Админ панель: настройка доступа

Исходный код проекта

В настоящий момент кто угодно может перейти на URL вида /admin/games и начать отправлять заказы по назначению и редактировать каталог товаров. Для решения этой проблемы мы настроим авторизацию, чтобы среда ASP.NET Framework разрешала доступ к административным страницам только аутентифицированным пользователям.

Защита административных страниц

Страницы в приложении Web Forms защищаются посредством записей в файле Web.config. В примере ниже показаны дополнения, внесенные для защиты веб-форм в папке \Pages\Admin:

```
<?xml version="1.0"?>
<configuration>
  ...
  <location path="admin">
    <system.web>
      <authorization>
        <deny users="?"/>
      </authorization>
    </system.web>
  </location>
  <system.web>
    <authentication mode="Forms">
      <forms loginUrl="~/Pages/Login.aspx">
      </forms>
    </authentication>
    ...
  </system.web>
</configuration>
```

Элемент location указывает политику авторизации, а его атрибут path сообщает ASP.NET Framework, что мы хотим защитить страницы, доступ к которым осуществляется через URL, начинающийся с admin — это соответствует нашей схеме маршрутизации URL и покрывает все страницы, которые нуждаются в защите. Элементы, содержащиеся внутри location, применяются для доступа к страницам, указанным в атрибуте path. Эти элементы разрешают доступ к административным страницам со стороны любых аутентифицированных пользователей.

Аутентификация не интегрирована в систему конфигурирования маршрутизации, а это означает необходимость обновления файла web.config при изменении конфигурации маршрутизации, чтобы гарантировать корректность применения политики авторизации.

Поскольку мы собираемся предоставлять авторизацию только аутентифицированным пользователям, необходимо настроить политику аутентификации, для чего предназначен **атрибут authentication**. Для аутентификации пользователей существует множество разных способов. На выбор конкретного способа влияет платформа, используемая для хостинга приложения, или открытая система аутентификации, которую необходимо задействовать (такая как учетные записи Vkontakte или Google).

Аутентификация — это процесс идентификации пользователей, обычно с помощью имен пользователей и паролей. После выяснения, кто эти пользователи, применяется авторизация, которая определяет, какими средствами приложения разрешено им пользоваться. (Более подробно системы аутентификации, авторизации, ролей и профилей ASP.NET рассматриваются в разделе "Безопасность в ASP.NET".)

Мы будем применять аутентификацию с помощью форм, которая очень проста и содержится внутри самой платформы ASP.NET Framework. Здесь мы опустим ряд важных деталей, поскольку хотим сосредоточить внимание на процессе авторизации, а не аутентификации.

Аутентификация с помощью форм выбирается установкой атрибута mode элемента authentication в Forms. Элемент forms сообщает ASP.NET Framework, что пользователи должны перенаправляться на страницу ~/Pages/Login.aspx, которая создается в следующем разделе.

Создание веб-формы для входа

Мы должны предоставить пользователям возможность выполнения аутентификации, которая в приложении GameStore будет осуществляться с помощью имени пользователя и пароля. Мы создали новую веб-форму по имени Login.aspx в папке Pages. Для этой веб-формы был выбран шаблон Web Form Using Master Page и указана мастер-страница \Pages\Admin\Admin.Master. Контент веб-формы Login.aspx показан ниже:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits
    MasterPageFile="~/Pages/Admin/Admin.Master" %>

<asp:Content ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" DisplayMode="Si

    <div class="loginContainer">
        <div>
            <label for="name">Имя:</label>
            <input name="name" />
        </div>
        <div>
            <label for="password">Пароль:</label>
            <input type="password" name="password" />
        </div>
        <button type="submit">Войти</button>
    </div>
</asp:Content>
```

Обратите внимание что веб-форма, используемая в качестве страницы для входа, создана за пределами защищаемой папки. Это позволяет намного упростить конфигурирование и управление процессом аутентификации, чем попытка защиты в папке всех файлов за исключением одного.

Созданная веб-форма очень проста и содержит элементы input для ввода имени

и пароля, а также кнопку "Войти" для отправки формы серверу. Мы добавили элемент управления ValidationSummary, так что можем легко отображать сообщения об ошибках входа с применением средства привязки модели. В примере ниже приведено содержимое файла отделенного кода \Pages\Login.aspx.cs, в котором обрабатывается отправка формы:

```
using System;
using System.Web.Security;

namespace GameStore.Pages
{
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

            if (IsPostBack)
            {
                string name = Request.Form["name"];
                string password = Request.Form["password"];
                if (name != null && password != null
                        && FormsAuthentication.Authenticate(name, password))
                {
                    FormsAuthentication.SetAuthCookie(name, false);
                    Response.Redirect(Request["ReturnUrl"] ?? "/");
                }
                else
                {
                    ModelState.AddModelError("fail", "Логин или пароль не прави
                        "Пожалуйста введите данные заново");
                }
            }
        }
    }
}
```

Чтобы извлечь значения, введенные пользователем для имени и пароля, применяется коллекция Request.Form, а с помощью метода FormsAuthentication.Authenticate() проверяется их корректность.

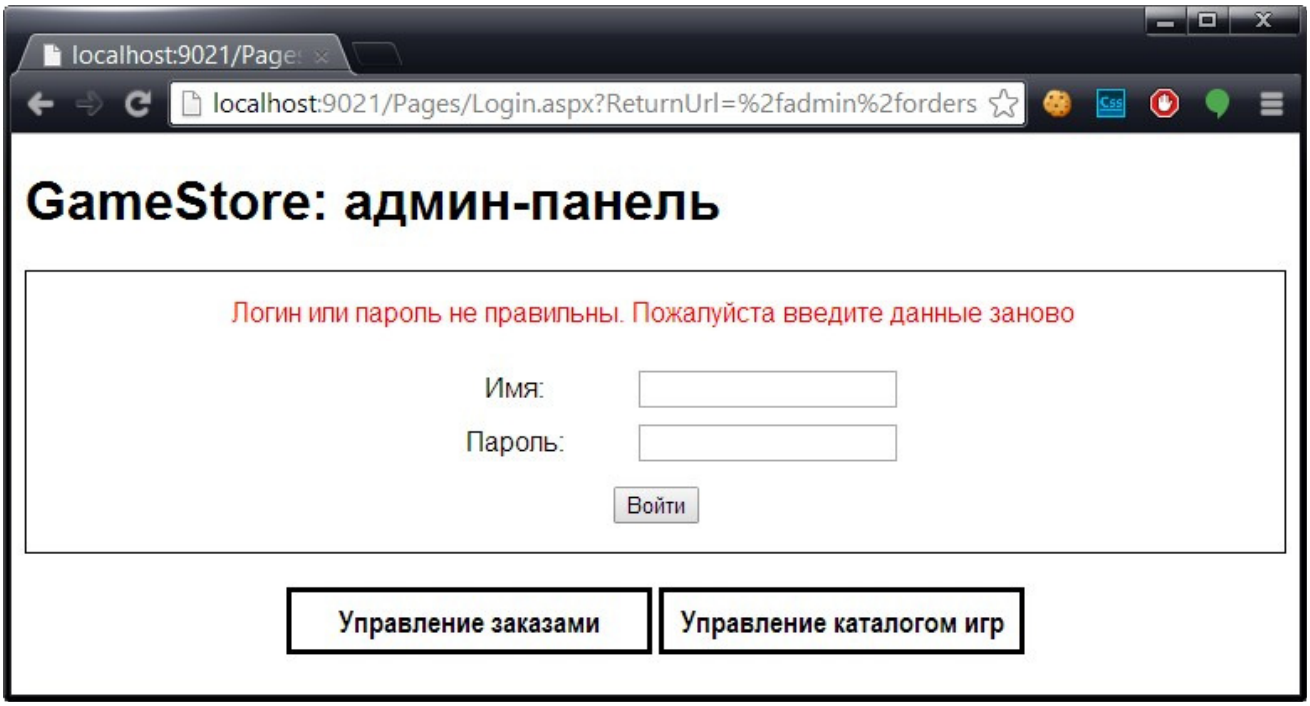
Если вы воспроизведете этот пример в Visual Studio, то получите предупреждение о том, что метод FormsAuthentication.Authenticate() был объявлен устаревшим. Не переживайте по этому поводу — нам всего лишь нужна простая система аутентификации. Более сложная система аутентификации - Membership API, описана в разделе "Безопасность в ASP.NET".

Если метод Authenticate() возвращает true, мы вызываем метод SetAuthCookie() для добавления к ответу cookie-набора, который позволит пользователю отправлять последующие запросы без необходимости в повторной аутентификации. (Методу SetAuthCookie() передается во втором аргументе значение false, которое означает, что аутентификация действительна только на протяжении сеанса пользователя — это особенно полезно во время тестирования, т.к. означает возможность делать любые аутентифицированные сеансы недействительными путем перезапуска приложения.) После создания cookie-набора мы вызываем метод Response.Redirect() для перемещения пользователя на URL, предоставляемый свойством ReturnUrl. Это устанавливается средой ASP.NET Framework, когда требуется аутентификация — вскоре будет показано, как все работает.

Если метод Authenticate() возвращает false, это означает, что пользователь не предоставил корректные учетные данные. Мы обрабатываем такую ситуацию вызовом (метода ModelSate.AddModelError(), добавляющего к ответу сообщение, которое будет отображаться в элементе управления ValidationSummary, определенном внутри веб-формы.

Тестирование случая, когда аутентификация не прошла

Мы не определили никаких пользовательских учетных данных для проверки средой ASP.NET Framework, поэтому любые попытки аутентификации потерпят неудачу. Чтобы увидеть конфигурацию аутентификации в работе, запустите приложение. В случае перехода на URL вида /admin/orders или admin/games произойдет перенаправление на веб-форму Login.aspx, которая предлагает ввести имя пользователя и пароль. Щелчок на кнопке "Войти" приведет к отправке формы и отображению сообщения об ошибке, как показано на рисунке ниже:



Если вы посмотрите на URL, на который был перенаправлен браузер, то увидите что изначально запрошенный URL включен в строку запроса. Вы также заметите, что этот URL не согласуется со схемой маршрутизации. Встроенная в ASP.NET Framework поддержка аутентификации и авторизации не особенно хорошо сочетается со средством маршрутизации, которое появилось относительно недавно.

Тестирование случая, когда аутентификация и авторизация прошли успешно

Мы собираемся определить учетные данные в файле Web.config. Подобный подход никогда не будет применяться в реальном проекте, т.к. он плохо масштабируется и не позволяет пользователям изменять свои учетные данные. Однако в примере приложения GameStore нам необходимы учетные данные только для одного пользователя, чтобы продемонстрировать средства авторизации в работе, и установка их в файле Web.config представляется наиболее простой. В примере ниже показаны дополнения, внесенные в элемент authentication:

```
<system.web>
    <authentication mode="Forms">
      <forms loginUrl="~/Pages/Login.aspx">
        <credentials passwordFormat="Clear">
          <user name="admin" password="123456"/>
        </credentials>
      </forms>
    </authentication>
</system.web>
```

Мы создали пользователя с именем admin и паролем 123456. Чтобы протестировать результат успешной аутентификации и авторизации, необходимо запустить приложение и перейти на URL вида /admin/orders. Снова будут запрошены учетные данные, и если ввести их корректно, произойдет перенаправление на запрошенную страницу.