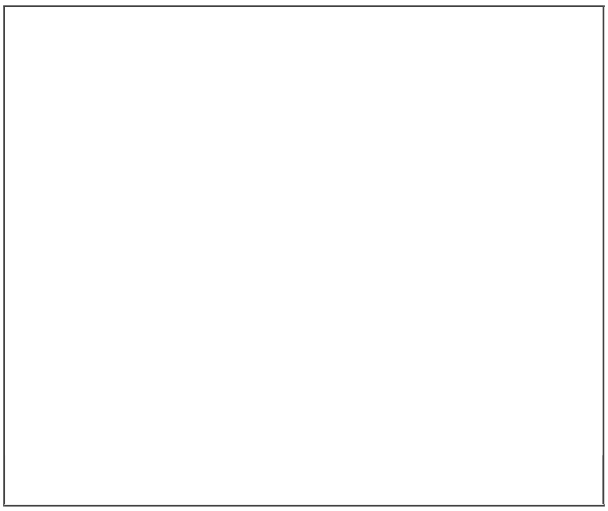
Список категорий



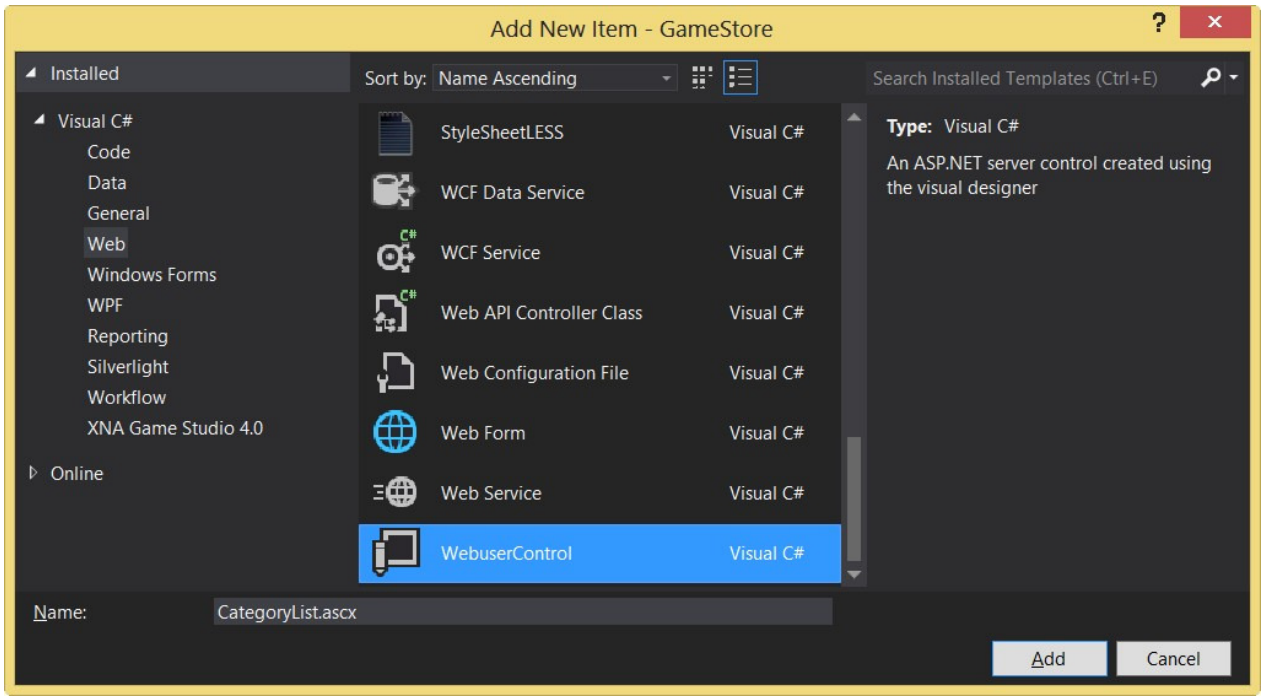
Исходный код проекта

В этой статье мы собираемся создать элемент управления Web Forms, который будет отображать информацию о категориях. Элементы управления обсуждаются более подробно статье "Пользовательские элементы управления". *Элементы управления* — это многократно используемые блоки функциональности, которые генерируют HTML-разметку как часть ответа для браузера. Мы создадим пользовательский элемент управления, в котором применяется такой же подход с разметкой, фрагментами кода и отделенным кодом, как в веб-формах и мастер-страницах.

Мало того, что пользовательские элементы управления являются многократно используемыми, они также помогают структурировать приложение Web Forms, предоставляя механизм, который позволяет разбить функциональность на автономные единицы. Это делает применение и тестирование изменений значительно проще, чем в случае включения всей разметки и кода в единственную веб-форму или мастер-страницу.

Создание пользовательского элемента управления

Щелкните правой кнопкой мыши на папке Controls в окне Solution Explorer и выберите в контекстном меню пункт Add New Item. Затем выберите шаблон Web User Control (Пользовательский веб-элемент управления), как показано на рисунке ниже, укажите в качестве имени CategoryList.ascx и щелкните на кнопке Add (Добавить). Среда Visual Studio добавит в окно Solution Explorer элемент CategoryList.ascx.cs, который можно развернуть, чтобы увидеть файл отделенного кода CategoryList.ascx.cs и файл CategoryList.ascx.designer.cs (последним мы пользоваться не будем, т.к. не являемся сторонниками визуального конструирования веб-приложений).



Расширение `.ascx` обозначает пользовательский элемент управления таким же образом, как расширение `.aspx` указывает на веб-форму. Расширения файлов важны в приложениях `Web Forms`.

Среда `Visual Studio` откроет файл `CategoryList.ascx` для редактирования автоматически после создания файлов элемента управления. Приведите содержимое файла `CategoryList.ascx` в соответствие со следующим кодом:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="CategoryList.ascx
   Inherits="GameStore.Controls.CategoryList" %>

<%= CreateHomeLinkHtml() %>

<% foreach (string category in GetCategories()) {
       Response.Write(CreateLinkHtml(category));
}%>
```

Как видите, файлы пользовательского элемента управления очень похожи на веб-формы за исключением того, что объявление в начале файла содержит директиву `Control`, которая сообщает `ASP.NET Framework` о том, что это элемент управления, а не полная веб-форма.

В файле `CategoryList.ascx` используются два фрагмента кода. Первый из них вызывает метод `CreateHomeLinkHtml()` класса отделенного кода для генерации ссылки, которая отобразит все товары независимо от их категорий — эта ссылка помечена как "Главная".

Второй фрагмент кода вызывает метод `GetCategories()` класса отделенного кода, чтобы получить перечисление доступных категорий товаров. В цикле `foreach` вызывался метод `CreateLinkHtml()` (также класса отделенного кода) для генерации `HTML`-разметки, представляющей каждую категорию.

В примере ниже показан код, добавленный в файл отделенного кода `CategoryList.ascx.cs` для определения методов, которые вызываются внутри фрагментов кода:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Routing;
using GameStore.Models.Repository;

namespace GameStore.Controls
{
    public partial class CategoryList : System.Web.UI.UserControl
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected IEnumerable<string> GetCategories()
        {
            return new Repository().Games
                .Select(p => p.Category)
                .Distinct()
                .OrderBy(x => x);
        }

        protected string CreateHomeLinkHtml()
        {
            string path = RouteTable.Routes.GetVirtualPath(null, null).Virtual
            return string.Format("<a href='{0}'>Главная</a>", path);
        }

        protected string CreateLinkHtml(string category)
        {
            string path = RouteTable.Routes.GetVirtualPath(null, null,
                new RouteValueDictionary() { { "category", category },
                    {"page", "1"} }).VirtualPath;

            return string.Format("<a href='{0}'>{1}</a>",
                path, category);
        }
    }
}
```

Класс отделенного кода для пользовательского элемента управления является

производным от класса System.Web.UI.UserControl. За исключением базового класса класс отделенного кода для пользовательского элемента управления очень похож на свой аналог для веб-формы. В классе были определены три метода, которые ранее вызывались внутри фрагментов кода. Метод GetCategories() применяет LINQ для генерации списка названий категорий, который отсортирован в алфавитном порядке и не содержит дубликатов. Метод CreateLinkHtml() использует систему маршрутизации для генерирования URL, которые содержат компонент категории, позволяя представлять пользователю неотфильтрованный список игр. Мы генерируем все элементы ссылок на основе конфигурации маршрутизации с помощью того же самого подхода, который применяется при создании страничных URL. Вскоре мы просмотрим и скорректируем URL.

Применение пользовательского элемента управления в мастер-странице

Мы собираемся применить пользовательский элемент управления в файле Store.Master, чтобы он был доступным любой веб-форме, использующей данную мастер-страницу. В примере ниже показаны изменения, внесенные в файл \Pages \Store.Master:

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Store.master.cs" I
<%@ Register TagPrefix="GS" TagName="CategoryLinks" Src="~/Controls/CategoryLi

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>GameStore</title>
    <link rel="stylesheet" href="/Content/Styles.css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <div id="header">
                <div class="title">GameStore - магазин компьютерных игр</div
            </div>
            <div id="categories">
                <GS:CategoryLinks runat="server" />
            </div>
            <div>
                <asp:ContentPlaceHolder ID="bodyContent" runat="server" />
            </div>
        </div>
    </form>
</body>
</html>
```

Для применения пользовательского элемента управления требуются два добавления. Для начала необходимо добавить директиву `Register`, которая сообщает `ASP.NET Framework`, какой файл содержит элемент управления и как ссылаться на этот элемент управления в разметке. Мы указали файл `\Controls \CategoryList.ascx` с использованием атрибута `Src`, а с помощью атрибутов `TagPrefix` и `TagName` отразили, что собираемся ссылаться на элемент управления посредством комбинированного дескриптора `GS:CategoryLinks`.

Когда среда `ASP.NET Framework` обнаружит элемент с таким дескриптором на мастер-странице, она обработает пользовательский элемент управления и добавит сгенерированную `HTML`-разметку в ответ, отправляемый браузеру. Как видно в примере, мы добавили единственный элемент, который ссылается на пользовательский элемент управления:

```
<GS:CategoryLinks runat="server" />
```

Обратите внимание, что для применения пользовательского элемента управления атрибут runat установлен в server — если этого не сделать, среда ASP.NET Framework проигнорирует элемент, и не будет обрабатывать элемент управления.

Добавление стилей CSS

Нам необходимо добавить ряд стилей в файл \Content\Styles.css для управления внешним видом генерируемых ссылок на категории. Новые стили показаны ниже:

```
div#categories a
{
    font: bold 1.1em "Arial Narrow","Franklin Gothic Medium",Arial; display: b
    text-decoration: none; padding: .6em; color: Black;
    border-bottom: 1px solid silver;
}
div#categories a.selected { background-color: #666; color: White; }
div#categories a:hover { background-color: #CCC; }
div#categories a.selected:hover { background-color: #666; }
```

После модификации CSS-файлов приложения часто требуется перезагружать веб-страницу в браузере. Браузер кеширует CSS-файл и перезапуск приложения не приводит к запросу браузером новой версии этого файла.

Расширение схемы URL

Запустив приложение, вы увидите, что пользовательский элемент управления обработан для создания ссылок на категории. Щелчок на этих ссылках пока что не дает никакого эффекта, поскольку соответствующая функциональность еще не реализована.

В этом разделе нас интересуют URL, которые были сгенерированы для ссылок. Если навести курсор мыши на одну из ссылок на категорию, можно увидеть URL в следующей форме:

```
http://localhost:9021/1?category=RPG
```

При создании ссылок в классе отделенного кода для пользовательского

элемента управления мы указывали переменную category, и среда ASP.NET Framework по умолчанию будет применять для выражения этой информации строки запросов. Необходимо генерировать URL, которые согласуются с новой схемой маршрутизации. Это означает что понадобится модифицировать конфигурацию маршрутизации, добавив поддержку для категории.

В примере ниже показано изменение, внесенное в файл \App_Start \RouteConfig.cs:

```
using System;
using System.Web.Routing;

namespace GameStore
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.MapPageRoute(null, "list/{category}/{page}",
                                      "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "list/{page}", "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "", "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "list", "~/Pages/Listing.aspx");
        }
    }
}
```

Мы добавили в конфигурацию маршрутизации поддержку нового типа URL, сделав допустимыми такие URL:

```
http://localhost:9021/list/Шутер/2
```

В этом URL указаны категория и страница, которые будут интерпретироваться как запрос для отображения заданной страницы товаров определенной категории. При определении поддержки таких URL мы по-прежнему применяем базовые средства маршрутизации.

Снова запустив приложение и наведя курсор мыши на одну из ссылок на категории, можно заметить, что формат URL изменился нужным образом.

Здесь необходимо отметить два момента. Во-первых, природа и структура созданных ссылок происходит целиком из конфигурации маршрутизации; это означает, что реализованный пользовательский элемент управления не имеет жестко закодированной информации, которую пришлось бы изменять в случае изменения схемы URL. Во-вторых, пользовательскому элементу управления ничего

не известно о веб-формах, указываемых URL категорий, т.е. если нужно нацелить URL категорий на другие веб-формы, то придется изменить только конфигурацию маршрутизации. Средство маршрутизации не только позволяет создавать удобные схемы URL, но также привносит в приложение гибкость и упрощает задачу применения изменений в будущем.

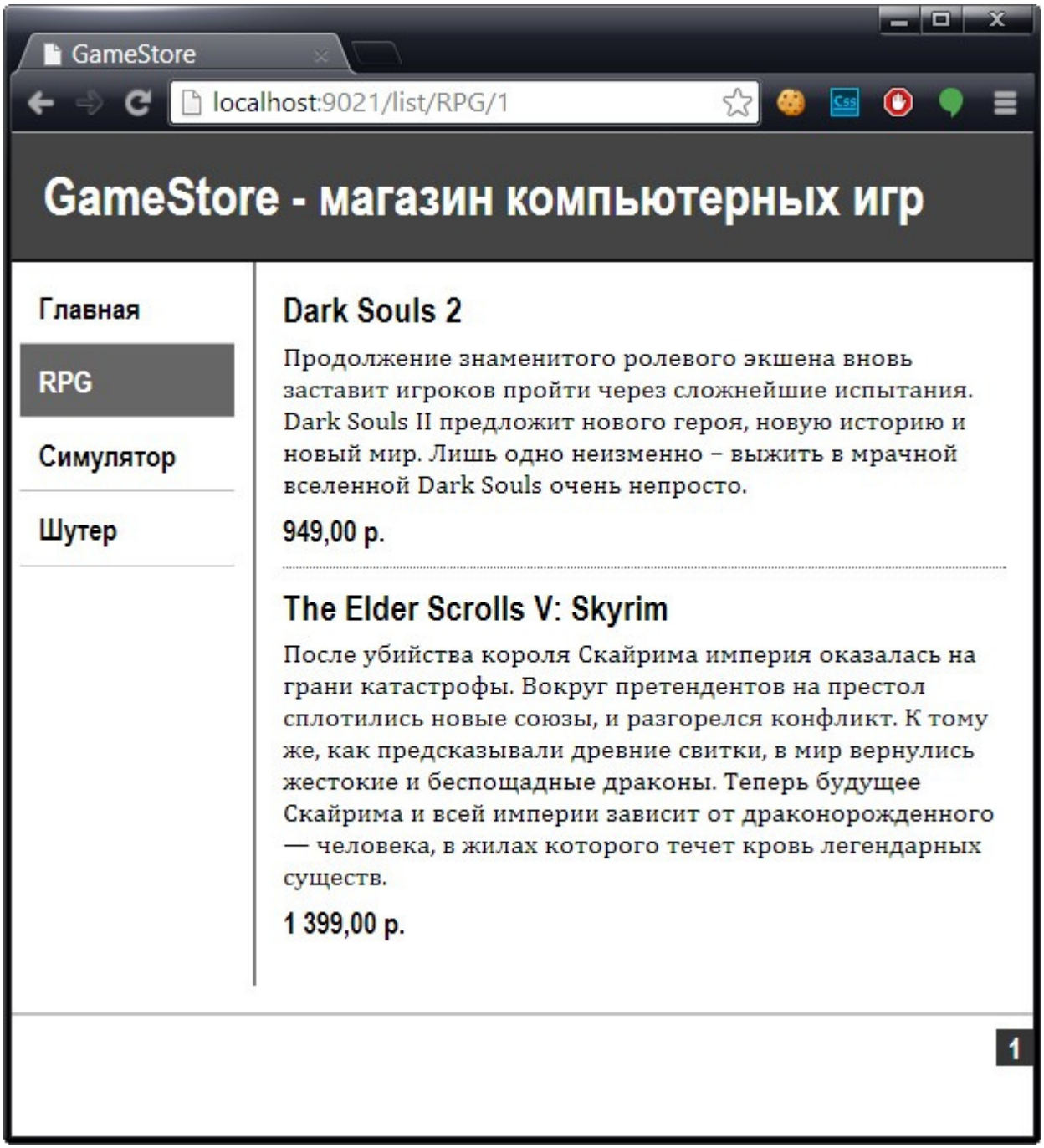
Добавление поддержки для отображения категорий

Мы планируем расширить существующую функциональность в файле \Pages \Listing.aspx, добавив поддержку фильтрации товаров по категориям. В примере ниже показаны изменения, внесенные в файл отделенного кода Listing.aspx.cs:

```
using System;
using System.Collections.Generic;
using GameStore.Models;
using GameStore.Models.Repository;
using System.Linq;

namespace GameStore.Pages
{
    public partial class Listing : System.Web.UI.Page
    {
        private Repository repository = new Repository();
        private int pageSize = 4;

        protected int CurrentPage
        {
            get {
                int page;
                page = GetPageFromRequest();
                return page > MaxPage ? MaxPage : page;
            }
        }

        protected int MaxPage
        {
            get {
                int prodCount = FilterGames().Count();
                return (int)Math.Ceiling((decimal)prodCount / pageSize);
            }
        }

        private int GetPageFromRequest()
        {
            int page;
            string reqValue = (string)RouteData.Values["page"] ??
                Request.QueryString["page"];
            return reqValue != null && int.TryParse(reqValue, out page) ? page
        }

        protected IEnumerable<Game> GetGames()
        {
            return FilterGames()
                .OrderBy(g => g.GameId)
```

Мы добавили поддержку для чтения выбранной категории из значений маршрутизации либо из значений строки запроса, а также для фильтрации набора игр, которые отображаются пользователю. Для фильтрации объектов товаров, когда категория указана (и для выяснения количества страниц товаров, которые должны быть отображены), применяется LINQ.

Эти изменения обеспечивают поддержку, необходимую для того, чтобы работали ссылки на категории. Чтобы протестировать их, нужно просто запустить приложение и щелкнуть на одной из ссылок. На рисунке представлен результат щелчка на ссылке RPG с целью отображения товаров только этой категории.



Веб форма Listing.aspx не должна знать или заботиться о том, как генерируются ссылки на категории – она просто ищет детали категории в получаемом запросе. Не существует прямого отношения между пользовательским элементом управления, создающим ссылки, и веб-формой, которая их использует – все делается опосредованно через конфигурацию маршрутизации. Это значит,

что пользовательский элемент управления и веб-форму можно изменять независимо друг от друга или создавать источники навигационных ссылок на категории.

Подсвечивание текущей категории

Нам необходимо внести одно последнее изменение в навигационные ссылки на категории, связанное с подсветкой текущей выбранной категории — это не только визуально подкрепит факт щелчка на одной из ссылок, но также и укажет пользователю, что отображается лишь подмножество товаров как показано на рисунке выше. В примере ниже приведено измененное содержимое файла отделенного кода `\Controls\CategoryList.ascx.cs`:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Routing;
using GameStore.Models.Repository;

namespace GameStore.Controls
{
    public partial class CategoryList : System.Web.UI.UserControl
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected IEnumerable<string> GetCategories()
        {
            return new Repository().Games
                .Select(p => p.Category)
                .Distinct()
                .OrderBy(x => x);
        }

        protected string CreateHomeLinkHtml()
        {
            string path = RouteTable.Routes.GetVirtualPath(null, null).Virtual
            return string.Format("<a href='{0}'>Главная</a>", path);
        }

        protected string CreateLinkHtml(string category)
        {

            string selectedCategory = (string)Page.RouteData.Values["category"
                ?? Request.QueryString["category"];

            string path = RouteTable.Routes.GetVirtualPath(null, null,
                new RouteValueDictionary() { { "category", category },
                    {"page", "1"} }).VirtualPath;

            return string.Format("<a href='{0}' {1}>{2}</a>",
                path, category == selectedCategory ? "class='selected'" : "",
        }
```

Пользовательский элемент управления получает доступ к деталям обрабатываемого запроса через свойство `Request`, но для извлечения значений переменных маршрутизации необходимо применять свойство `Page.RouteData`. Свойство `Page` предоставляет доступ к подробным сведениям о веб-форме, в которой используется элемент управления. Мы добавили класс `selected` к элементу ссылки, созданному для текущей выбранной категории.