++++++++++++++++++++++++++++++++++++++++++++++

    Console.WriteLine("Создание нового объекта анонимного типа");
    var q3 = from x in d1
             select new { IDENTIFIER = x.id, VALUE = x.value };
    foreach (var x in q3) Console.WriteLine(x);


    //++++++++++++++++++++++++++++++++++++++++++++++++++++
    //++++++++++++++++++++++++++++++++++++++++++++++++++++
    //++++++++++++++++++++++++++++++++++++++++++++++++++++

    Console.WriteLine("Условия");
```

```
var q4 = from x in d1
         where x.id > 1 && (x.grp=="group1" || x.grp=="group2")
         select x;
foreach (var x in q4) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Выборка по значению типа");
object[] array = new object[] {123, "строка 1", true, "строка 2"};
var qo = from x in array.OfType<string>()
         select x;

foreach (var x in qo) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Сортировка");
var q5 = from x in d1
         where x.id > 1 && (x.grp == "group1" || x.grp == "group2")
         orderby x.grp descending, x.id descending
         select x;
foreach (var x in q5) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Сортировка (с использованием методов)");
var q51 = d1.Where(
    (x) =>
        { return x.id > 1 && (x.grp == "group1" || x.grp == "group2"); }
    )
    .OrderByDescending(x => x.grp).ThenByDescending(x => x.id);

foreach (var x in q51) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Partitioning Operators");
Console.WriteLine("Постраничная выдача данных");
var qp = GetPage(d1, 2, 2);
foreach (var x in qp) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Использование SkipWhile и TakeWhile");

int[] intArray = new int[] { 1,2,3,4,5,6,7,8 };
var qw = intArray.SkipWhile(x => (x < 4)).TakeWhile(x=>x<=7);

foreach (var x in qw) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++
//++++++++++++++++++++++++++++++++++++++++++++++++++

Console.WriteLine("Декартово произведение");
var q6 = from x in d1
         from y in d2
         select new { v1 = x.value, v2 = y.value };
foreach (var x in q6) Console.WriteLine(x);

//++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
            Console.WriteLine("Inner Join с использованием Where");
            var q7 = from x in d1
                     from y in d2
                     where x.id == y.id
                     select new { v1 = x.value, v2 = y.value };
            foreach (var x in q7) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Cross Join (Inner Join) с использованием Join");
            var q8 = from x in d1
                     join y in d2 on x.id equals y.id
                     select new { v1 = x.value, v2 = y.value };
            foreach (var x in q8) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Cross Join (сохранение объекта)");
            var q9 = from x in d1
                     join y in d2 on x.id equals y.id
                     select new { v1 = x.value, d2Group = y };
            foreach (var x in q9) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++

            //Выбираются все элементы из d1 и если есть связанные из d2 (outer join)
            //В temp помещается вся группа, ее элементы можно перебирать отдельно
            Console.WriteLine("Group Join");
            var q10 = from x in d1
                     join y in d2 on x.id equals y.id into temp
                     select new { v1 = x.value, d2Group = temp };
            foreach (var x in q10)
            {
                Console.WriteLine(x.v1);
                foreach(var y in x.d2Group)
                    Console.WriteLine("   " + y);
            }

            //++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Cross Join и Group Join");
            var q11 = from x in d1
                     join y in d2 on x.id equals y.id into temp
                     from t in temp
                     select new { v1 = x.value, v2 = t.value, cnt = temp.Count() };
            foreach (var x in q11) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Outer Join");
            var q12 = from x in d1
                      join y in d2 on x.id equals y.id into temp
                      from t in temp.DefaultIfEmpty()
                      select new { v1 = x.value, v2 = ( (t==null) ? "null" : t.value ) };
            foreach (var x in q12) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Использование Join для составных ключей");
            var q12_1 = from x in d1
                     join y in d1_for_distinct on new { x.id, x.grp } equals new { y.id, y.grp
} into details
                     from d in details select d;
```

```
            foreach (var x in q12_1) Console.WriteLine(x);


            //++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++

            //Действия над множествами

            Console.WriteLine("Distinct - неповторяющиеся значения");
            var q13 = (from x in d1 select x.grp).Distinct();
            foreach (var x in q13) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Distinct - повторяющиеся значения для объектов");
            var q14 = (from x in d1_for_distinct select x).Distinct();
            foreach (var x in q14) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Distinct - неповторяющиеся значения для объектов");
            var q15 = (from x in d1_for_distinct select x).Distinct(new
DataEqualityComparer());
            foreach (var x in q15) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Union - объединение с исключением дубликатов");
            int[] i1 = new int[] { 1, 2, 3, 4 };
            int[] i1_1 = new int[] { 2, 3, 4, 1 };
            int[] i2 = new int[] { 2, 3, 4, 5 };
            foreach (var x in i1.Union(i2)) Console.WriteLine(x);

            Console.WriteLine("Union - объединение для объектов");
            foreach (var x in d1.Union(d1_for_distinct)) Console.WriteLine(x);

            Console.WriteLine("Union - объединение для объектов с исключением дубликатов 1");
            foreach (var x in d1.Union(d1_for_distinct, new DataEqualityComparer()))
Console.WriteLine(x);

            Console.WriteLine("Union - объединение для объектов с исключением дубликатов 2");
            foreach (var x in d1.Union(d1_for_distinct).Union(d2).Distinct(new
DataEqualityComparer())) Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Concat - объединение без исключения дубликатов");
            foreach (var x in i1.Concat(i2)) Console.WriteLine(x);

            Console.WriteLine("SequenceEqual - проверка совпадения элементов и порядка их
следования");
            Console.WriteLine(i1.SequenceEqual(i1));
            Console.WriteLine(i1.SequenceEqual(i2));

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Intersect - пересечение множеств");
            foreach (var x in i1.Intersect(i2)) Console.WriteLine(x);

            Console.WriteLine("Intersect - пересечение множеств для объектов");
            foreach (var x in d1.Intersect(d1_for_distinct, new DataEqualityComparer()))
Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
            Console.WriteLine("Except - вычитание множеств");
            foreach (var x in i1.Except(i2)) Console.WriteLine(x);

            Console.WriteLine("Except - вычитание множеств для объектов");
            foreach (var x in d1.Except(d1_for_distinct, new DataEqualityComparer()))
Console.WriteLine(x);

            //++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Функции агрегирования");

            Console.WriteLine("Count - количество элементов");
            Console.WriteLine(d1.Count());

            Console.WriteLine("Count с условием");
            Console.WriteLine(d1.Count(x => x.id > 1));

            //Могут использоваться также следующие агрегирующие функции
            //Sum - сумма элементов
            //Min - минимальный элемент
            //Max - максимальный элемент
            //Average - среднее значение

            //++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Aggregate - агрегирование значений");
            var qa1 = d1.Aggregate(new Data(0,"",""),
                (total, next) =>
                {
                    if (next.id > 1) total.id += next.id;
                    return total;
                }
                );
            Console.WriteLine(qa1);

            //++++++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++++++
            //++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Группировка");
            var q16 = from x in d1.Union(d2)
                      group x by x.grp into g
                      select new { Key = g.Key, Values = g };

            foreach (var x in q16)
            {
                Console.WriteLine(x.Key);
                foreach (var y in x.Values)
                    Console.WriteLine("   " + y);
            }

            //++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Группировка с функциями агрегирования");
            var q17 = from x in d1.Union(d2)
                      group x by x.grp into g
                      select new { Key = g.Key, Values = g, cnt = g.Count(), cnt1 =
g.Count(x=>x.id>1), sum = g.Sum(x=>x.id), min = g.Min(x=>x.id) };

            foreach (var x in q17)
            {
                Console.WriteLine(x);
```

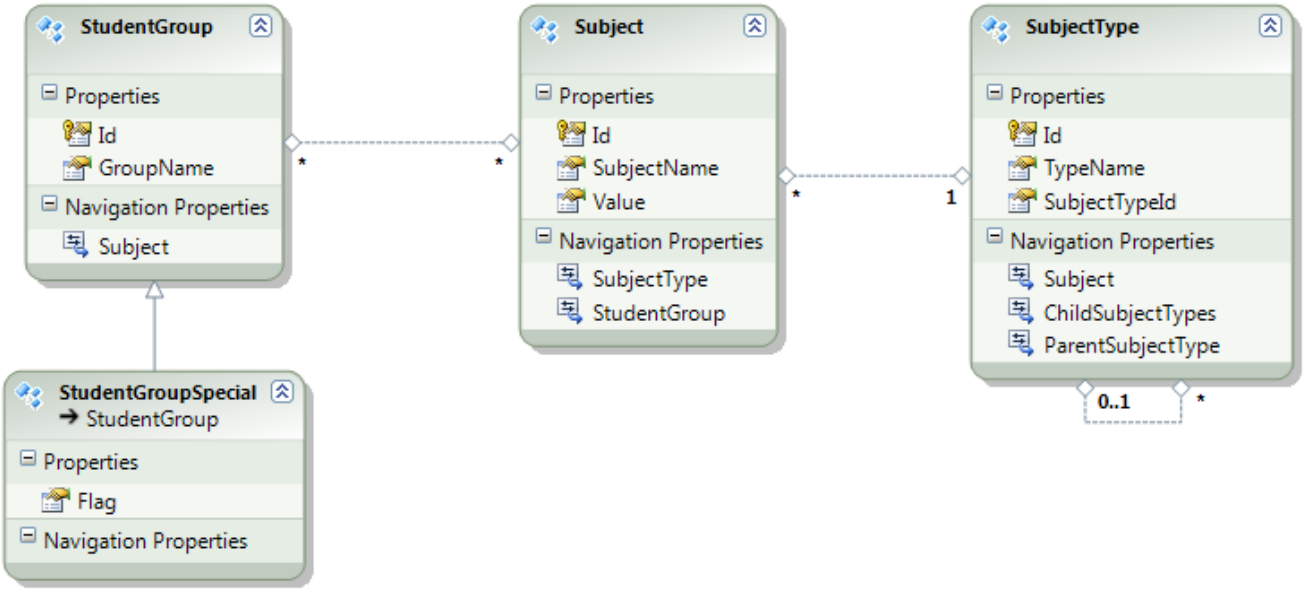
```
            foreach (var y in x.Values)
                Console.WriteLine("   " + y);
        }

        //++++++++++++++++++++++++++++++++++++++++++++++++++

        Console.WriteLine("Группировка - Any");
        var q18 = from x in d1.Union(d2)
                  group x by x.grp into g
                  where g.Any(x=> x.id > 3)
                  select new { Key = g.Key, Values = g };

        foreach (var x in q18)
        {
            Console.WriteLine(x.Key);
            foreach (var y in x.Values)
                Console.WriteLine("   " + y);
        }

        Console.WriteLine("Группировка - All");
        var q19 = from x in d1.Union(d2)
                  group x by x.grp into g
                  where g.All(x => x.id > 1)
                  select new { Key = g.Key, Values = g };

        foreach (var x in q19)
        {
            Console.WriteLine(x.Key);
            foreach (var y in x.Values)
                Console.WriteLine("   " + y);
        }

        //++++++++++++++++++++++++++++++++++++++++++++++++++

        Console.WriteLine("Имитация связи много-ко-многим");
        var lnk1 = from x in d1
                   join l in lnk on x.id equals l.d1 into temp
                   from t1 in temp
                   join y in d2 on t1.d2 equals y.id into temp2
                   from t2 in temp2
                   select new { id1 = x.id, id2 = t2.id };
        foreach (var x in lnk1) Console.WriteLine(x);

        //++++++++++++++++++++++++++++++++++++++++++++++++++

        Console.WriteLine("Имитация связи много-ко-многим, проверка условия");
        var lnk2 = from x in d1
                   join l in lnk on x.id equals l.d1 into temp
                   from t1 in temp
                   join y in d2 on t1.d2 equals y.id into temp2
                   where temp2.Any(t=>t.value == "24")
                   select x;
        foreach (var x in lnk2) Console.WriteLine(x);

        //++++++++++++++++++++++++++++++++++++++++++++++++++

        Console.WriteLine("Имитация связи много-ко-многим, использование let, проверка
условия");
        var lnk3 = from x in d1

                   let temp1 = from l in lnk where l.d1 == x.id select l

                   from t1 in temp1

                   let temp2 = from y in d2 where y.id == t1.d2 && y.value == "24"
```

```
                                select y
                        where temp2.Count() > 0

                        //let temp2 = from y in d2 where y.id == t1.d2
                        //            select y
                        //where temp2.Any(t=>t.value == "24")

                        select x;

            foreach (var x in lnk3) Console.WriteLine(x);

            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Deferred Execution - отложенное выполнение запроса");
            var e1 = from x in d1 select x;
            Console.WriteLine(e1.GetType().Name);
            foreach (var x in e1) Console.WriteLine(x);

            Console.WriteLine("При изменении источника данных запрос выдает новые
результаты");
            d1.Add(new Data(333, "",""));
            foreach (var x in e1) Console.WriteLine(x);

            //+++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Immediate Execution - немедленное выполнение запроса, результат
преобразуется в список ");
            var e2 = (from x in d1 select x).ToList();
            Console.WriteLine(e2.GetType().Name);
            foreach (var x in e2) Console.WriteLine(x);

            Console.WriteLine("Результат преобразуется в массив");
            var e3 = (from x in d1 select x).ToArray();
            Console.WriteLine(e3.GetType().Name);
            foreach (var x in e3) Console.WriteLine(x);

            Console.WriteLine("Результат преобразуется в Dictionary");
            var e4 = (from x in d1 select x).ToDictionary(x=>x.id);
            Console.WriteLine(e4.GetType().Name);
            foreach (var x in e4) Console.WriteLine(x);

            Console.WriteLine("Результат преобразуется в Lookup");
            var e5 = (from x in d1_for_distinct select x).ToLookup(x=>x.id);
            Console.WriteLine(e5.GetType().Name);
            foreach (var x in e5)
            {
                Console.WriteLine(x.Key);
                foreach (var y in x)
                    Console.WriteLine("    " + y);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Получение первого элемента из выборки");
            var f1 = (from x in d2 select x).First(x=>x.id==2);
            Console.WriteLine(f1);

            Console.WriteLine("Получение первого элемента или значения по умолчанию");
            var f2 = (from x in d2 select x).FirstOrDefault(x => x.id == 22);
            Console.WriteLine(f2==null ? "null" : f2.ToString());
```

```
            Console.WriteLine("Получение элемента в заданной позиции");
            var f3 = (from x in d2 select x).ElementAt(2);
            Console.WriteLine(f3);

            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++
            //+++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Генерация последовательностей");
            Console.WriteLine("Range");
            foreach (var x in Enumerable.Range(1, 5)) Console.WriteLine(x);

            Console.WriteLine("Repeat");
            foreach (var x in Enumerable.Repeat<int>(10,3)) Console.WriteLine(x);


            Console.ReadLine();
        }

        /// <summary>
        /// Получение нужной страницы данных
        /// </summary>
        static List<Data> GetPage(List<Data> data, int pageNum, int pageSize)
        {
            //Количество пропускаемых элементов
            int skipSize = (pageNum-1)*pageSize;

            var q = data.OrderBy(x => x.id).Skip(skipSize).Take(pageSize);

            return q.ToList();
        }

    }
}
```

7.4 *Лабораторная работа 4*

7.4.1 Диаграмма классов примера модели LINQ to Entities



7.4.2 Фрагмент программы, реализующей обработку данных с использованием библиотеки LINQ to Entities

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Objects;

namespace EntityLINQ
{
    class Program
    {
        /// <summary>
        /// Очистка данных
        /// </summary>
        static void ClearData()
        {
            LearningModelContainer db = new LearningModelContainer();

            //Удаление данных для связи много-ко-многим
            //для каждой записи StudentGroup удаляются все связи с Subject
            foreach (var gr in db.StudentGroupSet.ToList())
            {
                foreach (var gr_subj in gr.Subject.ToList())
                {
                    gr.Subject.Remove(gr_subj);
                }
            }
            db.SaveChanges();

            db.StudentGroupSet.ToList().ForEach(db.StudentGroupSet.DeleteObject);
            db.SaveChanges();

            db.SubjectSet.ToList().ForEach(db.SubjectSet.DeleteObject);
            db.SaveChanges();

            db.SubjectTypeSet.ToList().ForEach(db.SubjectTypeSet.DeleteObject);
            db.SaveChanges();
        }

        /// <summary>
        /// Заполнение данных
        /// </summary>
        static void InitData()
        {
            LearningModelContainer db = new LearningModelContainer();

            //Добавление типов предметов

            SubjectType st_tech = new SubjectType
            {
                TypeName = "технический цикл",
                ParentSubjectType = null
            };
            db.SubjectTypeSet.AddObject(st_tech);

            SubjectType st_hum = new SubjectType
            {
                TypeName = "гуманитарный цикл",
                ParentSubjectType = null
            };
```

```
SubjectType st1 = new SubjectType
{
    TypeName = "базовые",
    ParentSubjectType = st_tech
};

SubjectType st2 = new SubjectType
{
    TypeName = "специальные",
    ParentSubjectType = st_tech
};

SubjectType st3 = new SubjectType
{
    TypeName = "исторические",
    ParentSubjectType = st_hum
};

SubjectType st3_1 = new SubjectType
{
    TypeName = "новая история",
    ParentSubjectType = st3
};

SubjectType st3_2 = new SubjectType
{
    TypeName = "новейшая история",
    ParentSubjectType = st3
};

db.SubjectTypeSet.AddObject(st_tech);
db.SubjectTypeSet.AddObject(st_hum);
db.SubjectTypeSet.AddObject(st1);
db.SubjectTypeSet.AddObject(st2);
db.SubjectTypeSet.AddObject(st3);
db.SubjectTypeSet.AddObject(st3_1);
db.SubjectTypeSet.AddObject(st3_2);

//Добавление предметов

Subject sb1 = new Subject
{
     SubjectName = "математика",
     Value = 100, //часов
     SubjectType = st1
};

Subject sb2 = new Subject
{
    SubjectName = "физика",
    Value = 80, //часов
    SubjectType = st1
};

Subject sb3 = new Subject
{
    SubjectName = "информатика",
    Value = 120, //часов
    SubjectType = st2
};

Subject sb4 = new Subject
{
    SubjectName = "базы данных",
    Value = 150, //часов
```

```
            SubjectType = st2
        };

        Subject sb5 = new Subject
        {
            SubjectName = "сетевые технологии",
            Value = 170, //часов
            SubjectType = st2
        };

        db.SubjectSet.AddObject(sb1);
        db.SubjectSet.AddObject(sb2);
        db.SubjectSet.AddObject(sb3);
        db.SubjectSet.AddObject(sb4);
        db.SubjectSet.AddObject(sb5);

        //Добавление групп

        StudentGroup g1 = new StudentGroup
        {
            GroupName = "ИУ5-11"
        };

        StudentGroup g2 = new StudentGroup
        {
            GroupName = "ИУ5-51"
        };

        StudentGroupSpecial g3 = new StudentGroupSpecial
        {
            GroupName = "ИУ5с-11",
            Flag = true
        };

        db.StudentGroupSet.AddObject(g1);
        db.StudentGroupSet.AddObject(g2);
        db.StudentGroupSet.AddObject(g3);

        //Установка связи много-ко многим

        g1.Subject.Add(sb1);
        g1.Subject.Add(sb2);

        g2.Subject.Add(sb3);
        g2.Subject.Add(sb4);
        g2.Subject.Add(sb5);

        g3.Subject.Add(sb1);
        g3.Subject.Add(sb2);
        g3.Subject.Add(sb4);

        //Сохранение данных в БД
        db.SaveChanges();

    }

    /// <summary>
    /// Примеры запросов
    /// </summary>
    static void Queries()
    {
        //Выдача иерархии типов пердметов
        WriteSubjectTypeTree(-1, 0);
        Console.WriteLine();
```

```
            LearningModelContainer db = new LearningModelContainer();

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("Получение всех курсов и групп, которым они читаются");
            var q1 = from s in db.SubjectSet select s;

            foreach (var s in q1)
            {
                Console.WriteLine(s.SubjectName + " (" + s.SubjectType.TypeName + ")");
                foreach (var g in s.StudentGroup)
                {
                    Console.WriteLine("    " + g.GroupName);
                }
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nКоличество часов по всем предметам для каждой группы");
            var q2 = from g in db.StudentGroupSet
                     select new { GroupName = g.GroupName, ValueSum = g.Subject.Sum(x =>
x.Value), Subject = g.Subject };

            foreach (var g in q2)
            {
                Console.WriteLine(g.GroupName + " (" + g.ValueSum.ToString() + " часов)");
                foreach (var s in g.Subject)
                {
                    Console.WriteLine("    " + s.SubjectName + " (" + s.Value.ToString() + "
часов)");
                }
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nПредметы, читаемые группам");
            var q3 = from g in db.StudentGroupSet
                     from s in g.Subject
                     select new { SubjectName = s.SubjectName, Value = s.Value, GroupName =
g.GroupName };

            var q31 = from t in q3
                      orderby t.SubjectName, t.GroupName
                      select t;

            foreach (var g in q31)
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nКоличество часов по предмету для всех групп");
            var q32 = from t in q3
                      group t by t.SubjectName into temp
                      select new { GroupName = temp.Key, SumValue = temp.Sum(x=>x.Value) };

            foreach (var g in q32)
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
            Console.WriteLine("\nГруппы для которых читаются специальные курсы (с
использованием contains)");

            string[] arr = new string[] { "специальные", "другой" };

            var qc = from g in db.StudentGroupSet
                     from s in g.Subject
                     where arr.Contains(s.SubjectType.TypeName)
                     select new { GroupName = g.GroupName, SubjectTypeName =
s.SubjectType.TypeName };

            foreach (var g in qc)
            {
                Console.WriteLine(g);
            }


            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nТипы курсов, читаемых для группы (с повторяющимися
записями)");
            var q4 = from g in db.StudentGroupSet
                     from s in g.Subject
                     select new { GroupName = g.GroupName, SubjectTypeName =
s.SubjectType.TypeName };

            foreach (var g in q4)
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nТипы курсов, читаемых для группы");
            var q41 = from t in q4.Distinct()
                      select t;

            foreach (var g in q41)
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nГруппы, которым читаются базовые курсы");
            var q42 = from t in q4.Distinct()
                      where t.SubjectTypeName == "базовые"
                      select t;

            foreach (var g in q42)
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nГруппы, которым читаются базовые курсы (использование any)");
            var q43 = from t in q4.Distinct()
                      group t by t.GroupName into temp
                      where temp.Any(
                        (data) => data.SubjectTypeName == "базовые"
                      )
                      select temp.Key;

            foreach (var g in q43)
```

```
            {
                Console.WriteLine(g);
            }

            //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

            Console.WriteLine("\nГруппы, которым читаются только базовые курсы (использование
all)");
            var q44 = from t in q4.Distinct()
                      group t by t.GroupName into temp
                      where temp.All(
                        (data) => data.SubjectTypeName == "базовые"
                      )
                      select temp.Key;

            foreach (var g in q44)
            {
                Console.WriteLine(g);
            }


            Console.WriteLine("\nПолучение сгенерированного SQL");
            var trace = ((ObjectQuery)q44).ToTraceString();
            Console.WriteLine(trace);

        }

        /// <summary>
        /// Кэширование данных с использованием Include
        /// </summary>
        public static void IncludeExample()
        {

            LearningModelContainer db = new LearningModelContainer();

            //Отключение загрузки связанных данных
            db.ContextOptions.LazyLoadingEnabled = false;

            Console.WriteLine("\nБез использования Include");
            var q51 = (from x in db.SubjectSet
                       select x).ToList();

            WriteSubjectList(q51);

            Console.WriteLine("\nС использованием Include");
            var q52 = (from x in db.SubjectSet.Include("SubjectType").Include("StudentGroup")
                       select x).ToList();

            WriteSubjectList(q52);

        }

        /// <summary>
        /// Вывод списка
        /// </summary>
        /// <param name="list"></param>
        public static void WriteSubjectList(List<Subject> list)
        {
            foreach (var x in list)
            {
                string TypeName = "";
                if (x.SubjectType != null) TypeName = x.SubjectType.TypeName;
                else TypeName = "null";

                Console.WriteLine(x.SubjectName + " (" + TypeName + ")");
```

```
            if (x.StudentGroup != null)
            {
                foreach (var y in x.StudentGroup)
                {
                    Console.WriteLine("    " + y.GroupName);
                }
            }
        }

    }

    public static void WriteSubjectTypeTree(int ParentSubjectTypeParam, int LevelParam)
    {
        LearningModelContainer db = new LearningModelContainer();
        var q = from x in db.SubjectTypeSet select x;

        if (ParentSubjectTypeParam == -1)
        {
            //Поиск корневых элементов (ParentSubjectType == null)
            q = q.Where(x => x.SubjectTypeId.HasValue == false);
        }
        else
        {
            //Поиск элементов с заданным элементом верхнего уровня
            q = q.Where(x => x.SubjectTypeId.Value == ParentSubjectTypeParam);
        }

        //Если существуют элементы на данном уровне иерархии
        if (q.Count() > 0)
        {
            //Сортировка
            q = q.OrderBy(x => x.TypeName);

            //Перебор всех значений на заданном уровне иерархии
            foreach (var x in q)
            {
                //Вывод отступа
                if (LevelParam > 0)
                {
                    for (int i = 0; i < LevelParam; i++) Console.Write("  ");
                }
                //Вывод значения
                Console.WriteLine(x.TypeName);

                //Рекурсивный вызов функции для всех элементов, вложенных в текущий
                WriteSubjectTypeTree(x.Id, LevelParam + 1);
            }
        }
    }

    static void Main(string[] args)
    {
        /*
        //Заполнение данных в случае пустой БД
        LearningModelContainer db = new LearningModelContainer();
        if (db.SubjectTypeSet.Count() == 0)
        {
            //Инициализация данных для запросов
            InitData();
        }
        */

        ClearData();
        InitData();
```

```
            Queries();
            IncludeExample();

            Console.ReadLine();
        }
    }
}
```

7.5 Лабораторная работа 5

7.5.1 Пример контроллера, сгенерированного с использованием стандартного механизма «scaffolding»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using ReportExample.Models;

namespace ReportExample.Controllers
{
    public class ProcessorController : Controller
    {
        private Model1Container db = new Model1Container();

        //
        // GET: /Processor/

        public ActionResult Index()
        {
            return View(db.Processors.ToList());
        }

        //
        // GET: /Processor/Details/5

        public ActionResult Details(int id = 0)
        {
            Processor processor = db.Processors.Single(p => p.Id == id);
            if (processor == null)
            {
                return HttpNotFound();
            }
            return View(processor);
        }

        //
        // GET: /Processor/Create

        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Processor/Create

        [HttpPost]
        public ActionResult Create(Processor processor)
```

```
{
    if (ModelState.IsValid)
    {
        db.Processors.AddObject(processor);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(processor);
}

//
// GET: /Processor/Edit/5

public ActionResult Edit(int id = 0)
{
    Processor processor = db.Processors.Single(p => p.Id == id);
    if (processor == null)
    {
        return HttpNotFound();
    }
    return View(processor);
}

//
// POST: /Processor/Edit/5

[HttpPost]
public ActionResult Edit(Processor processor)
{
    if (ModelState.IsValid)
    {
        db.Processors.Attach(processor);
        db.ObjectStateManager.ChangeObjectState(processor, EntityState.Modified);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(processor);
}

//
// GET: /Processor/Delete/5

public ActionResult Delete(int id = 0)
{
    Processor processor = db.Processors.Single(p => p.Id == id);
    if (processor == null)
    {
        return HttpNotFound();
    }
    return View(processor);
}

//
// POST: /Processor/Delete/5

[HttpPost, ActionName("Delete")]
public ActionResult DeleteConfirmed(int id)
{
    Processor processor = db.Processors.Single(p => p.Id == id);
    db.Processors.DeleteObject(processor);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

```
        protected override void Dispose(bool disposing)
        {
            db.Dispose();
            base.Dispose(disposing);
        }
    }
}
```

7.5.2 Пример контроллера, используемого для формирования отчета

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace ReportExample.Controllers
{
    public class SimpleReportController : Controller
    {
        //
        // GET: /SimpleReport/

        public ActionResult Report()
        {
            return View();
        }

    }
}
```

7.5.3 Пример вида (файл «Report.cshtml»), формирующего выборку данных из модели данных Entity Framework в виде HTML-таблицы

```
@using ReportExample.Models
@{
    ViewBag.Title = "Report";

    Model1Container db = new Model1Container();
    var q1 = (from x in db.Computers
              orderby x.HDD
              select x
              ).ToList();

    int sumHDD = q1.Sum(x => x.HDD);

}

<h2>Жесткие диски компьютеров</h2>

<table>
<tr>
<th>Компьютер</th>
<th>Емкость жесткого диска</th>
</tr>

@foreach (var x in q1)
{
<tr>
    <td>@x.ComputerName</td>
    <td align="right">@x.HDD</td>
```

```
</tr>
}

<tr>
<td>ИТОГО</td>
<td align="right">@sumHDD</td>
</tr>

</table>
```

7.6 Лабораторная работа 6

7.6.1 Пример контроллера ASP.NET Web API

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using MvcAjax.Helpers;
using MvcAjax.Models;

namespace MvcAjax.Controllers
{
    /// <summary>
    /// Формат результата
    /// </summary>
    public class serverrandomdata
    {
        public string x { get; set; }
        public string y { get; set; }
        public numbersumdata sumdata { get; set; }
    }

    public class ServerRandomController : ApiController
    {
        public static List<serverrandomdata> data = new List<serverrandomdata>();
        static Random random = new Random();

        // GET api/serverrandom
        public List<serverrandomdata> Get()
        {
            int i = random.Next(100);

            //Генерация нового значения
            serverrandomdata current = new serverrandomdata()
            {
                x = DateTime.Now.ToString(ConstHelper.DateTimeFormat),
                y = i.ToString(),
                sumdata = NumberSum.AddNumber(i)
            };

            //Добавление в список
            data.Add(current);

            if (data.Count > 10)
            {
                data.RemoveRange(0, data.Count - 10);
            }

            return data;
        }
```

```
    }
}
```

7.6.2 Пример вида, реализующего обращение к контроллеру ASP.NET Web API

```
@{
    ViewBag.Title = "Графики";

    //интервал обновления - 3 сек.
    int refreshInterval = 3000;
    string refreshIntervalStr = refreshInterval.ToString();

    //uri для вызова Web API контроллера
    string uri = @"/api/serverrandom";
}

<h2>Графики</h2>

<div class="panel panel-primary">
  <div class="panel-heading">График по 10 последним точкам</div>
  <div class="panel-body">
        <div id="div_graph_current" style="width:100%; height:333px;"></div>
  </div>
</div>

<div class="panel panel-primary">
  <div class="panel-heading">График по всем точкам</div>
  <div class="panel-body">
        <div id="div_graph_total" style="width:100%; height:333px;"></div>
  </div>
</div>

<div class="panel panel-primary">
  <div class="panel-heading">График по всем точкам с прокруткой</div>
  <div class="panel-body">
        <div id="div_graph_total_roll" style="width:100%; height:333px;"></div>
  </div>
</div>

@section Scripts {
@Scripts.Render("~/bundles/dygraph")

<script type="text/javascript">

//----------------------------------------
// Преобразование даты-времени в тип Date из строки
//----------------------------------------
function ParseDateTime(dt)
{
    var year = dt.substring(0, 4);
    var month = dt.substring(5, 7);
    var day = dt.substring(8, 10);
    var hour = dt.substring(11, 13);
    var mnt = dt.substring(14, 16);
    var sec = dt.substring(17, 19);

    var Result = new Date(year, month-1, day, hour, mnt, sec, 0);
    return Result;
}

    //Признак первичного заполнения массива с накоплением
```

```
var firstTotalFlag = false;

//Данные с накоплением
var dataTotal = [];

//Текущие данные
var dataCurrent = [];

//Графики
var g_current = new Dygraph(document.getElementById("div_graph_current"), dataCurrent,
                        {
                            drawPoints: true,
                            valueRange: [0, 102],
                            labels: ['Время', 'Значение'],
                            'Значение': { fillGraph: true }
                        });

var g_total = new Dygraph(document.getElementById("div_graph_total"), dataTotal,
                        {
                            drawPoints: true,
                            valueRange: [0, 102],
                            labels: ['Время', 'Значение'],
                            'Значение': { fillGraph: true }
                        });

var g_total_roll = new Dygraph(document.getElementById("div_graph_total_roll"), dataTotal,
                        {
                            drawPoints: true,
                            valueRange: [0, 102],
                            labels: ['Время', 'Значение'],
                            'Значение': { fillGraph: true },
                            colors: ['#3333FF'],
                            showRangeSelector: true
                        });

//Uri для доступа к Web API контроллеру
var FullUri = "http://" + window.location.host + '@uri';

//Функция setIntervalGraph вызывается после загрузки страницы
$(document).bind("ready", setIntervalGraph);

function setIntervalGraph()
{
    //Функция RefreshProgress вызывается через заданный интервал времени
    RefreshGraph();
    setTimeout(setIntervalGraph, @refreshIntervalStr );
}

function RefreshGraph()
{
    //Отправка Ajax-запроса о состоянии индикатора прогресса
    $.ajax({
        url: FullUri,

        //функция вызывается после получения ответа на запрос
        success: function (data) {

            //data - массив данных, полученный с сервера

            //Очистка массива текущих данных
            dataCurrent.length = 0;

            //Заполнение массива текущих данных
            for (var i = 0; i < data.length; i++)
            {
```

```
                    var element = data[i];
                    var x = ParseDateTime(element.x);
                    var y = element.y;
                    dataCurrent.push([x, y]);
                }

                if(!firstTotalFlag)
                {
                    //При первом получении данных в массив с накоплением добавляются все
данные
                    dataTotal.length = 0;
                    for (var i = 0; i < data.length; i++)
                    {
                        var element = data[i];
                        var x = ParseDateTime(element.x);
                        var y = element.y;
                        dataTotal.push([x, y]);
                    }

                    firstTotalFlag = true;
                }
                else
                {
                    //Добавление последнего элемента в массив с накоплением
                    var lastElement = data[data.length - 1];
                    var x = ParseDateTime(lastElement.x);
                    var y = lastElement.y;
                    dataTotal.push([x, y]);
                }

                //Изменение графиков при изменении массивов данных
                g_current.updateOptions({ 'file': dataCurrent });
                g_total.updateOptions({ 'file': dataTotal });
                g_total_roll.updateOptions({ 'file': dataTotal });

            }
        });
    }

</script>
}
```

8 Контрольные вопросы

8.1 Лабораторная работа 1

1. Для чего предназначен язык разметки HTML?
2. Как сформировать список в HTML?
3. Как сформировать изображение в HTML?
4. Как сформировать таблицу в HTML?
5. Как сформировать фрейм в HTML?
6. Как сформировать гипертекстовую ссылку в HTML?
7. Как сформировать форму в HTML?

8.2 Лабораторная работа 2

1. Для чего используется библиотека Twitter Bootstrap?
2. Как подключить библиотеку Twitter Bootstrap локально и с удаленного сервера?
3. Как с использованием библиотеки Twitter Bootstrap реализовать таблицу?
4. Как с использованием библиотеки Twitter Bootstrap реализовать элементы формы?
5. Как с использованием библиотеки Twitter Bootstrap реализовать панель навигации?
6. Как с использованием библиотеки Twitter Bootstrap реализовать выпадающие списки кнопок?
7. Как с использованием библиотеки Twitter Bootstrap реализовать индикатор прогресса?

8.3 Лабораторная работа 3

1. Для чего предназначена библиотека LINQ to Objects?
2. Какие структуры данных могут являться источниками данных для библиотеки LINQ to Objects?
3. Как реализовать связь один-ко-многим в библиотеке LINQ to Objects?
4. Как реализовать связь много-ко-многим в библиотеке LINQ to Objects?
5. Как реализовать проверку условия в библиотеке LINQ to Objects?
6. Как реализовать сортировку в библиотеке LINQ to Objects?
7. Как реализовать группировку в библиотеке LINQ to Objects?
8. Как в библиотеке LINQ to Objects с использованием группировки реализовать проверку условия, что строковые данные всех записей (или только одной записи) начинаются с определенного символа?

8.4 Лабораторная работа 4

1. Что такое Entity Framework, LINQ to Entities, каким образом возможно их совместное использование?
2. Как создать модель данных в Entity Framework?
3. Какие структуры данных могут являться источниками данных для библиотеки LINQ to Entities?
4. Что такое навигационные свойства (navigation properties) и для чего они используются?
5. Как реализовать связь один-ко-многим в библиотеке LINQ to Entities?
6. Как реализовать связь много-ко-многим в библиотеке LINQ to Entities?
7. Как реализовать проверку условия в библиотеке LINQ to Entities?
8. Как реализовать сортировку в библиотеке LINQ to Entities?
9. Как реализовать группировку в библиотеке LINQ to Entities?
10. Как в библиотеке LINQ to Entities с использованием группировки реализовать проверку условия, что строковые данные всех записей (или только одной записи) начинаются с определенного символа?

8.5 Лабораторная работа 5

1. Как в Visual Studio создать проект ASP.NET MVC?
2. Что такое контроллер?
3. Что такое вид (view)?
4. Для чего предназначена технология Razor?
5. Для чего используется механизм scaffolding?
6. Как с помощью механизма scaffolding создать макет приложения ASP.NET MVC, позволяющий добавлять, редактировать и удалять данные?
7. Как реализовать контроллер и вид, формирующий выборку данных из модели данных Entity Framework в виде HTML-таблицы?

8.6 Лабораторная работа 6

1. Для чего используется технология ASP.NET Web API?
2. Каким образом можно совместно применить обычный контроллер, ASP.NET Web API контроллер и вид для формирования динамического графика?
3. Как реализовать AJAX-запрос к ASP.NET Web API контроллеру с использованием библиотеки jQuery?
4. Как с использованием ASP.NET Web API реализовать работающий индикатор процесса Twitter Bootstrap?
5. Как с использованием ASP.NET Web API реализовать динамически обновляющийся график?

9 Литература

1. Нейгел К., Ивьен Б., Глинн Д., Уотсон К. C# 4.0 и платформа .NET 4 для профессионалов. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2011. – 1440 с.
2. Фримен А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов. : Пер. с англ. – М. : ООО «И.Д. Вильямс», 2013. – 688 с.
3. Дронов В. А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. — СПб.: БХВ-Петербург, 2011. — 416 с.
4. Флэнаган Д. JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ Плюс, 2008. – 992 с.