**Лабораторная работа 1. Введение в разработку ASP.NET MVC Web Application**

В этой работе вы создадите простой проект на основе контроллера и модели, изучите основные элементы веб-приложения. Более подробно язык программирования С# и шаблон MVC будет рассматриваться в следующих работах.

***Упражнение 1. Создание веб-приложения ASP.NET MVC на основе контроллера***

В этом упражнении вы создадите веб-приложение, содержащее только контроллер, который будет содержать метод действия передающий информацию в браузер для ее отображения.

1. Запустите Visual Studio.

2. Выберите **File** **New Project.**

3. В открывшемся диалоговом окне **New Project** (Создание проекта) выберите в разделе Visual C# шаблоны Web, в списке версий .NET Framework оставьте значение по умолчанию и выберите тип проекта ASP.NET Web Application (как показано на рисунке 1.1).



*Рис.1.1 Создание проекта*

4. Назовите новый проект **WebMVCR1** и нажмите кнопку ОК, чтобы продолжить.

5. В следующем окне выбора типа шаблона (см. рис. 1.2) укажите **Empty** (Пустой) и в перечне флажков добавления папок и ссылок выберите **MVC** – будет создан проект с базовой структурой папок, но без файлов, необходимых для создания MVC приложений.



*Рис. 1.2 Выбор шаблона приложения*

6. В окне **Solution Explorer** (Обозреватель решений) изучите стандартную структуру MVC проекта. Поскольку был выбран пустой шаблон проекта и приложение ничего не содержит, запускать приложение пока что бессмысленно, будет ошибка 404 Not Found.

// Выполнено

**Создание контролера**

В архитектуре MVC входящие запросы обрабатываются контроллерами, которые являются классами (как правило, наследуются от System.Web.Mvc.Controller). В MVC контроллеры находятся в папке **Controllers**, которая была создана при создании проекта.

Каждый открытый метод в контроллере представляет собой метод действия, то есть его можно вызвать из Интернет через некоторый URL, чтобы выполнить действие.

В этом упражнении вы создадите в контроллере метод действия, который возвратит в браузер строку.

Для добавления пустого контроллера выполните следующие действия:

1. В контекстном меню папки **Controllers** выберите команду **Add** (Добавить) **Controller** (Контроллер).

2. В открывшемся окне добавления шаблона выберите пустой контроллер (см. рис. 1.3) и добавьте его.



*Рис. 1.3 Выбор шаблона контроллера*

3. В окне **Add Controller (**Добавление контролера**)** укажите имя контроллера **HomeController** (рис. 1.4) и нажмите кнопку добавления контроллера:



*Рис. 1.4 Указание имени контроллера*

4. Изучите код класса **HomeController**, обратите внимание на метод действия Index().

5. Для тестирование контроллера закомментируйте метод Index() и добавьте другой вариант метода, который возвращает строку приветствия в зависимости от времени дня:

public string Index()

{

int hour = DateTime.Now.Hour;

string Greeting = hour < 12 ? "Доброе утро" : "Добрый день";

return Greeting;

}

Операция x ? y : z вычисляет значение y, если значение x равно true, и значение z, если значение x равно false.

6. Запустите проект, выбрав в меню **Debug** команду **Start Debugging**. Браузер должен отобразить результат метода действия Index() – строку.

// Почему-то имя класса было HomeControllerController, а когда вручную изменил его на HomeController – все заработало и при запуске отобразило – строку "Добрый день". Потом понял – назвал неправильно, думая, что Controller – это расширение.

**Тестирование маршрутизации**

MVC-приложения используют систему маршрутизации (роутинговую систему) ASP.NET, которая решает, как URL-адреса понимают конкретные контроллеры и действия. При создании MVC-проекта среда разработки ASP.NET добавляет некоторые роуты по умолчанию.

В отличие от традиционных приложений ASP.NET, URL-адреса MVC не соответствуют физическим файлам. Каждый метод действия имеет свой URL, и MVC использует систему маршрутизации ASP.NET чтобы перевести эти URL в действия.

1. Для тестирования маршрутизации введите в адресной строке браузера ссылки, направленные на метод Index контроллера **HomeController** и после каждой обновите страницу:

https://localhost:44360/Home

https://localhost:44360//Home/Index

// Просто добавить Home и Home Index к существующему при запуске через WIDAUT DEBUGGING в строке локалу и скопировать ВСЮ строку. Потом строку можно запускать из ЛЮБОГО браузера – тоже выдает строку "Добрый день"

2. Проверьте, что в обоих случаях браузер отображает ту же самую строку приветствия.

// Получилось.

Это объясняется тем, что, когда браузер запрашивает адрес сайта, он получает выходные данные метода Index контроллера **HomeController**.

3. Переменяйте имя класса контроллера, например, вместо HomeController укажите MyController**.**

4. Повторно запустите приложение, в этом случае будет ошибка, так как система маршрутизации не нашла нужный маршрут.

// Получилось следующее

# Ошибка сервера в приложении '/'.

*Не удалось найти данный ресурс.*

**Описание:**HTTP 404. Возможно, искомый ресурс (или одна из его зависимостей) удален, получил другое имя или временно недоступен.  Просмотрите следующий URL-адрес и проверьте, что он введен правильно. //

5. Для исправления ошибки откройте находящийся в папке App\_Start файл RouteConfig и исправьте значение маршрута по умолчанию:

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "My", action = "Index", id = UrlParameter.Optional }

);

}

}

6. Повторно запустите приложение, в этом случае откроется требуемая страница, так как система маршрутизации нашла нужный маршрут.

// Так как поменяли значения в двух файлах - HomeController на MyController и "Home" на "My"

7. Теперь измените в классе контроллера имя метода Index() на, например, Start() и внесите соответствующие корректировку в класс **RouteConfig.**

8. Запустите приложение и проверьте ее работу.

// Получилось. Выдает строку "Добрый день"

***Упражнение 2. Передача параметров в контроллер***

В этом упражнении вы узнаете, как передавать параметры контроллеру с помощью URL-адреса и строки запроса.

При отправке GET-запроса значения передаются через строку запроса. Стандартный get-запрос принимает примерно следующую форму:

/метод?параметр1=значение1&параметр2=значение2

1. Откройте класс **HomeController**.

2. Измените метод Index(), добавив передаваемый ему строковый параметр, а в теле метода присвойте строковой переменной сообщение с передаваемым параметром:

public string Index(**string hel**)

{

int hour = DateTime.Now.Hour;

string Greeting = hour < 12 ? "Доброе утро" : "Добрый день" + ", " **+ hel**;

return Greeting;

}

// чтобы заработало пришлось в RouteConfig обратно поменять "Start" на "Index", что было изменено в предыдущем упражнении

3. Запустите приложение. Должна загрузиться страница Home.

// Загрузилось "Добрый день" на странице [https://localhost:44360](https://localhost:44360/)

4. Для проверки работы метода действия, принимающий параметр измените URL на следующий адрес (port – номер вашего порта) и учтите, что теперь имя контроллера – MyController:

localhost:port/My/Index?hel=Иван

// Получилась такая строка:

<https://localhost:44360/My/Index?hel=Иван>

Обновите страницу, должна появиться строка с переданным параметром.

// результат - Добрый день, Иван

***Упражнение 3. Реализация взаимодействия контроллера и модели***

Контроллер, реализованный в предыдущих упражнениях помимо обработки входных данных и вывода в браузер занимался и несвойственным ему действием – рассчитывал значение переменной, т.е. обрабатывал некую логику. В этом упражнении вы добавите в созданное веб-приложение, содержащее контроллер, модель, в которую будет перенесена логика приложения.

**Реализация модели**

Модель содержит данные и алгоритмы, которые будут определять логику приложения.

1. Добавьте в папку **Models** класс **ModelClass**.

2. Добавьте в класс статический метод и добавьте в его код расчета приветствия. В результате должно получиться следующее:

public class ModelClass

{

public static string ModelHello()

{

int hour = DateTime.Now.Hour; 8

string Greeting = hour < 12 ? "Доброе утро" : "Добрый день";

return Greeting;

}

}

3. В методе контроллера Index(string hel) присвойте переменной строкового типа результат вызова метода модели ModelHello():

public string Index(string hel)

{

string Greeting = ModelClass.ModelHello() + ", " + hel;

return Greeting;

}

4. Запустите приложение. Протестируйте работу метода действия, принимающий параметр, функциональность не должна измениться.

// результат - Добрый день