[^]

Песочница/Практика2018/Методические указания по выполнению практического задания

Содержание

Методические указания по выполнению практического задания

Введение

В данном руководстве описан пошаговый процесс создания ASP.NET MVC приложения, реализующего функционал сайта-блога "Клуб любителей мотоспорта".

В руководстве содержаться примеры кода. Реализация примеров имеет отличия от описанного в спецификации функционала. Исправить расхождение между приведенной реализацией и постановкой задачи предлагается самостоятельно.

Создание проекта

[^]

1. В среде MS Visual Studio создать проект ASP.NET MVC Web Application (.NET Framework). Наименование проекта задать как BikerBlog.
2. В качестве используемой технологии выбрать MVC, в качестве механизма авторизации выбрать "No authentication"
3. Установить проект BikerBlog как стартовый в решении.
4. Для проекта установить пакет Entity Framework.

Определение моделей и контекста работы с базой данных

При реализации проекта используется два вида моделей: описание сущностей, хранящихся в базе данных и модели обмена данными между контроллерами и представлениями системы.

В папке Models создадим подпапки DBModel, в которой будут располагаться описание сущностей БД и ViewModel, в которой расположим модели обмена данными.

Создание моделей сущности БД

В папке **DBModel** создадим классы:

1. **User** - описание сущности пользователя системы, описывающий атрибуты "никнейм", пол, дата рождения, другие контактные данные, а также фотографию пользователя и информацию о мотоцикле. Пример реализации класса **User**:

```
1     /// <summary>
2     /// Сущность пользователя системы, хранящаяся в БД.
3     /// </summary>
4     public class User
5     {
6         /// <summary>
7         /// Ключ сущности.
8         /// </summary>
9         [Key]
10        public int Id { get; set; }
11
12        #region Информация идентифицирующая пользователя.
13
14        /// <summary>
15        /// Никнейм.
16        /// </summary>
17        public string Nickname { get; set; }
```

```
18
19          /// <summary>
20          /// e-mail.
21          /// </summary>
22          public string Email { get; set; }
23
24          /// <summary>
25          /// Пароль.
26          /// </summary>
27          public string Password { get; set; }
28
29          #endregion
30
31          #region Персональные данные
32
33          /// <summary>
34          /// ФИО.
35          /// </summary>
36          public string Name { get; set; }
37
38          /// <summary>
39          /// Пол.
40          /// </summary>
41          public bool? Gender { get; set; }
42
43          /// <summary>
44          /// Дата рождения.
45          /// </summary>
46          public DateTime? Birthday { get; set; }
47
48          /// <summary>
49          /// Страна.
50          /// </summary>
51          public string Country { get; set; }
52
53          /// <summary>
54          /// Город/населённый пункт.
55          /// </summary>
56          public string City { get; set; }
57
58          #endregion
59
```

[^]

```
60          #region Увлечения
61
62          /// <summary>
63          /// Увлечения.
64          /// </summary>
65          public string Hobbie { get; set; }
66
67          #endregion
68
69          #region Контактная информация
70
71          /// <summary>
72          /// Телефон.
73          /// </summary>
74          public string Phone { get; set; }
75
76          /// <summary>
77          /// Скайп.
78          /// </summary>
79          public string Skype { get; set; }
80
81          /// <summary>
82          /// Прочая информация
83          /// </summary>
84          public string Information { get; set; }
85
86          /// <summary>
87          /// Мотоцикл.
88          /// </summary>
89          #endregion
90
91          #region Фотографии
92          /// <summary>
93          /// Фото
94          /// </summary>
95          public string PhotoUrl { get; set; }
96          #endregion
97
98          /// <summary>
99          /// Мотоцикл
100         /// </summary>
```

```
101         public virtual Bike Bike { get; set; }
102     }
```

[^]

Фотографию пользователя будем хранить на сайте. В классе **User** указывается ссылка, по которой можно загрузить файл фотографии (поле **PhotoUrl**).

Информацию о мотоцикле будем хранить в отдельной сущности **Bike.**

2. Класс **Bike** - описание сущности "мотоцикл". Класс описывает атрибуты: описание мотоцикла, максимальная скорость, объем двигателя и прочие характеристики мотоцикла. Пример реализации класса:

```
1     /// <summary>
2     /// Сущность мотоцикла, хранящаяся в БД.
3     /// </summary>
4     public class Bike
5     {
6         /// <summary>
7         /// Ключ сущности.
8         /// </summary>
9         [Key]
10        public int Id { get; set; }
11
12        /// <summary>
13        /// Описание мотоцикла.
14        /// </summary>
15        public string Name { get; set; }
16
17        /// <summary>
18        /// Максимальная скорость.
19        /// </summary>
20        public int MaxSpeed { get; set; }
21
22        /// <summary>
23        /// Объём двигателя.
24        /// </summary>
25        public int Volume { get; set; }
26
27        /// <summary>
28        /// Прочие характеристики.
29        /// </summary>
30        public string Character { get; set; }
31    }
```

[^]

3. Класс **Post** описывает сущность публикации пользователя. Класс описывает атрибуты: дата публикации, тема, описание публикации, фотографию и автора. Автор - поле с типом User. Пример реализации класса:

```
1     /// <summary>
2     ///  Сущность публикации, хранящаяся в БД.
3     /// </summary>
4     public class Post
5     {
6         /// <summary>
7         /// Ключ сущности.
8         /// </summary>
9         [Key]
10        public int Id { get; set; }
11
12        /// <summary>
13        /// Дата публикации поста
14        /// </summary>
15        public DateTime Data { get; set; }
16
17        /// <summary>
18        /// Тема.
19        /// </summary>
20        public string Theme { get; set; }
21
22        /// <summary>
23        /// Описание поста.
24        /// </summary>
25        public string Description { get; set; }
26
27        /// <summary>
28        /// Фото.
29        /// </summary>
30        public string PhotoUrl { get; set; }
31
32        /// <summary>
33        /// Автор.
34        /// </summary>
35        public virtual User Author { get; set; }
36    }
```

Во всех классах используется дополнительное свойство **Id** - идентификатор сущности. При хранении сущностей в базе данных в таблицах это поле предстанет первичным ключём таблицы. [^]

В папке **ViewModel** создадим классы: **LoginVewModel, PostViewModel, ProfileVewModel, RegisterViewModel**, описывающие модели обмена данными между контроллерами и представлениями. В классах с помощью аннотаций укажем обязательность полей модели(**Required**), информацию по отображению названия поля в представлении (**Display**), а также информацию по ограничением, накладываемых на значения полей.

Примеры классов моделей:

```
1     /// <summary>
2     /// Вью-модель логина.
3     /// </summary>
4     public class LoginVewModel
5     {
6         /// <summary>
7         /// Ник или почта пользователя.
8         /// </summary>
9         [Display(Name = "EMAIL или NICK")]
10        [Required(ErrorMessage = "Ошибка в EMAIL или NICK"), MaxLength(30)]
11        public string NickOrEmail { get; set; }
12
13        /// <summary>
14        /// Пароль.
15        /// </summary>
16        [Display(Name = "Пароль")]
17        [Required(ErrorMessage = "Ошибка в пароле"), MaxLength(30)]
18        public string Password { get; set; }
19
20        /// <summary>
21        /// Чекбокс запомнить.
22        /// </summary>
23        [Display(Name = "Запомнить")]
24        public bool RememberMe { get; set; }
25    }
```

```
1     /// <summary>
2     /// Вью-модель добавления поста.
3     /// </summary>
4     public class PostViewModel
5     {
```

```
        /// <summary>
        /// Тема.
        /// </summary>
        [Display(Name = "Тема")]
        [Required(ErrorMessage = "Указание темы обязательно"), MaxLength(40)]
        public string Theme { get; set; }

        /// <summary>
        /// Описание поста.
        /// </summary>
        [Display(Name = "Описание")]
        [Required(ErrorMessage = "Указание описания поста обязательно")]
        public string Description { get; set; }

        /// <summary>
        /// Фото
        /// </summary>
        [Display(Name = "Фото")]
        [Required(ErrorMessage = "Указание фото обязательно")]
        [DataType(DataType.Upload)]
        public HttpPostedFileBase PostImage { get; set; }
    }
```

```
    /// <summary>
    /// Вью-модель профиля пользователя.
    /// </summary>
    public class ProfileVewModel
    {
        #region Информация идентифицирующая пользователя.

        /// <summary>
        /// Идентификатор профиля.
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Никнейм для отображения на форме.
        /// </summary>
        public string Nickname { get; set; }

        /// <summary>
```

```
19          /// EMail.
20          /// </summary>
21          [Display(Name = "Email")]
22          [Required(ErrorMessage = "Указание адреса электронной почты обязательно")]
23          [EmailAddress(ErrorMessage = "Не верно указан адрес электронной почты")]
24          public string Email { get; set; }
25
26          #endregion
27
28          #region Персональные данные
29
30          /// <summary>
31          /// ФИО.
32          /// </summary>
33          [Display(Name = "ФИО")]
34          [Required(ErrorMessage = "Указание ФИО обязателено"), MaxLength(50)]
35          public string Name { get; set; }
36
37          /// <summary>
38          /// Пол.
39          /// </summary>
40          [Display(Name = "Пол")]
41          [Required(ErrorMessage = "Указание пола обязательно")]
42          public bool? Gender { get; set; }
43
44          /// <summary>
45          /// Дата рождения.
46          /// </summary>
47          [Display(Name = "Дата рождения")]
48          [Required(ErrorMessage = "Указание даты рождения обязательно")]
49          [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]
50          public DateTime? Birthday { get; set; }
51
52          /// <summary>
53          /// Страна.
54          /// </summary>
55          [Display(Name = "Страна")]
56          [Required(ErrorMessage = "Указание страны обязательно"), MaxLength(20)]
57          public string Country { get; set; }
58
59          /// <summary>
60          /// Город/населённый пункт.
```

```
61          /// </summary>
62          [Display(Name = "Город/населённый пункт")]
63          [Required(ErrorMessage = "Указание города/населенного пункта обязательно"), MaxLength(20)]
64          public string City { get; set; }
65
66          #endregion
67
68          #region Увлечения
69
70          /// <summary>
71          /// Увлечения.
72          /// </summary>
73          [Display(Name = "Увлечения")]
74          [MaxLength(1000)]
75          public string Hobbie { get; set; }
76
77          #endregion
78
79          #region Контактная информация
80
81          /// <summary>
82          /// Телефон.
83          /// </summary>
84          [Display(Name = "Телефон")]
85          [DataType(DataType.PhoneNumber)]
86          [RegularExpression(@"^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$", ErrorMessage = "Не верный телефонный номер")]
87          public string Phone { get; set; }
88
89          /// <summary>
90          /// Скайп.
91          /// </summary>
92          [Display(Name = "Скайп")]
93          public string Skype { get; set; }
94
95          /// <summary>
96          /// Прочая информация
97          /// </summary>
98          [Display(Name = "Прочая информация")]
99          public string Information { get; set; }
100
101         /// <summary>
102         /// Мотоцикл.
```

```
103         /// </summary>
104         #endregion
105 
106         #region Фотографии
107         /// <summary>
108         /// Фото
109         /// </summary>
110         public String Photo { get; set; }
111 
112         /// <summary>
113         /// Фото (файл для загрузки)
114         /// </summary>
115         [DataType(DataType.Upload)]
116         public HttpPostedFileBase UserImageUpload { get; set; }
117         #endregion
118 
119         #region Информация о мотоцикле
120 
121         /// <summary>
122         /// Описание мотоцикла.
123         /// </summary>
124         [Display(Name = "Байк")]
125         [Required, MaxLength(30)]
126         public string BikeName { get; set; }
127 
128         /// <summary>
129         /// Максимальная скорость.
130         /// </summary>
131         [Display(Name = "Максимальная скорость")]
132         [Required, Range(0, 999)]
133         public int BikeMaxSpeed { get; set; }
134 
135         /// <summary>
136         /// Объём двигателя.
137         /// </summary>
138         [Display(Name = "Объём двигателя, см^3")]
139         [Required, Range(0, 9999)]
140         public int BikeVolume { get; set; }
141 
142         /// <summary>
143         /// Прочие характеристики.
144         /// </summary>
```

```
        [Display(Name = "Прочие характеристики")]
        public string BikeCharacter { get; set; }

        #endregion
    }
```

```
    /// <summary>
    /// Вью-модель для регистрации.
    /// </summary>
    public class RegisterViewModel
    {
        /// <summary>
        /// Ник.
        /// </summary>
        [Display(Name = "Придумайте никнейм")]
        [Required(ErrorMessage = "Ошибка никнейме"), MaxLength(30)]
        public string Nick { get; set; }

        /// <summary>
        /// EMail.
        /// </summary>
        [Display(Name = "Введите свой EMAIL")]
        [Required(ErrorMessage = "Ошибка в почте")]
        [EmailAddress(ErrorMessage = "Ошибка в почте")]
        public string Email { get; set; }

        /// <summary>
        /// Пароль.
        /// </summary>
        [Display(Name = "Придумайте пароль")]
        [Required, MaxLength(30), MinLength(6)]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@$!%*?&])[A-Za-z\d$@$!%*?&]{6,30}", ErrorMessage = "Ошибка в пароле")]

        public string Password { get; set; }

        /// <summary>
        /// Пароль.
        /// </summary>
        [Display(Name = "Подтвердите пароль")]
        [Required, MaxLength(30), MinLength(6)]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@$!%*?&])[A-Za-z\d$@$!%*?&]{6,30}", ErrorMessage = "Ошибка в подтверждении пароля")]
```

```
36          public string PasswordConfirmation { get; set; }
37      }
```

[^]

Создание контекста и инициализатора базы данных

В папке DAL создадим класс контекста BikerBlogDBContext и инициализатора БД.

Примеры реализации классов:

```
1       /// <summary>
2       /// Контекст работы с БД.
3       /// </summary>
4       public class BikerBlogDBContext : DbContext
5       {
6           public BikerBlogDBContext() : base("BikerBlogDBContext")
7           {
8           }
9           protected override void OnModelCreating(DbModelBuilder modelBuilder)
10          {
11              modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
12          }
13          public DbSet<User> Users { get; set;}
14          public DbSet<Bike> Bykes { get; set; }
15          public DbSet<Post> Posts { get; set; }
16      }
```

```
1       /// <summary>
2       /// Инициализатор БД.
3       /// </summary>
4       public class BikerBlogDbInitializor :// DropCreateDatabaseAlways<BikerBlogDBContext>
5       DropCreateDatabaseIfModelChanges<BikerBlogDBContext>
6       {
7           protected override void Seed(BikerBlogDBContext context)
8           {
9           }
10      }
```

В файле **Global.asax** укажем, что БД в нашем приложении нужно инициализировать:

```
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

            //Инициализируем БД.
            Database.SetInitializer(new BikerBlogDbInitializor());
        }
    }
```

Создание вспомогательного механизма работы с изображениями

Изображения, загружаемые пользователем, хранятся в папке **/Content/Images/Uploads/** (подпапки папки Images и **Uploads** необходимо создать самостоятельно).

Создадим вспомогательный класс **ImageSaveHelper**, осуществляющий логику сохранения загруженного файла в папку Uploads. Класс расположим в папке **Infrastructure**.

Пример реализации класса:

```
namespace BikerBlog.Infrastructure
{
    /// <summary>
    /// Хелпер класс для работы с изображениями.
    /// </summary>
    public class ImageSaveHelper
    {
        /// <summary>
        /// Сохраняет изображение и возвращает путь до него.
        /// </summary>
        /// <param name="image"></param>
        /// <returns></returns>
        public static string SaveImage(HttpPostedFileBase image)
        {
            var filename = image.FileName;
            var filePathOriginal = HostingEnvironment.MapPath("/Content/Images/Uploads/");
```

```
            string savedFileName = Path.Combine(filePathOriginal ?? throw new InvalidOperationException(), filename);
            image.SaveAs(savedFileName);

            return $"/Content/Images/Uploads/{filename}";
        }
    }
}
```

[^]

Создание контроллеров

Создадим следующие контроллеры для организации работы приложения. В каждом контроллере напишем методы, осуществляющие выборки данных из БД и возвращающие соответствующие представления. Также, опишем методы, получающие модель из представлений и сохранение полученных данных в БД. Для работы с базой данных объявим и инициализируем контекст **BikerBlogDBContext** как приватное поле класса. Для работы с загруженными изображениями будем использовать описанных выше вспомогательный класс **ImageSaveHelper**.

1. **WelcomeController -** контроллер первой страницы приложения. Авторизация пользователя, выход из системы. Пример реализации контроллера:

```
/// <summary>
    /// Контроллер первой страницы приложения. Авторизация пользователя, выход из системы.
    /// </summary>
    public class WelcomeController : Controller
    {
        /// <summary>
        /// Контекст БД.
        /// </summary>
        private readonly BikerBlogDBContext _context = new BikerBlogDBContext();
        // GET: Welcome
        public ActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Login(LoginVewModel model)
        {
            if (ModelState.IsValid)
            {
                var existanceUser = _context.Users.FirstOrDefault(user =>
                    (user.Nickname == model.NickOrEmail || user.Email == model.NickOrEmail) &&
                    user.Password == model.Password);
```

```
                if (existanceUser!=null)
                {
                    // Авторизуем пользователя и переходим в ленту новостей.
                    Session["UserId"] = existanceUser.Id.ToString();
                    Session["UserNick"] = existanceUser.Nickname;
                    FormsAuthentication.SetAuthCookie(model.NickOrEmail, model.RememberMe);

                    return RedirectToAction("Index", "Feed");
                }
            }
            return View("Index", model);
        }

        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            //TODO: Очищать переменные сессии
            return RedirectToAction("Index");
        }
    }
```

2. **RegisterController** - контроллер регистрации пользователя в системе Пример реализации контроллера:

```
    /// <summary>
    /// Контроллер регистрации пользователя в системе.
    /// </summary>
    public class RegisterController : Controller
    {
        /// <summary>
        /// Контекст БД.
        /// </summary>
        private readonly BikerBlogDBContext _context = new BikerBlogDBContext();
        // GET: Register
        public ActionResult Index()
        {
            var model = new RegisterViewModel();
            return View(model);
        }

        [HttpPost]
        public ActionResult Register(RegisterViewModel model)
```

```
19          {
20              if (ModelState.IsValid)
21              {
22                  // Проверяем, что пароли совпадают.
23                  if (model.Password != model.PasswordConfirmation)
24                  {
25                      ModelState.AddModelError(string.Empty, "Введенные пароли не совпадают");
26                  }
27
28                  var existenceUserNick =
29                      _context.Users.FirstOrDefault(user => user.Nickname == model.Nick);
30                  if (existenceUserNick != null)
31                  {
32                      ModelState.AddModelError(string.Empty, "Пользователь с таким никнеймом уже существует");
33                  }
34
35                  var existenceUserEmail =
36                      _context.Users.FirstOrDefault(user => user.Email == model.Email);
37                  if (existenceUserEmail != null)
38                  {
39                      ModelState.AddModelError(string.Empty, "Пользователь с такой почтой уже существует");
40                  }
41
42                  if (!ModelState.IsValid)
43                  {
44                      return View("Index", model);
45                  }
46
47                  // Регистрируем пользователя
48                  var newUser = new User
49                  {
50                      Nickname = model.Nick,
51                      Email = model.Email,
52                      Password = model.Password
53                  };
54                  _context.Users.Add(newUser);
55                  _context.SaveChanges();
56
57                  // Авторизуем пользователя и переходим в ленту новостей.
58                  Session["UserId"] = newUser.Id.ToString();
59                  Session["UserNick"] = newUser.Nickname;
60
```

```
                FormsAuthentication.SetAuthCookie(newUser.Name, false);
                return RedirectToAction("Edit", "Profile", new { id = newUser.Id });

            }
            return View("Index", model);
        }
    }
```

[^]

3. **ProfileController** - контроллер редактирования профиля пользователя. Изображение пользователя из представления будем получать как HttpPostedFileBase imageData. Для сохранения изображения воспользуемся классом **ImageSaveHelper**. Пример реализации контроллера:

```
/// <summary>
    /// Контроллер редактирования профиля пользователя.
    /// </summary>
    public class ProfileController : Controller
    {
        /// <summary>
        /// Контекст БД.
        /// </summary>
        private readonly BikerBlogDBContext _context = new BikerBlogDBContext();

        [HttpGet]
        public ActionResult Edit(int id)
        {
            var user = _context.Users.FirstOrDefault(c => c.Id == id);
            if (user != null)
            {
                // Создаем модель профиля из данных пользователя
                var model = ConvertorUserToProfile(user);

                return View(model);
            }

            return HttpNotFound();
        }


        [HttpPost]
        public ActionResult Edit(ProfileVewModel model, HttpPostedFileBase imageData)
        {
            if (ModelState.IsValid)
```

```
31          {
32              var user = _context.Users.FirstOrDefault(c => c.Id == model.Id);
33              if (user == null)
34              {
35                  return HttpNotFound();
36              }
37
38              FillUserDataFromProfileVm(model, ref user);
39              //Если обновили фото - загружаем его и обнавляем информацию о уго пути в БД.
40              if (imageData!=null)
41              {
42                  user.PhotoUrl = ImageSaveHelper.SaveImage(imageData);
43              }
44
45              _context.SaveChanges();
46          }
47
48          return View(model);
49      }
50
51
52      private static ProfileVewModel ConvertorUserToProfile(User user)
53      {
54          var model = new ProfileVewModel
55          {
56              Id = user.Id,
57              Nickname = user.Nickname,
58              Email = user.Email,
59              Name = user.Name,
60              Birthday = user.Birthday,
61              Gender = user.Gender,
62              Country = user.Country,
63              City = user.City,
64              Hobbie = user.Hobbie,
65              Phone = user.Phone,
66              Skype = user.Skype,
67              Information = user.Information,
68              Photo = user.PhotoUrl
69          };
70          if (user.Bike != null)
71          {
72              model.BikeCharacter = user.Bike.Character;
```

```
73                 model.BikeMaxSpeed = user.Bike.MaxSpeed;
74                 model.BikeName = user.Bike.Name;
75                 model.BikeVolume = user.Bike.Volume;
76             }
77 
78             return model;
79         }
80 
81         private void FillUserDataFromProfileVm(ProfileVewModel profile, ref User user)
82         {
83             user.Email = profile.Email;
84             user.Name = profile.Name;
85             user.Birthday = profile.Birthday;
86             user.Gender = profile.Gender;
87             user.Country = profile.Country;
88             user.City = profile.City;
89             user.Hobbie = profile.Hobbie;
90             user.Phone = profile.Phone;
91             user.Skype = profile.Skype;
92             user.Information = profile.Information;
93 
94 
95             var bike = user.Bike;
96             if (bike == null)
97             {
98                 bike = new Bike();
99                 user.Bike = bike;
100            }
101
102            bike.Character = profile.BikeCharacter;
103            bike.MaxSpeed = profile.BikeMaxSpeed;
104            bike.Name = profile.BikeName;
105            bike.Volume = profile.BikeVolume;
106        }
107    }
```

[^]

4. **FeedController** - контроллер работы с лентой новостей, добавления публикации. В качестве модели будем использовать тип ProfileVewModel, коллекцию постов передадим в представление через **ViewBag**.

Пример реализации контроллера:

```
1     /// <summary>
2     /// Контроллер работы с лентой новостей, добавления публикации.
3     /// </summary>
4     public class FeedController : Controller
5     {
6         private readonly BikerBlogDBContext _context = new BikerBlogDBContext();
7
8         // GET: Feed
9         public ActionResult Index()
10        {
11            //Передаем посты через ViewBag
12            var posts = _context.Posts.OrderByDescending(c=>c.Data).ToList();
13            ViewBag.Posts = posts;
14            return View();
15        }
16
17        [HttpPost]
18        public ActionResult Index(PostViewModel model, HttpPostedFileBase imageData)
19        {
20            var newPost = new Post
21            {
22                Data = DateTime.Now,
23                Theme = model.Theme,
24                Description = model.Description
25            };
26
27            // добавляем автора
28            var sessionUserId = Convert.ToInt32(Session["UserId"]);
29            var user = _context.Users.FirstOrDefault(c => c.Id == sessionUserId);
30            newPost.Author = user;
31            // добавляем картинку
32            if (imageData != null)
33            {
34                newPost.PhotoUrl = ImageSaveHelper.SaveImage(imageData);
35            }
36
37            _context.Posts.Add(newPost);
38            _context.SaveChanges();
39
40            var posts = _context.Posts.OrderByDescending(c => c.Data).ToList();
41            ViewBag.Posts = posts;
42
```

```
43             return View();
44         }
45     }
```

[^]

Создание представлений

Для каждого контроллера в папке **Views** создадим представления (см рисунок). Представления реализуем с использованием разметки **Razor** и системой **bootstrap.**

Стартовая страница

Модель представления - тип **LoginVewModel**. Для него на странице определим форму заполнения и проверки полей модели. Данные будем передавать в контроллер **Welcome** в метод **Login**.

Пример реализации представления стартовой страницы системы (**Welcome/Index.cstml**):

```
1 @model BikerBlog.Models.ViewModel.LoginVewModel
2
3 @{
4     ViewBag.Title = "Добро пожаловать";
5 }
6
7 @*Область логина*@
8 <div class="col-lg-6">
9     <h2>Вход</h2>
10
11     @using (Html.BeginForm("Login", "Welcome", FormMethod.Post))
12     {
13         @Html.AntiForgeryToken()
14
15         <div class="form-horizontal">
16
17             @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18             <div class="form-group">
19                 <div class="col-md-10">
20                     @Html.EditorFor(model => model.NickOrEmail, new { htmlAttributes = new { @class = "form-control", placeholder = "EMAIL или NICK" } })
21                     @Html.ValidationMessageFor(model => model.NickOrEmail, "", new { @class = "text-danger" })
22                 </div>
23             </div>
24
25             <div class="form-group">
```

```
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Password, new { htmlAttributes = new { @class = "form-control", placeholder = "Пароль", type="Password" } })
                    @Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                <div class="col-md-10">
                    @Html.CheckBoxFor(model => model.RememberMe, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.LabelFor(model => model.RememberMe)
                </div>
            </div>

            <div class="form-group">
                <input class="btn btn-block" type="submit" value="Войти" />
            </div>

        </div>
    }
</div>

@*Область регистрации*@
<div class="col-lg-6">
    <h2>Регистрация</h2>
    <div class="form-horizontal">

        @using (Html.BeginForm("Index", "Register", FormMethod.Get))
        {
            <p>
                Блог любителей мотоспорта Harley Blog - это возможность тысячам байкеров делиться своим опытом и яркими моментами из своей жизни. Здесь ты сможешь обрести новые знакомства с людьми, которые любят мотоциклы.

            </p>
            <div class="form-group">
                <div class="col-md-offset-2 col-md-10">
                    <input type="submit" value="Регистрация" class="btn btn-block" />
                </div>
            </div>
        }
    </div>
</div>
```

Регистрация пользователя

[^]

Модель представления - тип **RegisterViewModel**. Для него на странице определим форму заполнения и проверки полей модели. Данные будем передавать в контроллер **Register** в метод **Register**.

Пример реализации представления страницы регистрации пользователя (**Register/Index.cshtml**):

```
@model BikerBlog.Models.ViewModel.RegisterViewModel

<div class="col-lg-6">
    <h2>Регистрация</h2>

    @using (Html.BeginForm("Register", "Register", FormMethod.Post))
    {
        @Html.AntiForgeryToken()

        <div class="form-horizontal">
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            <div class="form-group">
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Nick, new { htmlAttributes = new { @class = "form-control", placeholder = "Придумайте никнейм" } })
                    @Html.ValidationMessageFor(model => model.Nick, "",  new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control", placeholder = "Введите свой EMAIL" } })
                    @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Password, new { htmlAttributes = new { @class = "form-control", placeholder = "Придумайте пароль", type = "Password" } })
                    @Html.ValidationMessageFor(model => model.Password, "", new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-10">
                    @Html.EditorFor(model => model.PasswordConfirmation, new { htmlAttributes = new { @class = "form-control", placeholder = "Подтвердите пароль", type =
"Password" } })
```

```
                    @Html.ValidationMessageFor(model => model.PasswordConfirmation, "", new { @class = "text-danger" })
                </div>
            </div>

            <div class="form-group">
                <input class="btn btn-block" type="submit" value="Зарегистрироваться" />
            </div>
        </div>

    }
</div>

<div class="col-lg-6">
    <h2>Сомневаешся?</h2>
    <p>
        Став членом общества мотолюбителей ты получишь возможность общаться с людьми, которые, вероятно, так же как и ты, увлечены мотоциклами.
        У тебя будет доступ к новостям байкеров всего мира, просматривая их публикации, ты можешь получать новый опыт и эмоции, узнавать о крутых локация и просто общаться.
        Так же ты сам можешь делиться своими мыслями и выступать в роли наставника для начинающих байкеров.
        Так что, если тебе интересна тема мотоспорта и мотоциклов в целом, добро пожаловать!
    </p>
</div>
<div class="col-lg-6">
    <h2>Никнейм</h2>

    <p>
        В никнейме должно быть не мене 2 символов, можно использовать: <br />
        - любые латинские буквы (a-z, A-Z); <br />
        - любые цифры (0-9);<br />
        - знак нижнего подчёркивания "_". <br />
        Примечание: никнейм должен содержать не менее 2 букв.
    </p>

</div>
<div class="col-lg-6">
    <h2>Пароль</h2>
    <p>
        Пароль должен быть не менее 6 символов, может содержать:<br>
        - любые латинские буквы (a-z, A-Z);<br>
        - любые цифры (0-9);<br>
        - спец. символы (!@@#$^&-_+=;:,.?|`~<>',).<br>
        Примечание: пароль не может состоять из букв одного регистра, одних цифр или спецсимволов, не должен содержать пробелы, не должен совпадать с никнеймом или именем почтового ящика.
```

```
75     </p>
76 </div>
```

[^]

Профиль пользователя

Модель представления - тип **ProfileVewModel**. Для него на странице определим форму заполнения и проверки полей модели. Данные будем передавать в контроллер **Profile** в метод **Edit**.

Пример реализации представления страницы регистрации пользователя (**Profile/Edit.cshtml**):

```
@model BikerBlog.Models.ViewModel.ProfileVewModel

@{
    ViewBag.Title = "Профиль";
}
@using (Html.BeginForm("Edit","Profile",FormMethod.Post, new { enctype = "multipart/form-data" }))
{
    @Html.AntiForgeryToken()

    @*ФОТО*@
    <div class="form-horizontal">
        <div class="row">
            <div class="panel panel-primary col-md-6 text-center" style="height: 370px">
                <div class="panel-body">
                    @if (Model.Photo != null) {
                        <img src="@Url.Content(@Model.Photo)" height="300"/>
                    }
                    else
                    {
                        <img src="~/Content/Images/default-user.png" height="300"/>
                    }
                    @if (Session["UserId"]!=null && Convert.ToInt32(Session["UserId"]) == Model.Id) {
                        //Даем изменять фото профиля только пользователю с этим профилем
                        <input name="imageData" type="file" accept="image/x-png,image/gif,image/jpeg" style="position: absolute;right:0;bottom: 0;"/>
                    }
                </div>
            </div>
            @*Персональные данные*@
            <div class="panel panel-primary col-md-6" style="height: 370px">
                <div class="panel-heading">Персональные данные</div>
                <div class="panel-body">
```

```
                <h2 style="text-align: center;">@Model.Nickname</h2>
                <div class="form-group">
                    @Html.LabelFor(model => model.Name, htmlAttributes: new {@class = "control-label col-md-4"})
                    @Html.EditorFor(model => model.Name, new {htmlAttributes = new {@class = "form-control"}})
                    @Html.ValidationMessageFor(model => model.Name, "", new {@class = "text-danger"})
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Gender, htmlAttributes: new {@class = "control-label col-md-4"})
                    @Html.DropDownListFor(model => model.Gender,
                        new SelectList(
                            new[]
                            {
                                new {Value = "", Text = "Выберите пол"},
                                new {Value = "true", Text = "Мужской"},
                                new {Value = "false", Text = "Женский"},
                            },
                            "Value",
                            "Text"
                            ),
                        new {@class = "form-control"}

                        )
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Birthday, htmlAttributes: new {@class = "control-label col-md-4"})
                    @Html.EditorFor(model => model.Birthday, new {htmlAttributes = new {@class = "form-control", type = "date"}})
                    @Html.ValidationMessageFor(model => model.Birthday, "", new {@class = "text-danger"})
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.Country, htmlAttributes: new {@class = "control-label col-md-4"})
                    @Html.EditorFor(model => model.Country, new {htmlAttributes = new {@class = "form-control"}})
                    @Html.ValidationMessageFor(model => model.Country, "", new {@class = "text-danger"})
                </div>
                <div class="form-group">
                    @Html.LabelFor(model => model.City, htmlAttributes: new {@class = "control-label col-md-4"})
                    @Html.EditorFor(model => model.City, new {htmlAttributes = new {@class = "form-control"}})
                    @Html.ValidationMessageFor(model => model.City, "", new {@class = "text-danger"})
                </div>
            </div>
        </div>
    </div>
    <div class="row">
```

```
74              @*Увлечения*@
75              <div class="panel panel-primary col-md-12">
76                  <div class="panel-heading">Увлечения</div>
77                  <div class="panel-body">
78                      @Html.TextAreaFor(model => model.Hobbie, 10, 100, new { htmlAttributes = new { @class = "form-control", style="width:100%"} })
79                  </div>
80              </div>
81          </div>
82          <div class="row">
83
84              @*ТТХ Мотоцикла*@
85              <div class="panel panel-primary col-md-6" style="height: 440px">
86                  <div class="panel-heading">ТТХ Мотоцикла</div>
87                  <div class="panel-body">
88                      <div class="form-group">
89                          @Html.LabelFor(model => model.BikeName, htmlAttributes: new { @class = "control-label col-md-3" })
90                          @Html.EditorFor(model => model.BikeName, new { htmlAttributes = new { @class = "form-control" } })
91                          @Html.ValidationMessageFor(model => model.BikeName, "", new { @class = "text-danger" })
92                      </div>
93                      <div class="form-group">
94                          @Html.LabelFor(model => model.BikeMaxSpeed, htmlAttributes: new { @class = "control-label col-md-3" })
95                          @Html.EditorFor(model => model.BikeMaxSpeed, new { htmlAttributes = new { @class = "form-control" } })
96                          @Html.ValidationMessageFor(model => model.BikeMaxSpeed, "", new { @class = "text-danger" })
97                      </div>
98                      <div class="form-group">
99                          @Html.LabelFor(model => model.BikeVolume, htmlAttributes: new { @class = "control-label col-md-3" })
100                         @Html.EditorFor(model => model.BikeVolume, new { htmlAttributes = new { @class = "form-control" } })
101                         @Html.ValidationMessageFor(model => model.BikeVolume, "", new { @class = "text-danger" })
102                     </div>
103                     <div class="form-group">
104                         @Html.LabelFor(model => model.BikeCharacter, htmlAttributes: new { @class = "control-label col-md-3" })
105                         @Html.TextAreaFor(model => model.BikeCharacter, 10, 40, new { htmlAttributes = new { @class = "form-control" } })
106                         @Html.ValidationMessageFor(model => model.BikeCharacter, "", new { @class = "text-danger" })
107                     </div>
108                 </div>
109             </div>
110             @*Контактные данные*@
111             <div class="panel panel-primary col-md-6" style="height: 440px">
112                 <div class="panel-heading">Контактные данные</div>
113                 <div class="panel-body">
114                     <div class="form-group">
115                         @Html.LabelFor(model => model.Phone, htmlAttributes: new { @class = "control-label col-md-3" })
```

```
                    @Html.EditorFor(model => model.Phone, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Phone, "", new { @class = "text-danger" })
                </div>

                <div class="form-group">
                    @Html.LabelFor(model => model.Skype, htmlAttributes: new { @class = "control-label col-md-3" })
                    @Html.EditorFor(model => model.Skype, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Skype, "", new { @class = "text-danger" })
                </div>

                <div class="form-group">
                    @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-3" })
                    @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
                </div>

                <div class="form-group">
                    @Html.LabelFor(model => model.Information, htmlAttributes: new { @class = "control-label col-md-3" })
                    @Html.TextAreaFor(model => model.Information, 10, 40, new { htmlAttributes = new { @class = "form-control" } })
                    @Html.ValidationMessageFor(model => model.Information, "", new { @class = "text-danger" })
                </div>
            </div>
        </div>
    </div>
@if (Session["UserId"] != null && Convert.ToInt32(Session["UserId"]) == Model.Id)
{
    //Даем изменять только свой профиль
    <div class="row">

        <div class="form-group col-md-12">
            <input class="btn btn-block" type="submit" value="Сохранить"/>
        </div>
    </div>
}
    </div>
}
```

Лента новостей

Модель представления - тип **PostViewModel,** также будем использовать тип BikerBlog.Models.DBModel.Post для отображения публикаций в ленте . На странице определим форму заполнения и проверки полей модели. Данные будем передавать в контроллер **Feed** в метод **Index**.

В разметке представления определим хелпер **@helper PostHelper(Post post)** для отображения каждой публикации в ленте. Хелпер будем использовать при перечисления коллекции публикаций из переданного в представлении **ViewBag.Posts**.

Пример реализации представления страницы регистрации пользователя (**Feed /Index.cshtml**):

```
@using BikerBlog.Models.DBModel
@model BikerBlog.Models.ViewModel.PostViewModel

@{
    ViewBag.Title = "Лента новостей";
}

@*Добавление новости*@
@if (Session["UserId"] != null)
{
    using (Html.BeginForm("Index", "Feed", FormMethod.Post, new { enctype = "multipart/form-data" }))
    {
        @Html.AntiForgeryToken()

        <div class="form-horizontal">

            <div class="panel panel-primary col-md-12">
                <div class="panel-heading">Добавить публикацию</div>
                <div class="panel-body">

                    <div class="row form-group">

                        @Html.LabelFor(model => model.Theme, htmlAttributes: new { @class = "control-label col-md-4" })
                        @Html.EditorFor(model => model.Theme, new { htmlAttributes = new { @class = "form-control" } })
                        @Html.ValidationMessageFor(model => model.Theme, "", new { @class = "text-danger" })
                    </div>

                    <div class="row form-group">
                        @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-4" })
                        @Html.TextAreaFor(model => model.Description, 10, 60, new { htmlAttributes = new { @class = "form-control", Style = "width:100%" } })
                        @Html.ValidationMessageFor(model => model.Description, "", new { @class = "text-danger" })
                    </div>

                    <div class="row">
                        @Html.LabelFor(model => model.PostImage, htmlAttributes: new { @class = "control-label col-md-4" })
                        <input name="imageData" type="file" accept="image/x-png,image/gif,image/jpeg" style="position: absolute; right: 0; bottom: 0;" />
```

```
                    </div>
                </div>
            </div>
            <div class="form-group col-md-12">
                <input class="btn btn-block" type="submit" value="Опубликовать" />
            </div>
        </div>
    }
}
<hr>

@helper PostHelper(Post post)
{
    <div class="row">
        @*Фото и ссылка на автора*@
        <div class="col-lg-2 text-center">
            <div>
                @if (post.Author.PhotoUrl != null)
                {
                    <img src="@Url.Content(post.Author.PhotoUrl)" width="100" />
                }
                else
                {
                    <img src="~/Content/Images/default-user.png" width="100" />
                }
            </div>
            <div>
                @Html.ActionLink(post.Author.Nickname, "Edit", "Profile", new { id = post.Author.Id }, null)
            </div>
        </div>

        @*Область поста*@
        <div class="col-lg-10">
            <div class="panel panel-primary col-md-12">
                <div class="panel-heading">@post.Theme</div>
                <div class="panel-body">
                    @post.Description
                    <hr>
                    @if (post.PhotoUrl != null)
                    {
                        <div>
                            <img src="@Url.Content(@post.PhotoUrl)" width="200" />
```

```
                        </div>
                        <hr> }
                    <div class="text-right">@post.Data.ToString("mm:hh dd.MM.yyyy")</div>
                </div>
            </div>
        </div>
    </div>
}
@foreach (var post in ViewBag.Posts)
{
    @PostHelper(post);
}
```

[^]

Изменение шаблона представлений

В шаблон представлений **_Layout.cshtml** внесем изменение для навигации по сайту, кнопку выхода и ссылку на профиль пользователя.

Пример измененного шаблона:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Клуб любителей мотоспорта</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Harley Blog", "Index", "Feed", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav pull-right">
```

```
                    @if (Session["UserNick"] != null)
                    {
                        <li>@Html.ActionLink(Session["UserNick"].ToString(), "Edit", "Profile", new { id = Session["UserId"] }, null)</li>
                    }
                    <li>@Html.ActionLink("Выйти", "Logout", "Welcome")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
    </div>
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @Scripts.Render("~/bundles/jqueryval")
    @RenderSection("scripts", required: false)
</body>
</html>
```

[^]

Источник — «http://wiki.sms-it.ru/index.php?title=Песочница/Практика2018/Методические_указания_по_выполнению_практического_задания&oldid=497501»