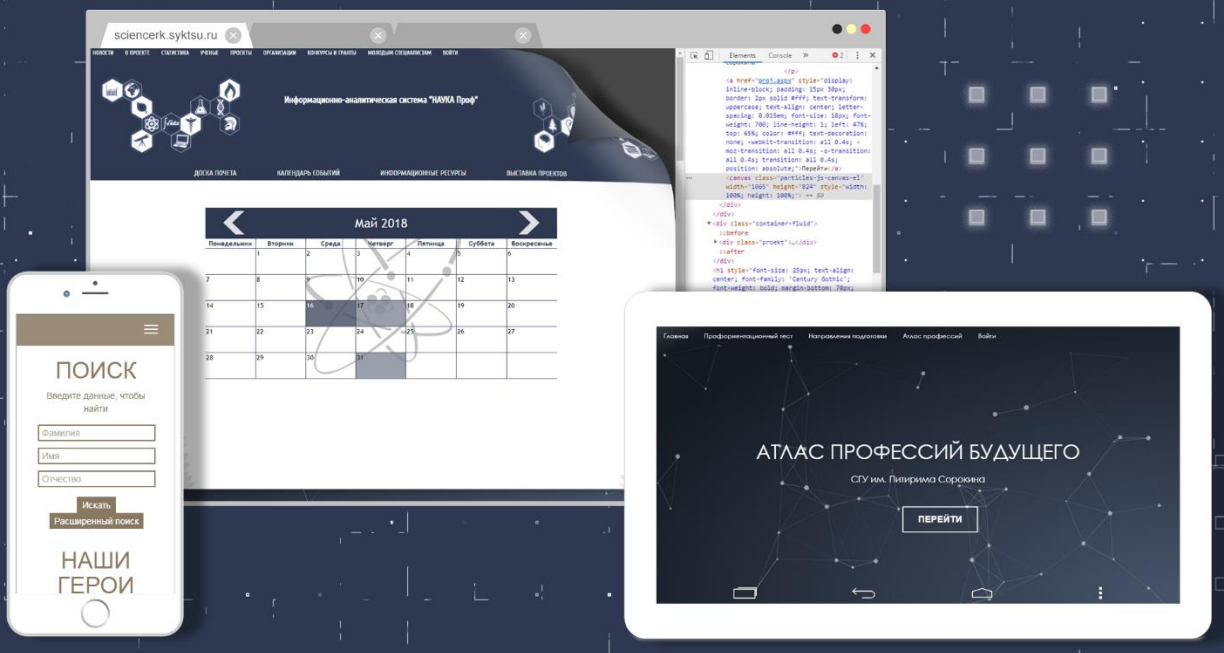
С. Т. ГУЛЯЕВА, В. В. МИРОНОВ, Н. О. КОТЕЛИНА

ASP.NET Web Forms

в задачах и примерах



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сыктывкарский государственный университет имени Питирима Сорокина»
(ФГБОУ ВО «СГУ им. Питирима Сорокина»)
Институт точных наук и информационных технологий

С. Т. Гуляева, В. В. Миронов, Н. О. Котелина

ASP.NET Web Forms
в задачах и примерах

Учебное пособие

Текстовое учебное электронное издание на компакт-диске

Сыктывкар
Издательство СГУ им. Питирима Сорокина
2020

ISBN 978-5-87661-625-8

Титул Об издании Производственно-технические сведения Оглавление

УДК 004.42; 004.415
ББК 32.973.2
Г 94

*Издается по постановлению научно-методического совета
ФГБОУ ВО «СГУ им. Питирима Сорокина»*

Рецензент: _____
_____

Гуляева, С. Т.

Г 94 ASP.NET Web Forms в задачах и примерах [Электронный ресурс] : учебное пособие : текстовое учебное электронное издание на компакт-диске / С. Т. Гуляева, В. В. Миронов, Н. О. Котелина; Федер. гос. бюдж. образоват. учреждение высш. образования «Сыктыв. гос. ун-т им. Питирима Сорокина». – Электрон. текстовые дан. (7,08 Мб). – Сыктывкар: Изд-во СГУ им. Питирима Сорокина, 2020. – 1 опт. компакт-диск (CD-ROM). – Систем. требования: ПК не ниже класса Pentium III; 256 Мб RAM; не менее 1,5 Гб на винчестере; Windows XP с пакетом обновления 2 (SP2); Microsoft Office 2003 и выше; видеокарта с памятью не менее 32 Мб; экран с разрешением не менее 1024 × 768 точек; 4-скоростной дисковод (CD-ROM) и выше; мышь. – Загл. с титул. экрана. – ISBN 978-5-87661-625-8.

Учебное пособие содержит теоретические аспекты и примеры задач по технологии программирования ASP.NET на примере программной модели Web Forms. Основой для написания ресурса послужил лекционный материал авторов на основе многолетнего практического опыта в области IT-проектов, также задействованы примеры из реальных проектов, выполненных с применением технологии ASP.NET студентами кафедры информационных систем института точных наук и информационных технологий СГУ им. Питирима Сорокина под руководством авторов.

Пособие предназначено для студентов высших учебных заведений, обучающихся по направлению 09.03.03 «Прикладная информатика», для освоения дисциплин «Программирование в интернет-среде с использованием технологии ASP.NET», «Web-программирование». Издание может быть полезно магистрантам (09.04.03 «Прикладная информатика»), аспирантам и преподавателям.

УДК 004.42; 004.415
ББК 32.973.2

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодняшний день цифровизация занимает неотъемлемую часть жизни – ПК, мобильные устройства, Интернет, социальные сети, SMART-технологии, Интернет вещей, виртуальная реальность и многое другое, – все это является тем, без чего не обходится ни один человек. В сфере ИТ есть небольшая градация языков и технологий программирования в зависимости от области их применения (направления ранжированы с учетом трендов 2019 года и начала 2020 года), а именно:

1. **Облачные вычисления** (в приоритете — AWS, применяемый многими компаниями по всему миру за удобство и надежность) [14, 15].
2. **AI и machine learning**. В создании ИС, основанных на machine learning, есть два ответвления: это нейронные сети (deep learning) и статистические методы (data mining), например, градиентный бустинг. Языки программирования: в основном C++, Python, Java; реже C#, R, Haskell, Go, Swift. Дополнительно используют библиотеки TensorFlow, Theano или Torch [14, 15].
3. **Разработка мобильных приложений** – Java, Kotlin (Android), Swift (Apple), C#, Python, JavaScript [9, 14, 15].
4. **Web-программирование** – JavaScript (задействуется в любом случае), PHP, Java, C#, VBA, SQL, Python, Ruby [9].
5. **Разработка игр** – C++, C#, Lua и JavaScript.
6. **Big Data, Data Science** (R, Python).
7. **Разработка desktop-приложений** – C++, C#, Objective-C, Swift.
8. **Программирование встроенных систем (бытовая техника)** – C, C++.
9. **Интернет вещей** – C, C++.
10. **Автоматизация бизнеса**:
    - популярные в России – «Мегаплан», amoCRM, «Битрикс24», 1С;
    - лидеры международного рынка – SAP, Salesforce, Microsoft Dynamics CRM, Siebel Oracle CRM и другие [9].

В последнее время широкую популярность приобрело такое направление, как web-программирование. Параллельно меняется и топ-10 распространенных языков программирования (далее – ЯП), – чаще всего это либо универсальные ЯП, либо ЯП, ориентированные на web-решения.

Очень часто при разработке очередного web-сайта у начинающих программистов возникает ряд вопросов:

— Какую технологию программирования лучше использовать?
— На каком ЯП писать?
— Стоит ли использовать CMS?
— Что лучше – ASP.NET или PHP?

Ответ простой. Инструментарий всегда следует выбирать исходя из поставленных целей и задач, не иначе. То же касается и вопроса использования конкретных CMS (*Рисунок 1*).

Рисунок 1. Популярные категории сайтов [8]

Иными словами, если необходимо разработать сайт-визитку, лендинг, сайт-витрину или стандартный портфолио, то можно вполне использовать CMS (чаще всего они на основе PHP) или технологию, поддерживающую быструю разработку приложений (такой вариант используется реже). Если же необходимо разработать полноценный портал или крупное web-приложение, то следует учитывать как возможности самой технологии, так и гибкость ЯП [17].

Очень часто возникают споры по поводу вопроса «Что лучше – ASP.NET или PHP?», поскольку большинство сайтов написано на PHP (некоторые даже пишутся с использованием конкретных CMS). Однако на втором месте после PHP по популярности располагается технология ASP.NET с поддержкой сразу нескольких ЯП. Стоит отметить, что некорректно сравнивать ЯП с технологией программирования, это разные понятия. Поэтому правильнее будет спросить «Что лучше – фреймворки и CMS на основе PHP или ASP.NET?». Ответ всегда один – цель должна оправдывать средства (*Таблица 1*).

Таблица 1. Сравнительный анализ ASP.NET и CMS на основе PHP

Критерий сравнения	**Фреймворки и CMS на основе PHP**	**Технология ASP.NET**
Транслятор	Интерпретатор	Компилятор
Масштаб проектов	Малые и средние проекты	Средние и крупные проекты
Распространенность [17], в процентах	79 (1 место)	11,2 (2 место)
Производительность	Средняя	Высокая с большой нагрузкой и ниже среднего (незначительно) с небольшой нагрузкой
Кроссплатформенность	имеется	имеется
ЯП	PHP	Visual C#, Delphi.NET, J#, VBA.NET и другие
Традиционные СУБД	MySQL, PostgreSQL	MS SQL Server
Сервер	Apache	IIS

По изучению ЯП PHP и конкретных CMS на основе PHP существует достаточное количество обучающих ресурсов и книг, чего нельзя сказать о ASP.NET. Начинающий разработчик ASP.NET, как правило, руководствуется открытой документацией производителя, изучает обзорные статьи и видеоролики зарубежных блоггеров по данной тематике, обсуждает отдельные нюансы на форумах, однако подобрать стоящую литературу довольно непросто.

ASP.NET поддерживает несколько программных моделей для создания web-приложений:

— **ASP.NET Web Forms** — фреймворк для создания модульных веб-страниц из компонентов с обработкой событий пользовательского интерфейса на стороне сервера [2].

— **ASP.NET MVC** — фреймворк для создания веб-страниц с использованием шаблона проектирования MVC [2].
— **ASP.NET Core** — свободно-распространяемый кроссплатформенный фреймворк для создания веб-приложений с открытым исходным кодом. Имеет модульную структуру и совместима с такими операционными системами, как Windows, Linux и macOS.
— **ASP.NET Web Pages** — упрощённый синтаксис для добавления динамического кода и доступа к данным внутри HTML разметки веб-страниц [2].
— **ASP.NET Web API** — фреймворк для создания Web API поверх .NET Framework.
— **ASP.NET WebHooks** — реализация шаблона Webhook для подписки на события и публикации событий через HTTP [2].
— **SignalR** — фреймворк для двунаправленного обмена сообщениями в реальном времени между клиентом и сервером [2].

Другие расширения ASP.NET:

— **ASP.NET Handler** — компоненты, реализующие интерфейс System.Web.IHttpHandler. В отличие от страниц ASP.NET у них нет файла с HTML-разметкой, нет поддержки обработки событий и других вспомогательных технологий. Они содержат только файл с программным кодом, написанным на любом из .NET-совместимых языков, который записывает какие-то данные в HTTP-ответ. HTTP-обработчики схожи с ISAPI-расширениями [2].
— **ASP.NET AJAX** — расширение, содержащее как клиентские, так и серверные компоненты для создания ASP.NET страниц, реализующих функциональность AJAX [2].
— **ASP.NET Dynamic Data** — скаффолдинговое расширение для создания управляемых данными приложений [2].

Таким образом, учебное пособие представляет собой руководство для работы с технологией ASP.NET с использованием программной модели Web Forms на примере реальных проектов:

— Автоматизированная информационно-аналитическая система «НАУКА Проф», *победитель VI Республиканского молодежного инновационного конвента в 2016 г.* (г. Ухта, Республика Коми), *победитель фонда президентских грантов в 2017 г.* (г. Москва), http://sciencerk.syktsu.ru.
— АИС «Книга Памяти Республики Коми», *республиканский проект, разработан при финансовой поддержке Администрации Главы Республики Коми*, http://memorybook-rk.ru.
— АИС «Атлас профессий будущего СГУ им. Питирима Сорокина», проект профориентационной направленности 2018 г., http://atlas-prof.syktsu.ru.
— Web-сайт для научного журнала «Человек. Культура. Образование» (соответствует всем необходимым требованиям) 2019 г., расположенный по адресу http://pce.syktsu.ru/.

ЧАСТЬ 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ТЕХНОЛОГИИ ASP.NET

Глава 1. Введение в технологию ASP.NET Web Forms

ASP.NET — это современная технология для разработки Web-сайтов и Web-сервисов, работающих под управлением IIS. Стоит отметить, что данная платформа отличается от сторонних решений высокой степенью интеграции с серверными продуктами (прослеживается по специфике самой технологии), а также отвечает требованиям безопасности [10]. Более того, используемые языки программирования, с помощью которых осуществляется разработка Web-приложений, соответствуют объектно-ориентированному подходу к программированию, что делает разработку, модификацию и отладку исполнимой части более простым занятием. В данном пособии рассматривается классический вариант технологии ASP.NET Web Forms.

Важным моментом в понимании архитектуры ASP.NET является тот факт, что она является частью инфраструктуры .NET Framework (*Рисунок 2*).

Рисунок 2. Архитектура .NET Framework [11]

К преимуществам технологии ASP.NET относят следующее:

- Интеграция с .NET Framework.
- Программный код компилируется, а не интерпретируется (первый запрос к странице запустит процесс компиляции, что облегчит работу при последующих запросах).
- Обслуживается CLR.
- Объектно-ориентированная технология (свободное использование классов и других атрибутов ООП[1]).
- Адаптивный рендеринг.
- Простота развертывания и конфигурации.

ASP.NET функционирует на серверах с операционной системой Windows от компании Microsoft и требует наличия соответствующего проприетарного набора серверов для служб

[1] При использовании классов в структуре сайта необходимо в программном коде страницы (файл *.cs) либо указать пространство имен нужного класса, либо использовать конструкцию using [12]. В случае использования варианта Web site файлы классов необходимо помещать в предопределенную папку ASP.NET App_Code [5].

Интернет (далее – IIS)[2]. В классическом понимании web-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиапотоком или другими данными. Следовательно, в основе функционирования технологии ASP.NET (как и в основе любой другой технологии создания web-приложений и web-сервисов) лежит клиент-серверная архитектура (*Рисунок 3*). Однако при этом важно понимать, что в действительности происходит, когда пользователь вводит какой-либо URL в адресную строку браузера (в понимании ASP.NET).

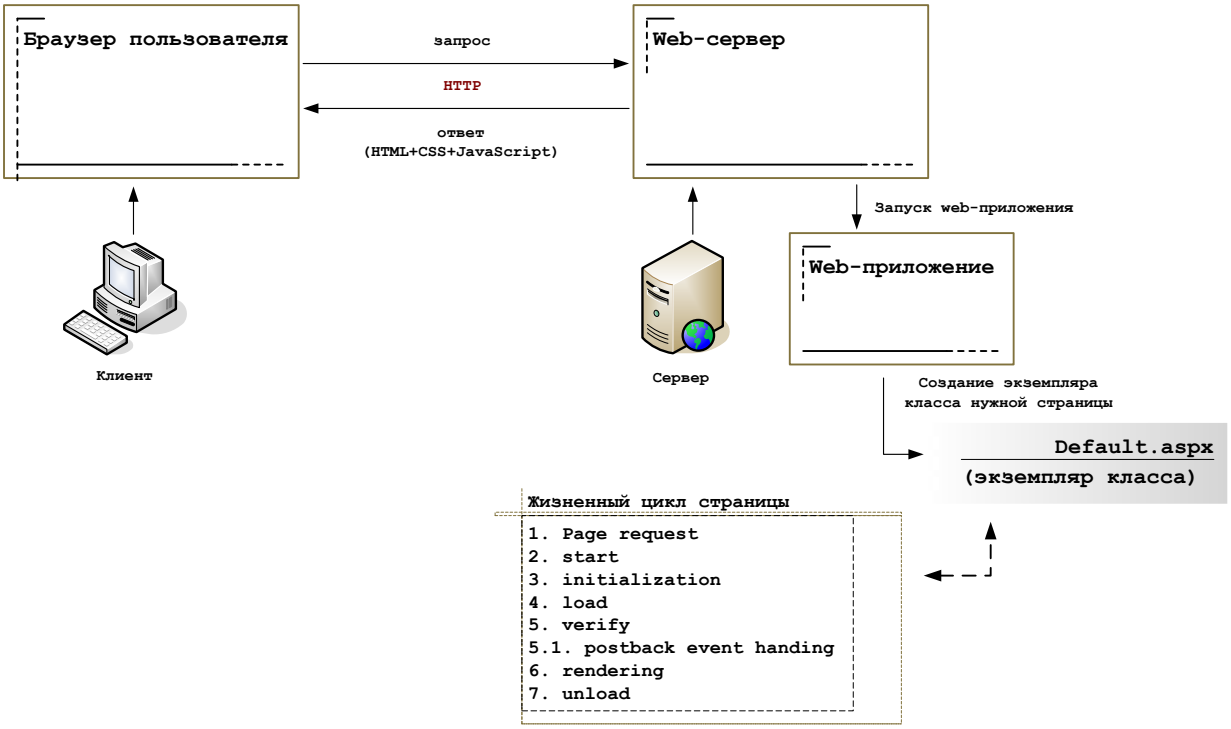


Рисунок 3. Клиент-серверная архитектура в понимании ASP.NET

Согласно базовому курсу по компьютерным сетям клиент-серверная архитектура имеет ряд преимуществ, что позволяет ей служить прочным фундаментом в процессе проектирования и разработки любой современной автоматизированной ИС. Также известно, что протокол HTTP – это протокол прикладного уровня для передачи данных от браузера клиента к серверу и обратно (сообщения обычно передаются через 80 порт или 443 при использовании Secure HTTP). Таким образом, на основе клиент-серверной архитектуры можно описать классический сценарий взаимодействия элементов web-приложения (сайта или сервиса) друг с другом и с браузером пользователя, осуществляющим запрос формы этого приложения.

При обращении пользователя к нужному web-ресурсу формируется запрос от клиента в адрес сервера (в случае технологии ASP.NET это сервер IIS), который запускает в работу соответствующее web-приложение. Запущенное приложение формирует ответ посредством создания на сервере экземпляра класса запрошенной Web-формы, после чего происходит генерация HTML-текста, который передается браузеру пользователя в качестве ответа на запрос. Сразу после этого экземпляр класса web-формы уничтожается (*Таблица 2*).

[2] Для разработки web-приложений, использующих сторонние сервера под управлением операционных систем группы Unix, используется технология ASP.NET Core [6, 13].

Таблица 2. Этапы жизненного цикла страницы [6]

Этап	Описание
Запрос страницы (page request)	Запрос страницы происходит перед началом жизненного цикла страницы. При запросе страницы пользователем ASP.NET определяет, нужно ли обрабатывать и компилировать страницу (до начала жизненного цикла страницы) или отправить в ответ на запрос кэшированную версию страницы, не запуская ее обработку
Запуск (start)	На начальном этапе устанавливаются свойства страницы, например, Request и Response. На этом этапе страница также определяет, является ли запрос обратной передачей или новым запросом, и устанавливает свойство IsPostBack. Кроме этого, на этом этапе устанавливается свойство страницы UICulture
Инициализация страницы (initialization)	Во время инициализации страницы элементы управления страницы являются доступными, устанавливаются все свойства элементов управления UniqueID. На странице также применяются темы. Если текущий запрос является обратным запросом, данные обратного запроса не загружены, а значения свойств элементов управления не восстановлены к значениям в состоянии просмотра
Загрузка (load)	Во время загрузки, если текущий запрос является обратным запросом, в свойства элементов управления будут переданы данные, восстановленные из состояния просмотра и состояния управления
Проверка (verify)	Во время проверки вызывается метод Validate всех проверяющих элементов управления, который устанавливает свойство IsValid отдельных проверяющих элементов управления и страницы
Обработка событий обратного запроса (postback event handing)	Если запрос является обратным, вызывается любой из обработчиков событий
Отрисовка (rendering)	Перед отрисовкой производится сохранение состояния просмотра страницы и всех элементов управления. На этапе отрисовки страница вызывает метод Render для каждого элемента управления, предоставляя модуль записи текста, который записывает полученные данные в OutputStream свойства страницы Response
Выгрузка (unload)	Выгрузка вызывается после завершения отрисовки страницы, отправки клиенту и готовности к удалению. На данном этапе свойства страницы, например, Response и Request, выгружаются, производится очистка

На протяжении всего жизненного цикла страницы вызываются события, которые могут быть использованы для управления выполнением программного кода (*Таблица 3*).

Таблица 3. События жизненного цикла страницы [6]

Событие страницы	**Типичные случаи использования**
PreInit	Событие используется: — С помощью свойства IsPostBack нужно проверить, обрабатывается ли страница в первый раз. — Создание или повторное создание динамических элементов управления
Init	Возникает после инициализации всех элементов управления и применения параметров обложки. Это событие используется для чтения или инициализации свойств элемента управления
InitComplete	Вызывается объектом Page. Это событие используется для обработки заданий, требующих завершения всех инициализаций
PreLoad	Это событие используется при необходимости обработки страницы или элемента управления до наступления события Load. После вызова события при помощи Page оно загружает состояние просмотра для себя и всех элементов управления, затем обрабатывает все данные из обратных запросов, включенных в экземпляр Request
Load	Page вызывает метод события OnLoad в Page, затем рекурсивно выполняет это действие для каждого дочернего элемента управления, до выполнения загрузки страницы и всех элементов управления. Метод события OnLoad используется для установки свойств элементов управления и создания подключения к базе данных
События элементов управления	Эти события используются для обработки определенными событиями в элементах управления
LoadComplete	Это событие используется для обработки заданий, требующих полной загрузки всех других элементов управления страницы
PreRender	До события: — Объект Page вызывает EnsureChildControls для каждого элемента управления и для страницы. — Каждый связанный элемент управления, свойство которого DataSourceID установлено, вызывает свой метод DataBind. Событие PreRender происходит для каждого элемента управления на странице. Это событие используется для внесения окончательных изменений на странице или в ее элементах управления
SaveStateComplete	До этого события ViewState сохраняется для страницы и всех элементов управления. Все изменения на странице или в ее элементах управления на данном этапе будут проигнорированы. Эти задачи используются для сохранения состояния просмотра без внесения изменений в элементы управления
Render	Это не событие; вместо этого на данном этапе обработки объект Page вызывает этот метод для каждого элемента управления. Все серверные веб-элементы управления ASP.NET обладают методом Render, который записывает разметку элемента управления, отправляемую клиенту. Обычно во время создания пользовательских элементов управления при выводе разметки элемента управления этот метод переопределяется. Однако если пользовательский элемент управления содержит только стандартные серверные элементы управления ASP.NET и не содержат пользовательской разметки, переопределять метод Render не требуется. Пользовательский элемент управления (файл

Событие страницы	Типичные случаи использования
	с расширением ASCX) содержит отрисовку по умолчанию, таким образом, нет необходимости отрисовывать элемент управления явным образом
Unload	Это событие происходит для всех элементов управления и затем для страницы. При работе с элементами управления используется для выполнения окончательной очистки определенных элементов управления, например, для закрытия подключений отдельных элементов управления к базам данных. Относительно страницы используется для выполнения окончательных действий, например, для закрытия открытых файлов и подключений к БД, а также закрытия пользовательских сеансов или других задач, связанных с запросами

℟ Страницы по умолчанию поддерживают режим автоматической обработки: если атрибут AutoEventWireup в директиве @Page страницы имеет значение true, то события страницы закрепляются за методами, которые имеют шаблон именования Page_event, а именно: Page_Load, Page_Init.

Глава 2. Особенности создания web-приложений на основе ASP.NET Web Forms

Для разработки web-приложений на основе ASP.NET Web Forms можно использовать абсолютно любой инструментарий, однако целесообразно работать в IDE от производителя технологии – MS Visual Studio Community Edition, получить которую можно на официальном сайте компании Microsoft. Интерфейс среды разработки MS Visual Studio имеет классический вариант (*Рисунок 4*), что значительно упрощает работу с ней.

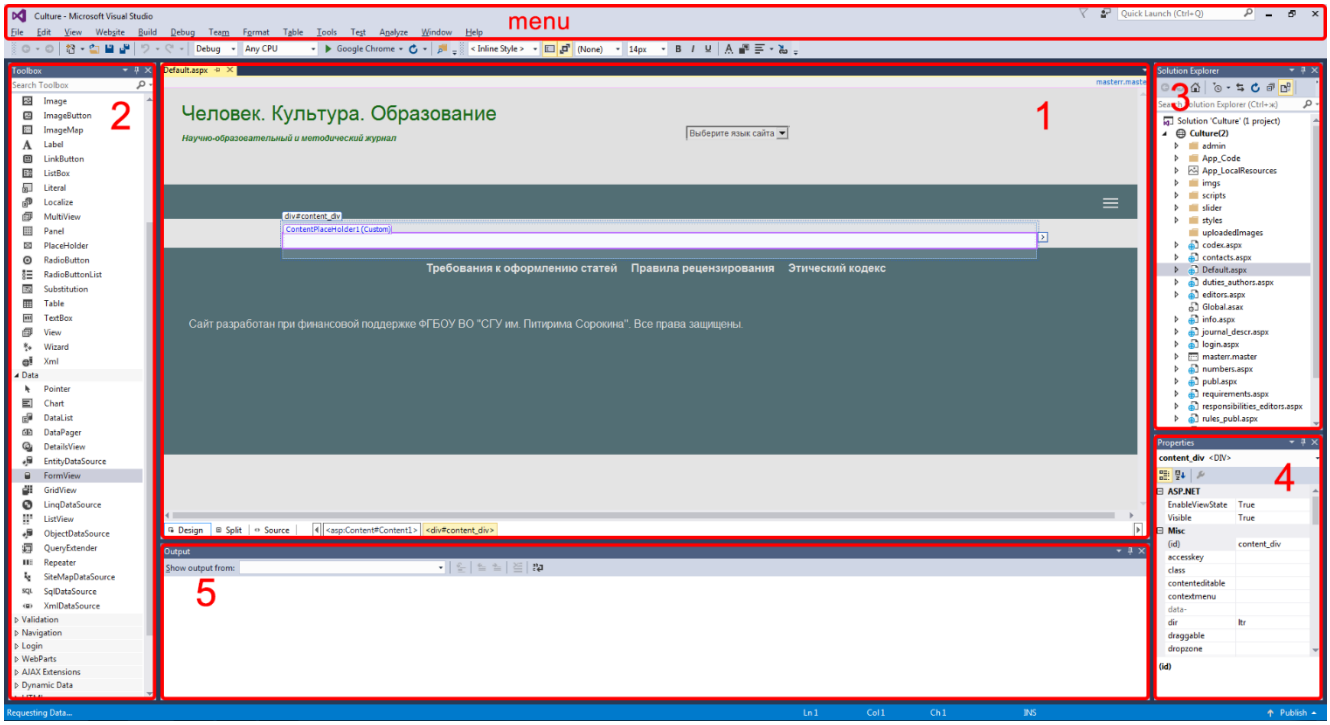


Рисунок 4. Интерфейс MS Visual Studio Community edition (v. 2015 full)

Назначение, названия и внешний вид дочерних окон среды интуитивно понятны, особенно для тех, кто занимается или занимался разработкой настольных приложений в среде Code Gear Rad Studio (Delphi 2007). По умолчанию при первичном создании сайта (предпочтительный вариант – File/New/Web site/ASP.NET Empty Web site[3]) будут открыты для работы следующие окна:

- Рабочая область для работы со страницей – позволяет работать в нескольких режимах: Design (Preview страницы), Split (смешанный вариант) и Source (XAML верстка).
- ToolBox – категорированный перечень встроенных и сторонних серверных элементов управления.
- Solution explorer – перечень файлов и директорий проекта.
- Properties/Events – свойства и события выделенного компонента.
- Output – окно вывода сообщений при компиляции и отладке проекта (при наличии одной или нескольких Breakpoint[4]), также включает в себя Error list.

Ниже (*Таблица 4*) приведен подробный перечень имеющихся окон MS Visual Studio с пояснениями.

[3] Данный вариант является предпочтительным, поскольку позволяет контролировать программисту структуру проекта и не содержит дополнительных файлов и библиотек, добавленных при создании проекта по умолчанию.
[4] Или иначе – точка останова.

Таблица 4. Часто используемые окна Visual Studio [1]

Окно	Описание
Solution Explorer	Отображает файлы и директории, находящиеся в папке веб-приложения (или проекта)
Toolbox (Панель инструментов)	Отображает встроенные серверные элементы управления ASP.NET и сторонние элементы управления или же пользовательские элементы управления, которые можно разработать самостоятельно и добавить в панель инструментов. Элементы управления могут быть написаны на любом языке и использоваться в любом языке
Server Explorer	Предоставляет доступ к базам данных, системным службам, очередям сообщений и другим серверным ресурсам
Properties (Свойства)	Позволяет конфигурировать выбранный в данный момент элемент, будь то файл в окне Solution Explorer или же элемент управления в окне проектирования веб-формы
Error List (Список ошибок)	Отображает отчеты об ошибках, которые были обнаружены Visual Studio в коде, но еще не были разрешены
Task List (Список задач)	Предоставляет перечень комментариев, которые начинаются с предопределенного моникера (moniker), позволяя отслеживать изменяемые части кода, а также быстро переходить в нужную позицию. Например, требующие внимания области могут быть помечены за счет создания комментария, начинающегося с такого предопределенного моникера, как // HACK или // TODO.
Macro Explorer (Проводник макросов)	Перечисляет файлы и подпапки, находящиеся в папке веб-приложения
Class View (Представление классов)	Отображает приложение в другом представлении, которое показывает все созданные в нем классы (вместе с их методами, свойствами и событиями)
Document (Документ)	Позволяет проектировать веб-страницу перетаскиванием, а также редактировать кодовые файлы внутри проводника Solution Explorer. Также поддерживает типы файлов, не относящиеся к ASP.NET, наподобие статических файлов HTML и XML
Team Explorer (Проводник командных проектов)	Отображает список командных проектов и позволяет изучать содержащиеся в них файлы с помощью окна Source Control Explorer (Проводник системы управления исходным кодом), чтобы получать возможность работать с ними. Появляется только в случае установки версии Visual Studio Team Suite
Manage Styles и Apply Styles	Позволяют изменять стили в связанной таблице стилей и применять их к текущей веб-странице

Создавать web-приложения можно несколькими способами:

— Web Application (формируется полноценная иерархия директорий с необходимыми файлами и данными).

— WebSite (в базовом варианте формируется отдельная директория с набором страниц и файлом настроек).

На практике предпочтительнее оказывается вариант создания WebSite, но это субъективно. Поэтому на усмотрение читателя автором ниже приводится небольшая аргументация в пользу как Web Application, так и WebSite.

Особенности варианта Web Application:

— Используется файл проекта (*.csproj) и файл решения проекта (*.sln).
— Весь код в проекте компилируется в одну сборку (один dll-файл).
— Поддерживает работу IIS (команда inetmgr в командной строке) и встроенный сервер ASP.NET Developer Server.
— Поддерживает все возможности VS (refactoring, generics, etc.) и возможности ASP.NET 2.0.

Особенности варианта WebSite:

— Нет файла проекта и файла решения проекта.
— Каждая страница компилируется в отдельную библиотеку (формируется несколько dll-файлов).
— Динамическая компиляция без необходимости пересобрать весь сайт (не требуется повторная сборка проекта в случае внесения изменений в интерфейс или программный код сайта).
— Поддерживает работу IIS и встроенный сервер ASP.NET Developer Server.
— Использует различные модели кода Code-Behind и Single File (при этом import используется как эквивалент using).

Допустимое приложение может содержать единственную веб-форму (файл с расширением *.aspx). Страница сайта, которая должна загружаться по умолчанию при запуске web-приложения, должна называться Default. Таким образом, в случае создания сайта с одной страницей с использованием модели кода Code-Behind в корневой директории проекта должны располагаться следующие файлы:

— Default.aspx – интерфейс страницы;
— Default.aspx.cs – программный код к странице;
— Web.config – файл настроек сайта (создается по умолчанию).

Каждое приложение ASP.NET может включать все перечисленные ниже компоненты:

1) Веб-формы (файлы *.aspx).
2) Мастер-страницы (файлы *.master).
3) Веб-службы (файлы *.asmx). Эти компоненты позволяют совместно использовать полезные функции приложениями, которые расположены на других компьютерах и платформах [16].
4) Обработчик HTTP (файлы *.ashx). Каждый поступающий в приложение ASP.NET запрос обрабатывается специально предназначенным для этого компонентом, который называется обработчиком HTTP (HTTP handler) [16].
5) Файлы отделенного кода [16].
6) Конфигурационный файл (web.config). В этом файле содержится множество параметров уровня приложения, которые отвечают за настройку всех аспектов, начиная с безопасности и заканчивая отладкой и управлением состоянием. Корневым элементом файла является тэг <configuration>, который содержит тэг <system.web>, – последний используется для параметров настройки ASP.NET. Внутри тэга <system.web> находятся отдельные составляющие для каждого аспекта конфигурации проекта (Таблица 5). Также имеется элемент <appSettings>, используемый для хранения специальных параметров, и элемент <connectionStrings>, применяемый для хранения строк подключения к базам данных, которые используете вы или на которые полагаются средства ASP.NET [6].

Таблица 5. Некоторые базовые разделы конфигурации <system.web> [6]

Элемент	Описание
authentication	Этот элемент конфигурирует систему авторизации — другими словами, он определяет, как будут проверяться идентификационные данные клиента, когда он запрашивает страницу
authorization	Этот элемент управляет тем, каким клиентам должен предоставляться доступ ресурсам, находящимся внутри веб-приложения или текущего каталога
compilation	Этот элемент идентифицирует версию .NET, на которую ориентировано веб-приложение (посредством атрибута targetFramework), и указывает, должны ли генерироваться символы отладки в файлах .pdb (через атрибут debug), чтобы можно было отлаживать приложение с помощью инструмента, подобного Visual Studio
customErrors	Этот элемент позволяет указывать специфичные URL-адреса, которые должны использоваться для переадресации в случае возникновения определенных (или стандартных) ошибок. Например, он может использоваться для перенаправления пользователя с неприглядной страницы ошибки 404 (page not found — страница не найдена) на более дружественную по отношению к пользователю страницу. Хотя этот параметр работает с встроенным тестовым веб-сервером Visual Studio, в IIS 7.x он заменен разделом <httpErrors>
membership	Этот элемент позволяет конфигурировать систему членства ASP.NET, которая управляет информацией пользовательских учетных записей и предоставляет высокоуровневый API-интерфейс для решения связанных с безопасностью задач, таких как вход пользователя в систему и переустановка пароля
profile	Этот элемент позволяет конфигурировать систему профилей ASP.NET, которая автоматически сохраняет и извлекает информацию по конкретному пользователю (обычно параметры профиля). Как правило, данные профилей сериализуются в базу данных
roleManager	Этот элемент позволяет конфигурировать систему безопасности на основе ролей ASP.NET, которая предоставляет способ сохранения информации о ролях и высокоуровневый API-интерфейс для авторизации на основе ролей
sessionState	Этот элемент конфигурирует различные опции, касающиеся обслуживания состояния сеанса для приложения, такие как, должно ли оно вообще поддерживаться, и если да, то где (в SQL, отдельная служба Windows и т.д.)
trace	Этот элемент конфигурирует трассировку, т.е. средство ASP.NET, которое позволяет отображать диагностическую информацию на странице (или собирать ее для отдельного просмотра)

7) Файл Global.asax. В этом файле содержатся обработчики событий, реагирующие на глобальные события приложения.
8) Другие компоненты. К их числу относятся скомпилированные сборки, в которых содержатся либо отдельные компоненты, разработанные вами, либо компоненты сторонних поставщиков, имеющие полезную функциональность. Эти компоненты позволяют отделять бизнес-логику от логики доступа к данным и создавать специальные элементы управления.Default.aspx – стартовая страница сайта [1].

Стоит отметить, что технология ASP.NET предусматривает ряд предопределенных директорий, предназначенных для хранения конкретной информации (*Таблица 6*).

Таблица 6 - Структура каталога приложения [1]

Bin (.dll)	Содержит все предварительно скомпилированные сборки .NET, которые обычно представляют собой DLL-библиотеки. Эти библиотеки используются web-приложением и могут включать предварительно скомпилированные классы web-страниц и служб, а также другие сборки, на которые ссылаются данные классы
App_Code (.vb, .cs, .xcd, …)	Содержит классы исходного кода, динамически скомпилированные для использования в рамках приложения. Обычно эти файлы кода представляют собой отдельные компоненты, такие как библиотеки доступа к данным, web-сервисы, классы и т. п.
App_GlobalRecources (.resx)	Хранит глобальные ресурсы, доступные каждой странице Web-приложения
App_LocalRecources (.resx)	Хранит локальные ресурсы, доступные только специальной странице
App_WebReferences (.wsdl)	Хранит ссылки на Web-службы, используемые приложением
App_Data (.mdb, .mdf, .xml)	Хранит файлы данных, включая XML-файлы и файлы SQL Express
App_Browsers (.browser)	Содержит определения браузера, записанные в формате XML. Эти файлы определяют характеристики браузеров на стороне клиентов и влияют на визуализацию страницы
App_Themes (.skin, .css, .xsl, …)	Хранит темы, используемые web-приложением

Если проект имеет средний уровень сложности, или подразумевает непосредственное подключение к источнику данных, то наличие предопределенных директорий ASP.NET неизбежно.

Глава 3. Особенности управления состоянием

Как говорилось ранее, механизм инициализации запроса клиента серверу запускает жизненный цикл запрашиваемой страницы. При этом на стороне сервера каждый раз создается и уничтожается экземпляр класса нужной страницы, а вместе с ним теряются абсолютно все сведения и элементы управления, имеющие отношение к данной web-форме. Однако в процессе разработки того или иного ПП на основе web-технологий возникает потребность в хранении ряда сведений при работе с web-приложением на протяжении некоторого времени. Данная задача категорируется следующим образом:

Хранение данных на стороне клиента

Хранение данных на стороне клиента осуществляется несколькими способами:

- Состояние представления ViewState – представляет собой объект словаря для конкретной страницы (используется по умолчанию), который может быть отключен разработчиком по желанию.
- Состояние элемента управления ControlState – позволяет хранить сведения, имеющие отношение к конкретному элементу управления (по умолчанию хранится в скрытых полях страницы).
- Скрытые поля HiddenField – одно поле позволяет хранить только одно значение в свойстве Value и должен быть явно добавлен на страницу как стандартный серверный элемент управления.
- Файлы cookie – представляют собой текстовые файлы для временного или постоянного (на усмотрение разработчика) хранения информации в файловой системе клиента или в памяти во время клиентского сеанса обозревателя (имеются ограничения в размере файла – до 4096 байт, в новых версиях обозревателей – до 8192 байт).
- Строки Get-запроса – возможность хранить данные в виде пар «атрибут = значение» с разделителем & в строке Get-запроса сразу после знака «?».

Хранение данных на стороне сервера

Хранение данных на стороне сервера осуществляется за счет использования следующих контейнеров глобального уровня:

1. **Состояние приложения HttpApplicationState** – глобальное хранилище, доступное со всех страниц в рамках текущего процесса web-приложения (хранится в словаре ключей/значений, создается в процессе каждого запроса к определенному адресу URL, поддерживает объекты любого типа). Переменные HttpApplicationState являются глобальными для всех пользователей и сессий и зависят только от одного текущего процесса. Не является потокобезопасным (при внесении изменений необходимо временно включать блокировку объекта приложения). *Пример: глобальный счетчик.*

2. **Состояние сеанса HttpSessionState** – глобальное хранилище, доступное со всех страниц web-приложения в рамках сеанса обозревателя (хранится в словаре ключей/значений, поддерживает объекты любого типа). В ASP.NET имеются следующие режимы работы сессии (*Таблица 7*):

- InProc (память сервера) – используется по умолчанию. Полезен для маленьких приложений и небольшого количества пользователей.
- StateServer – внепроцессный режим. Использует автономные службы ОС, не зависящие от IIS. Находится под управлением aspnet_state.exe.

- SQLServer – дает более безопасное решение в части управления сессиями, поскольку все данные сериализуются и сохраняются в MS SQL Server.
- Custom – позволяет полностью контролировать все аспекты работы с сессиями.

Таблица 7. Сравнительный анализ режимов работы сессии [7]

Режимы работы сессии	Преимущества	Недостатки
InProc	Быстрый доступ к данным[5]. Не требуется сериализация данных. Простая реализация	Если рабочий процесс или домен приложения возвращается в исходное состояние, все данные сессии теряются. Объем данных сессий и количество пользователей могут снизить быстродействие. Нельзя использовать в веб-саде и для веб-фермы
StateServer	Данные хранятся отдельно от IIS, поэтому проблемы с IIS не мешают данным сессии. Полезен в веб-ферме и веб-саде	Процесс медленный из-за сериализации и десериализации. Сервер состояний всегда должен быть запущен и работать
SQLServer	Данные сессии не страдают при перезапуске IIS. Самое надежное и безопасное управление сессиями. Доступность данных для сторонних приложений. Полезен для веб-фермы и веб-сада	Низкая скорость обработки данных. Сериализация и десериализация объекта создают дополнительные издержки. Поскольку данные сессии обрабатываются в сервере БД, он всегда должен быть запущен и работать
Custom	Не зависит от IIS, поэтому перезапуск веб-сервера никак не влияет на данные сессии. Можно написать собственный алгоритм генерации SessionID	Низкая скорость обработки данных. Создание пользовательского поставщика состояния – низкоуровневая задача

Сеансы определяются с помощью уникального идентификатора, который можно получить при помощи свойства SessionID. Если для приложения ASP.NET включено состояние сеанса, во всех запросах страницы в приложении рассматривается значение SessionID, полученное от обозревателя. Если значения идентификатора SessionID нет, ASP.NET запускает новый сеанс, а значение идентификатора SessionID этого сеанса передается обозревателю при ответе. По умолчанию значения SessionID хранятся в файле Cookie. Однако можно настроить приложение таким образом, чтобы хранить значения SessionID в URL-адресе для сеансов без поддержки файлов Cookie. *Пример: элементы в корзине для осуществления покупок пользователя.* Ниже представлены свойства и методы класса HttpSessionState, которые чаще всего используются в работе:

- Item[index] – возвращает элемент данных по его индексу.
- Item[key] – возвращает элемент данных по его ключу.
- Remove(index) – удаляет элемент данных по его индексу.
- Remove(key) – удаляет элемент данных по его ключу.
- Clear() – удаляет все данные.
- Count – возвращает общее количество элементов данных для текущей сессии.
- Abandon() – принудительно завершить сессию.
- SessionID – возвращает идентификатор текущей сессии.
- IsNewSession – возвращает true, если сессия была создана в рамках текущего запроса.

[5] Данные сессии хранятся в объекте памяти текущего домена приложения.

— Timeout – возвращает число минут, допустимое между запросами, перед тем как сессия завершится по причине таймаута (по умолчанию, 20 минут).

3. **Свойства профиля** – хранилище пользовательских данных, доступное со всех страниц web-приложения с помощью строго типизированного API (позволяет хранить объекты любого типа), альтернатива использованию полноценной БД.

4. **Использование БД** – как правило, используется вместе с файлами cookie или с состоянием сеанса (высокая безопасность, но сложная реализация).

5. **Использование Cache** – вариант глобального хранения информации в памяти сервера, которую нецелесообразно создавать повторно (данные могут быть удалены из памяти, если сервер испытывает сильное давление).

Критериями выбора конкретного решения служат ответы на следующие вопросы:

— Какому количеству пользователей должна быть доступна хранимая информация?

— Как долго информация должна храниться?

— В каком объеме необходимо хранить информацию?

— Какой уровень безопасности подходит для данного вида информации?

Таким образом, целесообразно проанализировать варианты организации хранения информации на стороне пользователя (*Таблица 8*) для понимания специфики каждого из них.

Таблица 8. Сравнительный анализ организации управления состоянием на стороне клиента

Критерий	**ViewState**	**ControlState**	**HiddenField**	**Файлы cookie**	**Строка get-запроса**
Ресурсы сервера	Не требуются	Не требуются	Не требуются	Не требуются	Не требуются
Срок хранения	Короткий	Короткий	Короткий	Длительный	Короткий
Уровень сложности	Простой	Простой	Простой	Простой	Простой
Надежность	Низкая (может быть отключен)	Высокая	Высокая	Низкая (могут быть отключены)	Высокая
Скорость работы	Зависит от объема информации	Зависит от объема информации	Относительно высокая	Высокая (без кода)	Высокая
Ограниченность в объеме	Нет	Нет	Да	Да	Да
Ограниченность устройств	Имеется	Имеется	Нет	Нет	Нет
Использование кода	Да	Да	Нет	Возможно	Да
Безопасность	Данные хэшируются, сжимаются и кодируются. Угроза безопасности при просмотре источника выходных данных	Низкая	Поле может быть подделано; простая архитектура хранилища	Файлы могут быть взломаны	Информация доступна для просмотра

Таким образом, организация управления состоянием на стороне клиента в большинстве случаев не использует программный код и по своей природе реализуется достаточно просто. При этом производительность решения в большинстве случаев зависит от объема хранимой информации.

Аналогичным способом можно сравнить варианты организации хранения состояния на стороне сервера (*Таблица 9*).

Таблица 9. Сравнительный анализ организации управления состоянием на стороне сервера

Критерий	Состояние приложения Http Application State	Состояние сеанса HttpSession State	Свойства профиля	БД	Cache
Ресурсы сервера	Требуются	Требуются	Требуются	Требуются	Требуются
Срок хранения	Ограниченный (при удалении процесса)	На протяжении сессии	Высокий (должны быть предусмотрены механизмы очистки)	Высокий	Относительно высокий (до перезагрузки сервера или приложения иди до появления давления на сервер)
Уровень сложности	Простой	Простой	Сложный	Очень сложный	Простой
Универсальность	В рамках одного процесса	В рамках сессии	Да	?	
Масштабируемость	Невысокая	Да	Да		
Расширяемость	?	Да	Да		
Количество пользователей	Несколько	Один	Один		Несколько
Надежность	Высокая	Высокая	Высокая	Очень высокая	Слабая
Скорость работы	Зависит от объема данных и параметров сервера (память сервера)	Зависит от объема данных (память сервера)	Невысокая (за счет хранилища данных)	Невысокая (зависит от количества запросов к БД)	Высокая
Использование кода	Да	Да	Да	Да	Да
Безопасность	Высокая	Рекомендуется использовать модуль сеанса и SSL для безопасности	?	Высокая	?

Стоит отметить, что из всех рассмотренных выше решений (*Таблица 9*) для организации глобального хранилища на стороне сервера чаще всего используется вариант состояния сеанса, поскольку довольно часто необходимо организовать личный кабинет пользователя на сайте интернет-магазина или web-портала. Свойства профиля целесообразно использовать для организации сайтов знакомств, социальных сетей или игр в интернет-среде.

Глава 4. Использование шаблонов элементов управления в проекте

Элементы управления в проекте ASP.NET можно условно поделить на две категории – специальные серверные (автоматически генерируют собственную разметку и компилируются в отдельные DLL-сборки) и пользовательские (имеют статическую HTML-разметку, могут использоваться многократно). Основные различия между данными категориями компонентов приведены ниже (*Таблица 10*).

Таблица 10. Сравнительный анализ элементов управления

Критерий сравнения	**Пользовательские**	**Серверные**
Начальная директива	Control	Page
Расширение файла	*.ascx	*.aspx
Наследование файлов кода	UserControl	Page
Возможность запроса браузером клиента	Не могут запрашиваться клиентским браузером	Могут запрашиваться клиентским браузером

Заметим, что специальные серверные элементы управления можно создавать как вручную, так и с помощью интерфейса IDE – выбрать последовательно Projects / Add New Item (Проект / Добавить новый элемент), перейти в раздел Visual C# Items / Web (Элементы Visual C# / Веб) и указать нужный шаблон ASP.NET. Рассмотрим ниже (*Таблица 11*) галерею шаблонов специальных элементов управления ASP.NET, а также перечень поддерживаемых файлов в ASP.NET для расширения функционала проекта.

Таблица 11. Назначение специальных серверных элементов упарвления

Название элемента	**Назначение элемента**
Web Configuration File	Файл, который содержит параметры настройки веб-приложения
Site Map	Файл, используемый для создания карты сайта
Class	Файл для определения классов
	Форматы работы со стилями
Style Sheet (*.css)	Файл каскадной таблицы стилей (CSS) для расширения возможностей стилистичекого оформления веб-страниц и размещённых на них элементов по сравнению с простым HTML
SCSS Style Sheet (или SASS)	SASS (Syntactically Awesome Stylesheets) — модуль, включенный в HAML[6]. SASS — это метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS-кода и упрощения файлов каскадных таблиц стилей [4]. Имеет два синтаксиса: - SASS — не содержит в синтаксисе фигурных скобок (заменены отступами). - SCSS — содержит в синтаксисе фигурные скобки. SCSS является языком, который компилируется в CSS
LESS Style Sheet (*.less)	Leaner Style Sheets, компактная таблица стилей. Это динамический язык стилей, надстройка над стандартным CSS. Он создан под влиянием языка стилей SASS и, в свою очередь, оказал влияние на его новый синтаксис «SCSS», в котором также использован синтаксис, являющийся расширением CSS [3]. LESS является языком, который компилируется в CSS «на лету» (с помощью less.js)[7] или предварительно
	Поддерживаемые форматы файлов

[6] HAML – это язык разметки для упрощённой генерации HTML. HAML компилируется в HTML.
[7] LESS-файл прикрепляется к проекту строго перед JS, после чего работает как обычный CSS.

Название элемента	Назначение элемента
JavaScript File	JavaScript – это скриптовый язык, предназначенный изначально[8] и в основном для Frontend[9]. используется для реализации следующих часто используемых задач: - математические операции; - обработка и валидация данных в формах; - взаимодействие с пользователем сайта и обработка событий; -взаимодействие с элементами формы, управление содержимым и стилями элементов; - добавление анимации и графики; - и другое. Клиентский JavaScript-код можно применить в проекте четырьмя способами: - встроенные сценарии за счет парного тэга <script>; - как внешний файл атрибутом src тега <script>; - использовать в обработчике события в качестве значения HTML-атрибута (например, onclick или onmouseover); - как тело URL-адреса, использующего специальный спецификатор псевдопротокола JavaScript
JSON File	JavaScript Object Notation. Файл текстового формата обмена данными как между браузером и сервером, так и между серверами. Основан на JavaScript. Использует два возможных формата: - набор пар «ключ: значение»; - набор значений, чаще всего в виде массива, списка, вектора или последовательности
XML File	eXtensible Markup Language – расширяемый[10] язык разметки. Описывает XML-документы и частично XML-процессоры. Файл текстового формата обмена данными
XSLT File	eXtensible Stylesheet Language Transformations – язык преобразований XML-документов в XML/HTML/XHTML/TXT. Был разработан как часть расширенного языка стилей XSL. Позволяет реализовать схему, при которой данные хранятся отдельно, а их представления отдельно
Resource File	Файл для хранения ресурсов, которые могут содержать объекты пользовательского интерфейса или данные локализации
JSX File	XML-подобное расширение JavaScript. Представляет собой своеобразный язык шаблонов с возможностями JavaScript (производительность JSX выше обычного JavaScript). Файлы JSX используются библиотекой React.js для более структурированного UI
CoffeeScript File	CoffeeScript – это язык программирования, разработанный изначально на Ruby и транслируемый в JavaScript. Используется для повышения читабельности, компактности и сжатия по размеру программного кода
Skin File	Файл, используемый для определения темы (стиля) ASP.NET
Browser File	XML-файл, который предназначен для хранения данных о браузерах
Text File	Стандартный текстовый файл
Дополнительные файлы описаний (для схем и диаграмм)	
Class Diagram	Диаграмма классов, демонстрирующая общую структуру иерархии классов системы, их кооперации, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними
JSON Schema File	Файл, который используется для описания формата данных JSON-файла
XML Schema	Шаблон для создания схемы для XML-документов
Основные и часто используемые шаблоны	

[8] Долгое время JavaScript был единственным языком программирования для браузеров (использовался как инструмент для интерактивности сайтов). На сегодняшний день JavaScript за счет Node.js стал более универсальным и вышел за пределы браузеров.

[9] Клиентская часть пользовательского интерфейса.

[10] Разметка, используемая в документах, не фиксируется языком, а прописывается разработчиком согласно решаемым задачам.

Название элемента	Назначение элемента
HTML Page	Шаблон HTML-страницы, в которой может быть размещен клиентский код
Web Form	Компонентно-ориентированный шаблон web-формы, который позволяет создавать динамические веб-сайты с использованием модели на основе событий
Master page	Элемент, который позволяет задавать единообразныую структуру веб-сайта и управлять страницами, а также упорядочивает их согласно бизнес-логике веб-приложения, однако сам элемент Master Page не является страницей
Content page	Страница, которая содержит элементы управления контентом и использует Master page в качестве основы
Layout page	Элемент, который определяет макет - шаблон верхнего уровня для всех страниц
Empty page	Шаблон пустой страницы без каких-либо элементов
Helper	Вспомогательный многократно используемый компонент
SQL Server Database	БД на основе СУБД MS SQL Server
ADO.NET Entity Data Model (ADO.NET EDM)	ADO.NET Entity Framework (EF) представляет собой объектно-ориентированную технологию доступа к данным, формируя Data-логику на основе модели данных (EDM). В качестве элемента управления источником данных используется EntityDataSource
DataSet	Независимый от Data Source и, соответственно, базы данных объект, который предоставляет автономный доступ к данным
LINQ to SQL Classes	Классы сущностей LINQ to SQL, представляющие собой таблицы SQL
Dynamic Data Field	Пользовательский элемент управления Dynamic Data, который предназначен для настройки отображения полей данных
Web user control	Серверный элемент управления, создаваемый при помощи конструктора форм (visual designer)
Варианты web-сервисов	
Web Service (ASMX)	Класс для создания веб-службы (веб-сервиса) для обмена данными, используя SOAP только через HTTP. ASMX-сервисы могут быть размещены только в IIS, используют для сериализации класс XmlSerializer
WCF Service	Windows Communication Foundation – программная платформа от Microsoft, предназначенная для разработки распределенных сетевых сервисно-ориентированных приложений и организации обмена данными. Стандарт организации сетевых взаимодействий в приложениях для dotNET. WCF-сервис поддерживает обмен данными любого стандарта (по умолчанию SOAP) через любой транспортный протокол (HTTP, TCP/IP, MSMQ, Named Pipes). WCF-сервисы используют DataContractSerializer (лучше по производительности, чем XmlSerializer)
Web API Controller Class	Класс контроллера обработки HTTP-запросов веб-API
WCF Data Service	Службы данных, которые позволяют модифицировать и поставлять данные по интерфейсу HTTP REST
Дополнительные шаблоны	
EF 5.x DbContext Generator	Базовый класс Entity Framework, который предоставляет широкие возможности по работе с базами данных (создание запросов, отслеживание изменений и сохранение данных в базе)
Generic Handler	Страница, на которой размещается обработчик для осуществления динамического возврата данных
OWIN Startup Class	Класс, предназначенный для настройки приложения, использующего шаблон OWIN (уровень абстракции, который изолирует веб-приложения из среды, в которой они размещены)
Runtime Text Template	C# шаблон, который задает класс, используемый для генерирования текста, выполняемого во время исполнения веб-приложения
Text Template	C# шаблон, который используется для генерирования текста
SignalR Hub Class	Класс протокола взаимодействия высокого уровня, который позволяет клиенту и серверу напрямую вызывать методы друг друга

Название элемента	Назначение элемента
SignalR Persistent Connection Class	Класс низкоуровневого протокола взаимодействия, где подключения представляют конечную точку, к которой подключаются клиенты
Шаблоны для работы с Silverlight	
Silverlight 1.0 JavaScript Page	JavaScript-страница приложения Silverlight
Silverlight Application	Шаблон приложения Silverlight
Silverlight-enabled WCF Service	Шаблон элемента службы WCF с поддержкой Silverlight, который создает веб-службу, предоставляющую данные клиенту Silverlight или внешнему интерфейсу

Таким образом, достаточно внушительный перечень поддерживаемых форматов файлов (в том числе используемый в технологиях обмена данными), дополнительные файлы схем и структур данных, действующие варианты web-сервисов, а также специфика и галерея шаблонов серверных элементов управления говорят об универсальности и многогранности технологии ASP.NET.

Глава 5. Коллекция серверных веб-элементов управления, используемых в проекте

ASP.NET предлагает широкий перечень серверных элементов управления, которые можно условно поделить на несколько категорий:

- Серверные элементы управления HTML (вкладка HTML).
- Базовые элементы управления (вкладка Standard, таблица).
- Многофункциональные элементы управления (вкладка Standard) – усовершенствованные элементы управления, такие как, например, Calendar, AdRotator и TreeView.
- Элементы управления проверкой достоверности (вкладка Validation) – позволяют проверять достоверность введенной пользователем информации в базовые элементы управления.
- Элементы управления данными (вкладка Data) – содержат сложные элементы управления источниками данных, таблицы, формы и списки для работы с большими объемами данных.
- Элементы управления навигацией (вкладка Navigation) – предназначены для отображения карт сайта.
- Элементы управления входом в систему (вкладка Login) – поддержка аутентификации пользователя с помощью форм.
- Элементы управления Web Parts – набор элементов управления для построения компонентных и легко конфигурируемых порталов.
- Элементы управления ASP.NET AJAX – позволяют использовать приемы Ajax без написания программного кода.
- Элементы управления ASP.NET Dynamic Data – позволяют создавать управляемые данными сайты на основе шаблонов без написания кода.

Базовые и навигационные элементы управления составляют хорошую основу для разработки статичных страниц или сайтов-визиток без взаимодействия со сторонними источниками данных (например, различные СУБД). Элементы управления категории Validation, как правило, необходимы для проверки достоверности введенной пользователем информации, что автоматически делает эту категорию элементов незаменимой с точки зрения безопасности.

При разработке более серьезных web-сайтов или порталов (а их неотъемлемое большинство) необходимо использование различных внешних источников данных, что подчеркивает незаменимость серверных элементов типа Data. Характер компонентов для входа в систему немного противоречив – некоторые из них значительно упрощают жизнь разработчику, сводя написание программного кода к минимуму, а некоторые можно не использовать вовсе ввиду морального устаревания или специфики технологии, на основе которой реализован данный элемент.

Знание специфических серверных элементов категорий Web Parts, ASP.NET AJAX и ASP.NET Dynamic Data не является строго обязательным, поскольку данные компоненты призваны снизить нагрузку разработчика, при этом они не являются фундаментальными.

Глава 6. Элементы управления данными

Доступ к источникам данных в web-приложении можно получать как на уровне программного кода (используя пространства имен System.Data, System.Xml), так и на декларативном уровне, не задействовав, в основном, программный код, а именно:

1) Подключение к источнику данных.
2) Выбор и просмотр данных (поддерживается элементами управления с привязкой к данным).
3) Вставка, обновление и удаление данных (чаще всего поддерживается элементами управления с привязкой к данным).
4) Сортировка (чаще всего поддерживается элементами управления с привязкой к данным) и фильтрация данных (частично кодируется).
5) Постраничное разбиение и кэширование данных (чаще всего поддерживается элементами управления с привязкой к данным).

Элементы управления данными можно условно поделить на две категории – элементы управления источником данных и элементы управления с привязкой к данным.

Элементы управления источником данных

Предназначены для организации подключения к источнику данных (предпочтительно через строку подключения, прописанную в конфигурационном файле проекта, свойство ConnectionString), а также для организации работы с данными (просмотр, добавление, обновление, удаление, сортировка, фильтрация, кэширование информации) посредством SQL-запросов четырех видов – Select, Insert, Update, Delete. Таким образом, согласно действующей категоризации SQL-запросов у элементов управления источником данных имеются соответствующие свойства – SelectCommand, InsertCommand, UpdateCommand и DeleteCommand, для каждого из которых устанавливается режим работы Text (указав текст запроса) или StoredProcedure (через хранимую процедуру). Например, для свойства SelectCommand имеется дополнительное свойство SelectCommandType, в котором указывается вариант работы со свойством SelectCommand – Text и StoredProcedure.

Запросы в свойствах SelectCommand, InsertCommand, UpdateCommand и DeleteCommand могут быть статическими (указываются разово в XAML-верстке) и динамическими (указываются на уровне программного кода один или несколько раз в зависимости от бизнес-логики). В ASP.NET предусмотрен ряд параметров, которые можно использовать для построения параметризированного запроса (*Таблица 12*).

Таблица 12. Типы параметров запроса [1]

Источник	Дескриптор	Описание
Свойство элемента управления	<asp:ControlParameter>	Свойство другого элемента управления на странице
Строковое значение запроса	<asp:QueryStringParameter>	Значение из текущей строки запроса
Значение состояния сеанса	<asp:SessionParameter>	Значение, сохраненное в текущем сеансе пользователя
Значение cookie-набора	<asp:CookieParameter>	Значение из любого cookie-набора, присоединенного к текущему запросу
Значение профиля	<asp:ProfileParameter>	Значение из текущего пользовательского профиля
Переменная формы	<asp:FormParameter>	Значение, отправленное странице от элемента управления вводом. Обычно вместо этого будет использоваться

Источник	Дескриптор	Описание
		свойство элемента управления, но если у соответствующего элемента управления отключена поддержка состояния представления, может понадобиться извлечь значение непосредственно из коллекции Forms
Значение маршрута	<asp:RouteParameter>	Значение из маршрутизированного URL
Программная установка	<asp:Parameter>	Базовый класс, от которого наследуются все прочие параметры. Никогда не устанавливается автоматически, поэтому он имеет смысл, когда для установки значения параметра вручную используется код

Элементами управления источником данных являются:

— **AccessDataSource** – элемент управления источником данных для работы с Microsoft Access.
— **EntityDataSource** – серверный элемент управления источником данных, формирующий Data-логику (декларативное связывание данных) на основе Entity Data Model (EDM). Доступен начиная с версии .NET Framework 3.5.
— **LinqDataSource** – элемент управления источником данных, который позволяет использовать технологию LINQ[11]. Работает с массивами и коллекциями, с технологией Entity Framework, с XML-файлами, с объектом DataSet, взаимодействует с MS SQL Server.
— **ObjectDataSource** – достаточно сложный элемент управления, который использует в качестве источника данных бизнес-класс (в терминологии Microsoft это бизнес-объект, описывающий бизнес-логику работы с информацией без привязки к ее содержимому) со всеми необходимыми методами (например, для организации работы Select, Insert, Update, Delete), свойствами и событиями.
— **SiteMapDataSource** – серверный элемент управления, который необходим для размещения и отображения карты сайта; используется совместно с Web.sitemap (взаимодействие с источником данных Web.sitemap) и TreeView/Menu (отображение на странице).
— **SqlDataSource** (часто используемый на практике) – элемент управления источником данных для работы с классическими СУБД.
— **XmlDataSource** – элемент управления источником данных для работы с XML-файлами.

Элементы управления с привязкой к данным

Элементы управления источником данных используются совместно с элементами управления с привязкой к данным. К последним относят:

— **Элементы управления списка**:
  - BulletedList – маркированный список элементов.
  - CheckBoxList – группа флажков с множественным выбором.
  - DropDownList – раскрывающийся список.
  - ListBox – стандартный список элементов, позволяющий выбрать одно значение.
  - RadioButtonList – группа флажков без возможности множественного выбора.
— **GridView** – отображает все значения источника данных в виде классической таблицы. Позволяет выделять записи, сортировать данные и разбивать их постранично, а также

[11] С английского Language-Integrated Query – язык запросов к источнику данных. Аналогичен SQL, но используется на уровне разметки страницы или напрямую в программном коде.

изменять и удалять записи. Поддерживает функции автоматического выбора, обновления и удаления данных. Не поддерживает режим добавления записей.

— **ListView** – отображает все значения источника данных, используя определенные разработчиком шаблоны. Позволяет выделять, добавлять, изменять, удалять и сортировать элементы, разбивать данные постранично, а также группировать элементы. Поддерживает функции автоматической сортировки, добавления, обновления и удаления данных (при соединении с источником данных). Не отображает данные без отрисовки шаблона.

— **DetailsView** – отображает значения источника данных по одной записи за раз в виде таблицы. Однако при использовании функции постраничного разбиения данных (по умолчанию отключена) позволяет переходить к другим записям в источнике данных. Позволяет выделять, добавлять, изменять, удалять записи, разбивать данные постранично. Поддерживает функции автоматического выбора, добавления, обновления и удаления данных, а также разбиения на страницы (при соединении с источником данных). Заимствует часть модели GridView.

— **FormView** – отображает значения источника данных по одной записи за раз в относительно свободном формате (обязательная поддержка шаблонов). Однако при использовании функции постраничного разбиения данных (по умолчанию отключена) позволяет переходить к другим записям в источнике данных. Позволяет выделять, добавлять, изменять и удалять записи, а также разбивать данные постранично. Поддерживает функции автоматического выбора, добавления, обновления и удаления данных, а также разбиения на страницы (при соединении с источником данных). Заимствует часть модели GridView. Не отображает данные без отрисовки шаблона.

— **DataList** – отображает все значения источника данных, используя шаблоны. Имеет шаблон по умолчанию. Позволяет выделять, добавлять, изменять и удалять записи. Не позволяет разбивать данные постранично и не группирует их.

— **Repeater** – отображает все значения источника данных в виде шаблонного списка. Максимально «легкий» и производительный. Не поддерживает выделение, добавление, редактирование, удаление записей. Не поддерживает сортировку данных, постраничное разбиение и автоматическое создание столбцов. Не отображает данные без отрисовки шаблона.

— **TreeView** – генерирует многофункциональные представления деревьев или неоформленных иерархических данных для отображения общей структуры web-приложения.

— **Menu** – отображает меню сайта.

Очевидно, что каждый из представленных выше элементов управления имеет свою специфику (*Таблица 13*).

Таблица 13. Сравнительный анализ элементов управления с привязкой к данным

Критерий	GridView	ListView	DetailsView	FormView	DataList	Repeater
Шаблон	Нет	Обязателен	Да	Обязателен	Да	Нет
Количество записей	Все	Все	Одна за раз	Одна за раз	Все	Все
Select	Да	Да	Да	Да	Да	Нет
Insert	Нет	Да	Да	Да	Да	Нет
Update	Да	Да	Да	Да	Да	Нет
Delete	Да	Да	Да	Да	Да	Нет
Сортировка	Да	Да	Нет	Нет	Нет	Нет
Постраничное разбиение	Да	Да	Да	Да	Нет	Нет
Группировка	Нет	Да	Нет	Нет	Нет	Нет

Таким образом, самым ограниченным по функционалу элементом является Repeater (но говорит о высокой производительности), а самым «насыщенным» – ListView.

Глава 7. Элементы управления входом в систему

К элементам управления входом в ASP.NET относятся элементы, позволяющие реализовать регистрацию новых пользователей, проверку подлинности для их идентификации, отображение информации о вошедших пользователях, а также изменение/восстановление существующего пароля. По умолчанию данные элементы используют функционал членства ASP.NET, которое обеспечивает проверку и хранение учетных данных пользователей в совокупности с проверкой подлинности форм на веб-узлах.

К элементам управления входом относят:

- **Login** – отображает классическую форму для аутентификации и авторизации пользователей сайта. Данный элемент позволяет выполнять действия перед аутентификацией (событие LoggingIn), после нее (событие LoggedIn, следующее после удачной попытки входа) и в случае возникновения ошибки (событие LoginError, следующее после неудачной попытки входа). Обработчик событий Authentificate подключается при необходимости самостоятельной идентификации пользователя без использования функционала членства ASP.NET.
- **LoginView** – отображает разные наборы элементов управления (различное содержимое) в зависимости от того, вошел пользователь в систему (шаблон LoggedInTemplate) или нет (шаблон AnonymousTemplate), а также в зависимости от роли пользователя, прошедшего аутентификацию.
- **LoginStatus** – отображает ссылку, перенаправляющую анонимных пользователей на страницу аутентификации, или ссылку, меняющую текущий статус пользователя на анонимный.
- **LoginName** – отображает имя пользователя.
- **PasswordRecovery** – отображает форму восстановления пароля, запрашивая логин пользователя. Пароль высылается на адрес электронной почты, привязанный к учетной записи пользователя. При использовании необратимого шифрования вместо прежнего пароля автоматически генерируется и отправляется новый. В том случае, если включена опция контрольного вопроса, данный элемент управления запустит процедуру восстановления пароля только после верного ответа на контрольный вопрос. Для корректной работы элементу управления необходим доступ к серверу SMTP. Текст сообщения, шаблон которого прописывается в файле конфигурации проекта web.config, настраивается с использованием свойства MailDefinition.
- **CreateUserWizard** – отображает форму для сбора информации о новом пользователе и его регистрации. После отправки заполненной формы данный элемент управления выводит подтверждающее сообщение. По умолчанию запрашивается имя пользователя, пароль и подтверждение пароля, адрес электронной почты, контрольный вопрос и ответ на него.
- **ChangePassword** – отображает форму замены пароля, на которой запрашивается имя пользователя и старый пароль. При правильно указанном старом пароле происходит его замена новым.

Глава 8. Валидация данных

В процессе формирования бизнес-логики web-приложения необходимо учитывать все аспекты – как внутренние, так и внешние. Одним из таких факторов является пользовательский ввод данных, которые затем обрабатываются на стороне сервера и сохраняются в БД проекта. Это означает, что семантика данных может создавать потенциальную угрозу для проекта. Следовательно, все вводимые пользователем сведения подлежат обязательной проверке, которую удобно реализовать с помощью серверных компонентов проверки достоверности вводимой информации, а именно:

- **RequiredFieldValidator** – проверяет содержимое конкретного элемента управления (базового) на пустоту.
- **RangeValidator** – контролирует, принадлежит ли введенное пользователем значение допустимому диапазону значений. Работает с числом, датой и строкой (используется таблица символов ASCII).
- **CompareValidator** – контролирует, подходит ли содержимое серверного компонента для конкретной операции сравнения со значением другого серверного компонента или константы.
- **RegularExpressionValidator** – проверяет, является ли содержимое стандартного элемента допустимым на основе указанных регулярных выражений.
- **CustomValidator** – позволяет организовывать валидацию данных как на стороне клиента, так и на сервере.
- **ValidationSummary** – отображает конечную информацию о результатах проверки данных на текущей странице в виде пользовательских сообщений для каждого элемента управления, содержимое которого не прошло проверку.

Исходя из специфики элементов управления проверкой достоверности становится ясно, что технология ASP.NET позволяет защитить web-приложение от различных SQL-инъекций, регулярных выражений, а также от элементарных ошибок пользователя при работе с приложением как на уровне клиента, так и на уровне сервера.

ЧАСТЬ 2. ASP.NET В ЗАДАЧАХ И ПРИМЕРАХ

Создание статичных страниц с использованием CSS-стилей

Формулировка задачи. Разработать простой сайт, состоящий из одной статичной страницы

Уровень сложности: простой.

Решение

Создать проект можно путем выбора пункта меню *File/new/website/ASP.NET Empty Web Site…*, указав в левой части язык программирования C# и расположение проекта в нижней части окна. Добавить страницу в качестве элемента проекта можно, выбрав *Add/Add new Item*, выбрав в окне *Добавление нового элемента* пункт *Web Form*.

Структура страницы может выглядеть следующим образом (*Рисунок 5*):

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="styles/StyleSheet.css" />
    <title>Минералы Севера</title>
</head>
<body>
    <form id="form1" runat="server">
        <div class="wrapper">

            <div class="header">
            </div>

            <div class="slider">
            </div>

            <div class="content">
            </div>

            <div class="footer">
            </div>

        </div>
    </form>
</body>
</html>
```

Рисунок 5. Предлагаемая структура страницы

Каскадные таблицы стилей подключаются к проекту следующим образом:

```
<link rel="stylesheet" type="text/css" href="styles/StyleSheet.css" />
```

R файл стилей создается через обозреватель решения путем создания директории «styles» в корне проекта, после чего необходимо добавить элемент `StyleSheet` через контекстное меню.

Содержание файла стилей может быть произвольным.

R Для проектирования и разработки статичной страницы разрешается использовать серверные элементы управления вкладки Standard, а также стандартные тэги HTML5.

Определение подчиненности страниц в структуре web-приложения

Формулировка задачи. Необходимо разработать несложный сайт, используя блочную верстку, с организацией подчиненности страниц со следующей структурой (*Рисунок 6*).

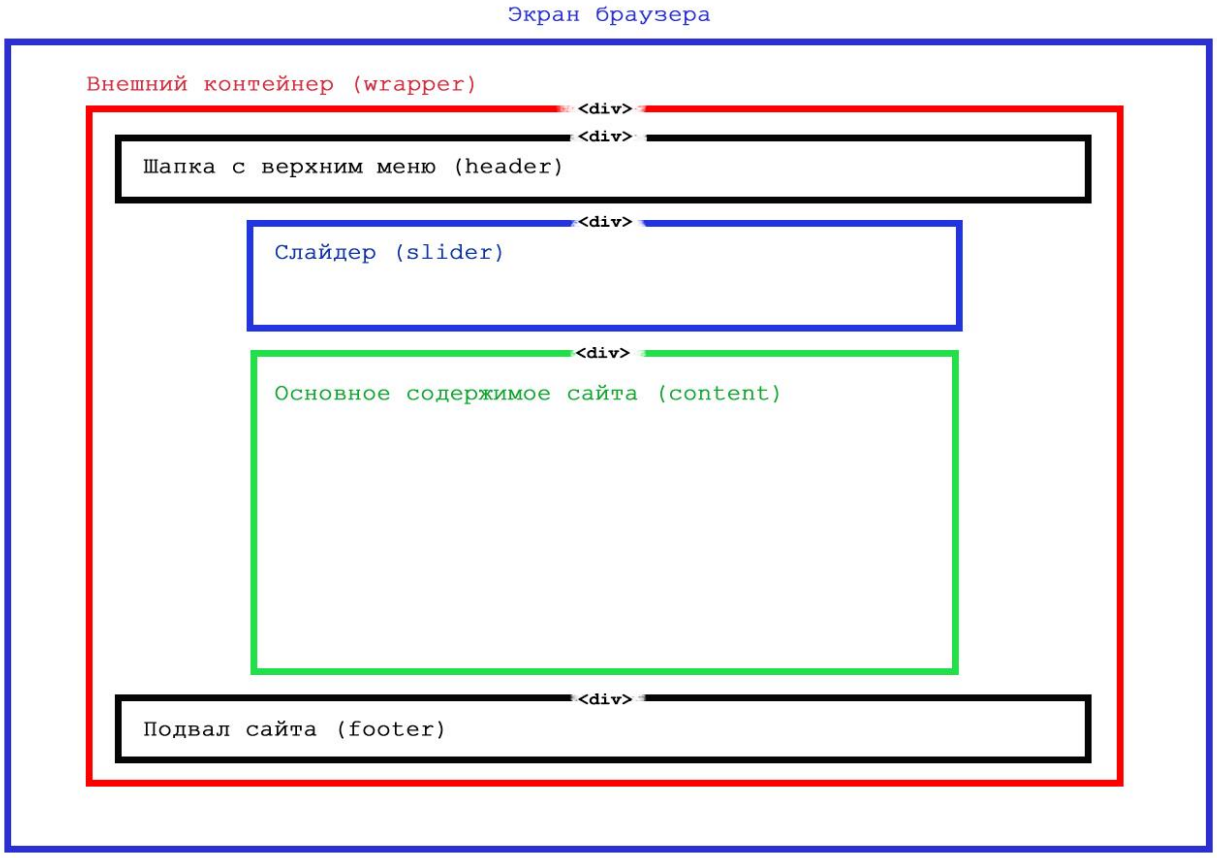


Рисунок 6. Структура сайта, которую требуется реализовать

Теоретическая справка. Элемент Master Page представляет собой возможность управления страницами web-приложения (после добавления в проект Master Page при каждом последующем добавлении элемента Web Form необходимо выбрать пункт Select Web master), а также позволяет задавать структуру web-сайта. Однако стоит заметить, что сам элемент Master Page не является страницей, он только позволяет управлять страницами и упорядочивает их согласно бизнес-логике web-приложения. Таким образом, без наличия в web-приложении хотя бы одной страницы проект будет выводить в окно браузера ошибку.

Уровень сложности: простой.

Решение

Создать проект можно путем выбора пункта меню File/new/website/ASP.NET Empty Web Site…, указав в левой части язык программирования C# и расположение проекта в нижней части окна. Добавить мастер страниц можно путем выбора в окне Solution Explorer посредством контекстного меню пункт Add/Add new Item, указав в окне *Добавление нового элемента* пункт *Master Page*. Для задания структуры сайта удобней работать в режиме Design (листинг 1):

Листинг 1. Структура Master Page

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="qwe.master.cs" Inherits="qwe" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no" />
    //подключение файла CSS-стилей
    <link rel="stylesheet" type="text/css" href="styles/StyleSheet.css" />
    //подключение bootstrap-стилей (можно взять с официального сайта)
    <link rel="stylesheet" type="text/css" href="styles/bootstrap.min.css" />
    //подключение скриптов
    <script src="scripts/bootstrap.min.js"></script>
    <title>Книга памяти Республики Коми</title>
</head>

<body>
    <form id="form1" runat="server">
        <div class="wrapper">

            <div class="header>
            </div>

            // область сайта с меняющимся содержимым (в зависимости от пункта меню)
            <div class="content" style="padding-top: 60px; margin-top: 60px;">
                <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
                </asp:ContentPlaceHolder>
            </div>

            //блок footer, или подвал сайта
            <div class="footer style="width:100%; height:160px;">
            </div>

        </div>
    </form>
</body>
</html>
```

Дополнительно к содержимому элемента Master Page необходимо использовать CSS-стили (в окне Solution Explorer посредством контекстного меню пункт Add/Add new Item, указав в окне *Добавление нового элемента* пункт *Style Sheet*), указанные ранее (листинг 2):

Листинг 2. Содержимое файла стилей StyleSheet.css

```
html, body {
    font-size: 16px;
    font-family: 'Open Sans', sans-serif;
    background-color: #fff;
    margin: 0;
    padding: 0;
    height: 100%;
    width: 100%;
}

.wrapper {
    min-height: 100%;
    height: auto !important;
    width: 100%;
}

.header {
    background-color: #897862;
    background-color: rgba(137, 120, 98,
0.85);
    font-size: 14px;
}

.header a {
    display: block;
    text-decoration: none !important;
    padding: 5px 25px;
    font-size: 114%;
    color: #fff;
}

.header a:hover {
    background-color: #fff !important;
    color: #000;
}

.icon-bar {
    background-color: #fff;
}

.navbar {
    margin-bottom: 0px;
    min-height: 40px;
    width: 100%;
}

.navbar-toggle {
    padding: 4px 10px;
}
```

```
.nav-pills > li > a, .nav-list > li > a {
    border-radius: 8px;
}

.nav-pills > li.active > a,
.nav-pills > li.active > a:hover,
.nav-pills > li.active > a:focus {
    color: #fff;
    background-color: #B8AF3D;
}

@media only screen and (max-width: 768px) {
       .nav-pills>li {
           float: none;
       }
}

.active>a {
    background-color: #fff !important;
    color: #897862 !important;
}

.active>a:hover {
    background-color: #fff !important;
    color: #897862 !important;
}

.content {
    padding-top: 40px;
    padding-bottom: 30px;
    width: 100%;
    color: #897862;
}

.footer {
    background-color: #484231;
    font-size: 16px;
    padding: 0;
    height: 160px;
    color: #fff;
    margin: 0 auto;
    line-height: 60px;
    width: 70%;
    position: relative;
}
```

Шапка сайта может иметь любой вид на усмотрение разработчика, однако автором предлагается в качестве примера следующая структура (листинг 3):

Листинг 3. Пример меню сайта с использованием bootstrap и базовых элементов управления

```
<div class="header navbar navbar-fixed-top">
                <div class="container-fluid" style="padding-top: 4px;">
                    <div class="navbar-header">
                        <button type="button" class="navbar-toggle" data-toggle="collapse"
                            data-target=".navbar-collapse">
                            <span class="sr-only">Toggle navigation</span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                        </button>
                        <div class="navbar-collapse collapse">
                            <ul class="nav nav-pills">
                                <li><asp:HyperLink ID="HyperLink2" runat="server"
NavigateUrl="~/default.aspx" Text="ГЛАВНАЯ" /></li>
                            </ul>
                        </div>
                    </div>
                </div>
</div>
```

Большинство классов, используемых в листинге, являются реализацией bootstrap, что значительно упрощает написание собственных стилей.

Далее необходимо добавить в проект элемент Web Form для создания простой страницы default.aspx, указав при этом нужный Master Page в настройках (пункт Select Master Page в нижней части окна). XAML-верстка страницы будет иметь довольно простой вид (листинг 4).

Листинг 4. XAML-верстка страницы default.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/qwe.master" AutoEventWireup="true"
CodeFile="rear.aspx.cs" Inherits="temp" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
<div class="container-fluid" style="text-align: center">
    <h1>Главная страница</h1>
    <p>Привет, Мир!</p>
    </div>
</asp:Content>
```

Таким образом, все страницы сайта, подчиненные элементу Master Page, содержат свойство MasterPageFile с указанием нужного мастера страниц. Содержимое «дочерней» страницы должно быть заключено в элемент-тэг <asp:Content>, в свойствах которого указан ContentPlaceHolderID со значением ContentPlaceHolder1, которое установлено в блоке content XAML-верстки Master Page.

Подключение БД через файл конфигурации

Формулировка задачи. Настроить грамотно соединение web-приложения с БД с помощью файла конфигурации.

Уровень сложности: простой.

Решение

Достаточно в файле конфигурации проекта воспользоваться разделом connectionStrings и прописать следующий код (листинг 5):

Листинг 5. Файл конфигурации проекта с подключением к БД

```
<connectionStrings>
    <clear />
    <add connectionString="Data Source=.\sqlexpress; Initial Catalog=database1; Integrated Security=True;" name="qwe" providerName="System.Data.SqlClient" />
</connectionStrings>
```

В строке подключения, как правило, указывается имя сервера (свойство Data Source), название БД (свойство Initial Catalog), имя провайдера, а также настройки безопасности.

В случае, если среда разработки MS Visual Studio установлен некорректно, при установлении соединения с БД средствами интерфейса IDE могут возникать непредвиденные исключения.

Организация аутентификации пользователя (на базовом уровне)

Формулировка задачи. Необходимо организовать аутентификацию пользователя в web-приложении на базовом уровне.

Теоретическая справка. Организация аутентификации пользователя осуществляется на основе SessionState в несколько этапов – это создание страницы для аутентификации пользователя Login.aspx, использование файла Global.asax для работы на глобальном уровне проекта (в основном в части ролей), а также настройка конфигурации проекта в целом. Однако для осуществления управления уникальными идентификаторами сеанса в процессе работы с SessionState используется класс SessionIDManager. Считается хорошим тоном самостоятельно написать данный класс, переопределив методы CreateSessionID и Validate.

Уровень сложности: средний.

Решение

Данная задача включает в себя несколько аспектов. Прежде всего, необходимо выполнить настройки файла конфигурации (листинг 6):

Листинг 6. Настройка файла конфигурации

```
  <appSettings>
    <add key="ValidationSettings:UnobtrusiveValidationMode" value="None"/>
  </appSettings>
  <location>
    <system.web>
      <sessionState mode = "InProc" timeout = "30" sessionIDManagerType = "Samples.AspNet.Session.MySessionIDManager">
      </sessionState>
      <authentication mode="Forms">
        <forms loginUrl="~/login.aspx" />
      </authentication>
      <compilation debug="true" targetFramework="4.5.2">
        <assemblies>
          <add assembly="System.Security, Version=4.0.0.0, Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
        </assemblies>
      </compilation>
      <httpRuntime maxRequestLength="1048576" targetFramework="4.5.2"/>
      <identity impersonate="false"/>
      <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"/>
    </system.web>
  </location>
```

Основные моменты настройки файла конфигурации следующие:

— ValidationSettings: UnobtrusiveValidationMode – указывает, как глобально ASP.NET позволяет встроенным элементам проверки использовать ненавязчивые скрипты на стороне клиента. Значение None означает, что приложение будет использовать поведение pre-4.5 (встроенные скрипты на страницах).

— Настройки сессии указаны в подразделе sessionState – таймаут сессии составляет 30 минут, данные сессии будут храниться в оперативной памяти, а также для управления идентификатором сеанса прилагается дополнительный файл кода MySessionIDManager (листинг 7), расположенный в директории App_Code.

В подразделе authentication файла конфигурации выбран вариант аутентификации с помощью форм (mode="Forms"), а также указан путь к странице авторизации пользователя.

Листинг 7. Содержимое файла MySessionIDManager.cs

```
using System;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Web.SessionState;

namespace Samples.AspNet.Session
{
    public class MySessionIDManager : SessionIDManager
    {
        public string genToken()
        {
            var ua = HttpContext.Current.Request.Headers["User-Agent"].ToString();
            var ip = HttpContext.Current.Request.UserHostAddress;
            byte[] key = Encoding.UTF8.GetBytes("Что-то секретное");
            HMACSHA1 qq = new HMACSHA1(key); //применение алгоритма необратимого шифрования
            byte[] outqq = qq.ComputeHash(Encoding.UTF8.GetBytes(ua + ip));
            var token = new StringBuilder();
            foreach (byte b in outqq)
                token.AppendFormat("{0:x2}", b);
            return token.ToString();
        }

        public override string CreateSessionID(HttpContext context)
        {
            return genToken() + Guid.NewGuid().ToString();
        }

        public override bool Validate(string clientToken)
        {
            try
            {
                var testToken = genToken();
                var testGuid = new Guid(clientToken.Substring(40));
                if (clientToken == testToken + testGuid.ToString())
                    return true;
                else
                    return false;
            }
            catch { }
            return false;
        }
    }
}
```

Программный код аутентификации является довольно простым для понимания и представлен ниже (листинг 8).

Листинг 8. Программный код аутентификации пользователя

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
    {
        Page.Validate();
        if (!Page.IsValid) return;
        if (Page.IsValid)
        {
            string username = Login1.UserName;
            string pwd = Login1.Password;
            string s;
            byte[] key = Encoding.UTF8.GetBytes("Что-то секретное");
            HMACSHA1 qq = new HMACSHA1(key); //применение алгоритма необратимого шифрования
            byte[] outqq = qq.ComputeHash(Encoding.UTF8.GetBytes(pwd));
            var pass = new StringBuilder();
            foreach (byte b in outqq)
                pass.AppendFormat("{0:x2}", b);
            s = ConfigurationManager.ConnectionStrings["qwe"].ConnectionString;
            SqlConnection con = new SqlConnection(s); //соединение с БД
            con.Open();
            string sqlUserName;
            sqlUserName = "select id from users where login ='" + username + "' and pass ='" +
pass.ToString() + "'";
            SqlCommand cmd = new SqlCommand(sqlUserName, con);
            cmd.CommandType = CommandType.Text;
            try
            {
                //сохранение идентификатора пользователя в сессию
                Session.Add("UserAuthentication", cmd.ExecuteScalar().ToString());
                var SQLQuery = "select role from users where id = " +
Session["UserAuthentication"].ToString();
                var comm = new SqlCommand(SQLQuery, con);
                var roleId = comm.ExecuteScalar().ToString();
                string rulesStr;
                switch (roleId)
                {
                    case "1":
                        rulesStr = "admin";
                        break;
                    case "2":
                        rulesStr = "professor";
                        break;
                    default:
                        rulesStr = "teacher";
                        break;
                }
                e.Authenticated = true;
            }
            catch
            {
                con.Close();
                Session["UserAuthentication"] = null;
                e.Authenticated = false;
            }
            con.Close();
        }
    }
```

Также необходимо добавить в проект файл Global.asax и воспользоваться событием Application_AuthenticateRequest (листинг 9):

Листинг 9. Файл Global.asax

```
public void Application_AuthenticateRequest(Object sender, EventArgs e)
    {
        if (Context.User != null)
        {
            int rules;
            var con = new System.Data.SqlClient.SqlConnection(ConfigurationManager.
ConnectionStrings["qwe"].ConnectionString);
            con.Open();
            string sqlrules = "select role from users where login = @login";
            var cmd2 = new System.Data.SqlClient.SqlCommand(sqlrules, con);
            cmd2.Parameters.Clear();
            cmd2.Parameters.Add("@login", System.Data.SqlDbType.NVarChar);
            cmd2.Parameters["@login"].Value=HttpContext.Current.User.Identity.Name;
            if (String.IsNullOrEmpty(cmd2.ExecuteScalar().ToString()))
                rules = 3;
            else rules = int.Parse(cmd2.ExecuteScalar().ToString());
            string rulesStr;
            switch (rules)
                {
                    case 1:
                        rulesStr = "admin";
                        break;
                    case 2:
                        rulesStr = "professor";
                        break;
                    default:
                        rulesStr = "teacher";
                        break; }
            if (rules != 0)
            {
                Context.User = new System.Security.Principal.GenericPrincipal
(Context.User.Identity, new string[] {rulesStr} );
            }
        }
    }
```

Организация нескольких ролей в web-приложении с использованием разграничения прав доступа

Формулировка задачи. Необходимо организовать доступ к личному кабинету пользователя с категоризацией по ролям.

Теоретическая справка. Личный кабинет пользователя включает в себя набор нескольких страниц сайта с реализацией конкретного функционала. Следовательно, доступ к страницам сайта принято организовывать через файлы конфигурации с использованием поддиректорий проекта. Так, например, для организации личного кабинета администратора (соответственно роль admin в web-приложении) в проекте создается поддиректория admin с набором соответствующих страниц сайта и с собственным файлом конфигурации web.config, где будут указаны соответствующие настройки доступа.

Замечание. Данный функционал целесообразно реализовывать при наличии в проекте модуля аутентификации пользователя. Также следует предусмотреть принадлежность пользователя к конкретной роли на уровне БД проекта.

Уровень сложности: средний.

Решение

Для организации доступа к личному кабинету менеджера новостной ленты (например, роль newsman в проекте) необходимо создать в проекте поддиректорию admin_news с набором соответствующих страниц и собственным файлом конфигурации. Настройки корневого файла конфигурации будут иметь стандартный вид (листинг 10).

Листинг 10. Содержимое корневого web.config

```
  <appSettings>
    <add key="ValidationSettings:UnobtrusiveValidationMode" value="None"/>
  </appSettings>
  <location>
    <system.web>
      <sessionState mode="InProc" timeout="30" sessionIDManagerType =
"Samples.AspNet.Session. MySessionIDManager">
      </sessionState>
      <authentication mode="Forms">
        <forms loginUrl="~/login.aspx" />
      </authentication>
      <compilation debug="true" targetFramework="4.5.2">
        <assemblies>
          <add assembly="System.Security, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
        </assemblies>
      </compilation>
      <httpRuntime maxRequestLength="1048576" targetFramework="4.5.2"/>
      <identity impersonate="false"/>
      <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"/>
    </system.web>
  </location>
```

Структура файла конфигурации для роли менеджера новостной ленты будет значительно проще и лаконичней по сравнению с основным файлом конфигурации проекта (листинг 11).

Листинг 11. Содержимое web.config поддиректории newsman

```
  <location>
    <system.web>
      <authorization>
        <allow roles="newsman" />
        <deny users="?" />
      </authorization>
    </system.web>
  </location>
```

Таким образом, листинг web.config поддиректории newsman говорит о том, что доступ к набору страниц директории newsman открыт (allow) для пользователей с ролью newsman. Для незарегистрированных пользователей сайта доступ к набору страниц директории newsman закрыт (deny).

Организация модального окна для просмотра изображений на сайте

Формулировка задачи. Необходимо организовать работу модального окна для просмотра изображений с использованием CSS-стилей и JavaScript.

Уровень сложности: простой.

Решение

Для решения данной задачи необходимо организовать форму модульного окна в XAML-верстке (листинг 12), задействовав при этом стили bootstrap.

Листинг 12. XAML-верстка модального окна

```
<div id="myModal" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-body">
                    <img id="modall-image" style="width: 100%" />
                    <a class="left carousel-control left-btn">
                        <span class="glyphicon glyphicon-chevron-left"></span>
                        <span class="sr-only">Previous</span>
                    </a>
                    <a class="right carousel-control right-btn">
                        <span class="glyphicon glyphicon-chevron-right"></span>
                        <span class="sr-only">Next</span>
                    </a>
                </div>
                <div class="modal-footer">
                    <div class="modal-caption" style="width: 100%; text-align:
justify;"></div>
                    <button type="button" class="btn btn-default" data-
dismiss="modal">Закрыть</button>
                </div>
            </div>
        </div>
</div>
```

Реализация функционала просмотра изображений в модальном окне имеет довольно простую структуру на языке JavaScript (листинг 13).

Листинг 13. JavaScript решения

```
<script>
        $(document).ready(function () {
            var currentImg;
            //при нажатию на любую кнопку, имеющую класс .btn
            $(".ImageToModal").click(function () {
                currentImg = this;
                //открыть модальное окно с id="myModal"
                $("#myModal").modal('show');
                $("#modall-image").attr("src", $(this).attr("src"));
                $(".modal-caption").text($(this).next().text());
            });
            $('.left-btn').click(function (e) {
                var leftImg = $(currentImg).parent().parent().prev().find('img');
                if (leftImg.length > 0) {
                    $("#modall-image").attr("src", leftImg.attr("src"));
                    $(".modal-caption").text(leftImg.next().text());
                    currentImg = leftImg;
                }
            });
            $('.right-btn').click(function (e) {
                var rightImg = $(currentImg).parent().parent().next().find('img');
                if (rightImg.length > 0) {
                    $("#modall-image").attr("src", rightImg.attr("src"));
                    $(".modal-caption").text(rightImg.next().text());
                    currentImg = rightImg;
                }
            });
            document.onkeydown = function (e) {
                if (e.keyCode === 27) { $("#myModal").modal('hide'); }
            };
        });
</script>
```

Вызов модального окна осуществляется следующим образом (листинг 14):

Листинг 14. Вызов класса модального окна

```
<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1">
        <ItemTemplate>
            <div class="editors_card ">
                <figure>
                    <img src='<%# nullcheck(Eval("fname")) %>' id='<%# Eval("id") %>'
class="ImageToModal" />
                    <figcaption style="white-space: nowrap; overflow: hidden; text-overflow:
ellipsis;"><%# Eval("caption") %></figcaption>
                </figure>
            </div>
        </ItemTemplate>
</asp:ListView>
```

Использование модального окна в проектах ASP.NET (*Рисунок 7*) повышает функциональность сайта.

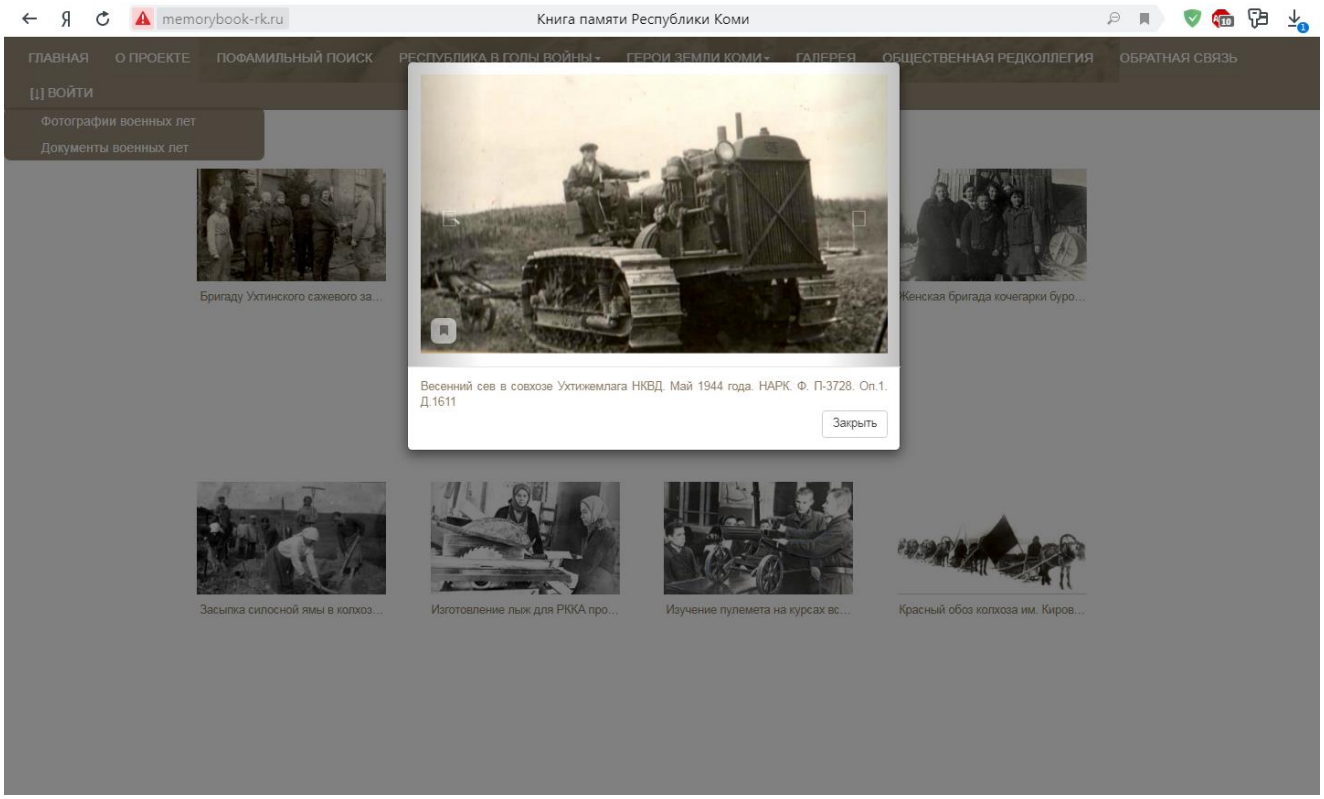


Рисунок 7. Пример использования модального окна

Организация новостной ленты (для клиента)

Формулировка задачи. Необходимо организовать новостную ленту на сайте с использованием блочной верстки для удобного просмотра пользователями (*Рисунок 8*).

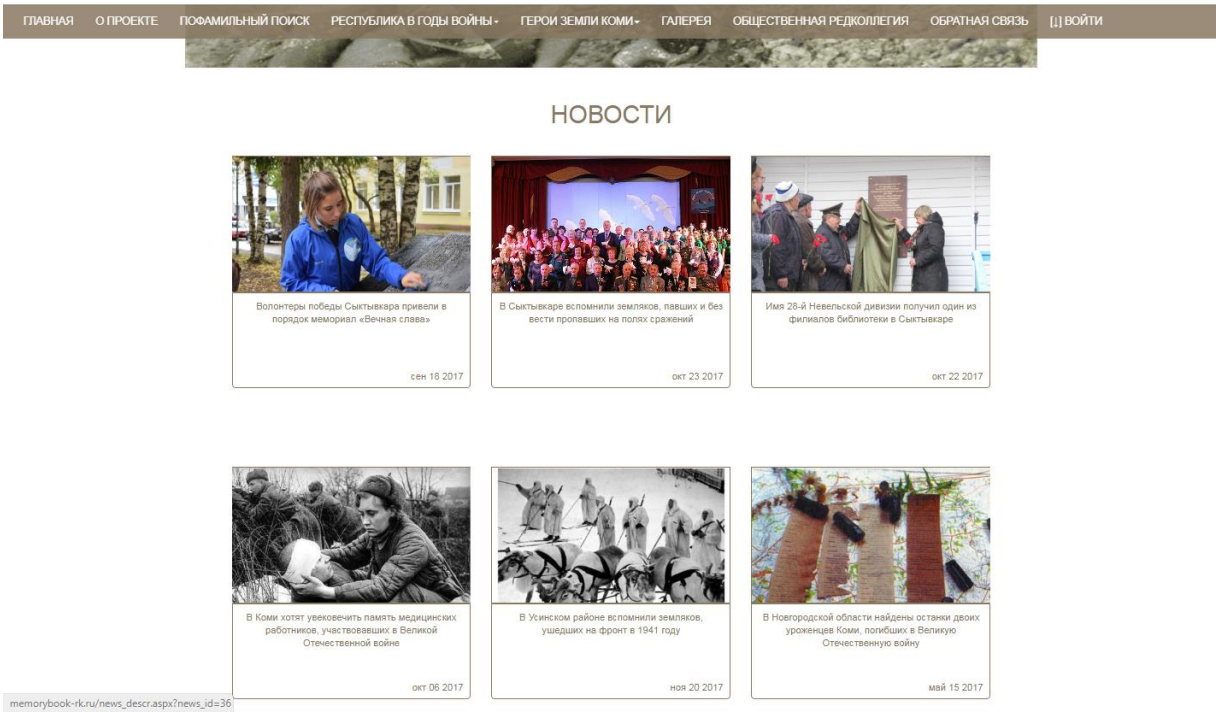


Рисунок 8. Пример новостной ленты

Уровень сложности: простой.

Решение

Для организации новостной ленты на сайте необходимо создать одну или несколько сущностей (на усмотрение разработчика) на уровне БД, а именно (сущность «news»):

- Id (int) – идентификатор записи, первичный ключ.
- Title (nvarchar(max)) – заголовок новости.
- Sphere (int) – область, к которой относится новость, внешний ключ.
- Datee (datetime) – дата размещения.
- Popular (bit) – является ли популярной для сайта.
- Short_new (nvarchar(max)) – сокращенный вариант текста новости.
- Filee (nvarchar(max)) – полный текст новости.
- Source (nvarchar(max)) – первоисточник.
- Image (varbinary(max)) – изображение к заголовку.
- Rubrics (int) – рубрика, внешний ключ (необязательное поле).

Добавив несколько тестовых записей в таблицу news, можно с помощью элемента управления SQLDataSource организовать статичный запрос к БД (листинг 15).

Листинг 15. XAML-верстка фрагмента страницы

```
<div style="width: 60%; float: left;">
    <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString = "<%$
ConnectionStrings:qwe %>"
        SelectCommand="select news.id, news.title, convert(VARCHAR(10), news.datee, 104)
as datee, news.short_new, news_rubrics.caption, news.source from news inner join
news_rubrics on news.rubrics = news_rubrics.id where news.image != 0x order by news.datee
desc">
    </asp:SqlDataSource>
```

Элемент управления ListView позволяет использовать шаблоны для организации нестандартного вида новостной ленты (листинг) при подключении к БД через SqlDataSource.

Шаблоны элемента управления ListView можно редактировать, подключив предварительно к компоненту источник данных. Данный элемент в части функционала достаточно широк (поддерживает автоматическое добавление, удаление и редактирование данных, а также реализует группировку и сортировку сведений с возможностью постраничного разбиения), имеет достаточный перечень шаблонов по умолчанию.

Таким образом, целесообразно добавить постраничное разбиение содержимого, а также группировку данных по нескольким атрибутам для удобства просмотра информации пользователями. Вывод необходимой информации осуществляется в блоке ItemTemplate (листинг 16).

Листинг 16. XAML-верстка фрагмента страницы

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:ListView ID="ListView1" runat="server" AllowPaging="True" AutoGenerateRows
="False" DataKeyNames="id" GroupPlaceholderID="groupPlaceHolder1"
DataSourceID="SqlDataSource1" ItemPlaceholderID="itemPlaceHolder1">
            <EmptyDataTemplate>
                <span>Новости отсутствуют.</span>
            </EmptyDataTemplate>
            <LayoutTemplate>
                <asp:PlaceHolder runat="server" ID="groupPlaceHolder1"> </asp:PlaceHolder>
                <asp:DataPager ID="DataPager1" runat="server" PagedControlID = "ListView1"
PageSize="4">
                    <Fields>
                        <asp:NumericPagerField ButtonType="Link" NextPageText = "Вперед"
PreviousPageText = "Назад" />
                    </Fields>
                </asp:DataPager>
            </LayoutTemplate>
            <GroupTemplate>
                <asp:PlaceHolder runat="server" ID = "itemPlaceHolder1">
</asp:PlaceHolder>
            </GroupTemplate>
            <ItemTemplate>
                <label for="modal" data-newsid="<%# Eval("id") %>">
                    <div class="news-image-container">
                        <img src="images/<%# Eval("id") %>.jpg" />
                    </div>
                    <div style="padding: 20px;">
                        <span><%# Eval("datee") + " | " + Eval("caption") %> </span>
                        <br />
                        <span class="news-title"><%# Eval("title") %></span><br />
                        <span><%# Eval("short_new") %></span>
                        <span class="news-content"></span><br />
                    </div>
                </label>
            </ItemTemplate>
        </asp:ListView>
    </ContentTemplate>
</asp:UpdatePanel>
</div>
```

Просмотр новости можно осуществить с использованием JavaScript в модальном окне или с помощью Get-запроса в новой вкладке браузера.

Организация модуля ведения новостной ленты в панели администратора

Формулировка задачи. Необходимо разработать модуль для ведения контент-менеджером новостной ленты в панели администратора. В частности, разработаем модуль для добавления новостей на сайт.

Уровень сложности: средний.

Решение

Верстка страницы будет иметь простой вид (*Рисунок 9*):

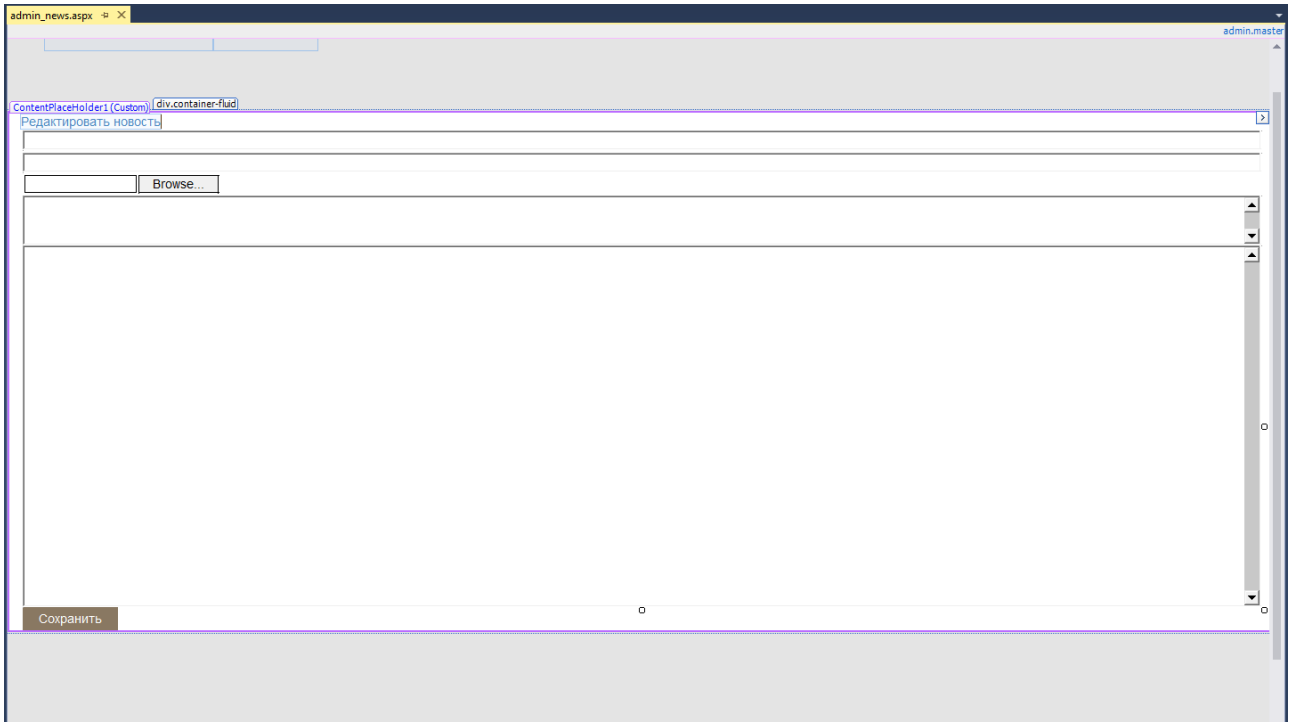


Рисунок 9. Структура страницы

XAML-верстка при будет выглядеть следующим образом (листинг 17):

Листинг 17. XAML-верстка страницы

```
    <div class="container-fluid" style="margin-top:20px; margin-bottom:90px;
display:block;">
        <a href="admin_editnews.aspx">Редактировать новость</a>
        <asp:TextBox ID="title" runat="server" Width="100%" placeholder="Заголовок"
            style="margin-bottom:3px;" />
        <asp:TextBox ID="source" runat="server" Width="100%" placeholder="Источник"
            style="margin-bottom:3px;" />
        <asp:FileUpload ID="FileUpload1" runat="server" Width="100%" BorderStyle="Solid"
            ForeColor="Black" style="margin-bottom:3px;" />
        <asp:TextBox ID="text_short" runat="server" TextMode="MultiLine" Width="100%"
            Height="50px" placeholder="Сокращенный вариант новости" />
        <asp:TextBox ID="text_new" runat="server" Height="400px" TextMode="MultiLine"
            Width="100%" Wrap="true" ClientIDMode="Static" ValidateRequestMode="Disabled"
/>
        <asp:Button ID="Button1" runat="server" Text="Сохранить" CssClass="content-button"
            OnClick="Button1_Click" />
    </div>
```

Используя строку подключения к БД, приложение обращается к серверу БД с запросом типа Insert для добавления записи в таблицу news с возможностью сохранения изображения (бинарное чтение данных из потока), формируя запрос с параметрами (листинг 18).

Листинг 18. Событие Click кнопки Button1

```
protected void Button1_Click(object sender, EventArgs e)
    {
        using (SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings ["qwe"].ConnectionString))
        {
            string s = "insert into news (title, datee, text_short, text_new, source,
image)" + "values (@title, @datee, @text_short, @text_new, @source, @image)";
            SqlDataAdapter adapter = new SqlDataAdapter();
            //организация запроса с параметром
            SqlCommand command = new SqlCommand(s, con);
            command.Parameters.Clear();
            command.Parameters.Add("@title", SqlDbType.NVarChar);
            command.Parameters["@title"].Value = title.Text;
            command.Parameters.Add("@datee", SqlDbType.DateTime);
            command.Parameters["@datee"].Value = DateTime.Now;
            command.Parameters.Add("@text_short", SqlDbType.NVarChar);
            command.Parameters["@text_short"].Value = text_short.Text;
            command.Parameters.Add("@text_new", SqlDbType.Text);
            command.Parameters["@text_new"].Value = text_new.Text;
            command.CommandType = CommandType.Text;
            command.Parameters.Add("@source", SqlDbType.NVarChar);
            command.Parameters["@source"].Value = source.Text;
            command.CommandType = CommandType.Text;
            if (FileUpload1.HasFile)
            {
                HttpPostedFile postedfile = FileUpload1.PostedFile;
                string filename = Path.GetFileName(postedfile.FileName);
                string fileextension = Path.GetExtension(filename);
                int filesize = postedfile.ContentLength;
                if (fileextension.ToLower() == ".jpg")
                {
                    //использование потоков для загрузки файла на сервер
                    Stream stream = postedfile.InputStream;
                    BinaryReader br = new BinaryReader(stream);
                    byte[] bytes = br.ReadBytes((int)stream.Length);
                    command.Parameters.Add("@image", SqlDbType.VarBinary);
                    command.Parameters["@image"].Value = bytes;
                    command.CommandType = CommandType.Text;
                }
            }
            con.Open();
            command.ExecuteNonQuery();
            title.Text = "";
            source.Text = "";
            text_short.Text = "";
            text_new.Text = "";
        }
    }
```

По завершении программного кода данные формы очищаются для ввода пользователем новой информации.

Отображение данных в виде классической таблицы (GridView в применении с SQLDataSource)

Формулировка задачи. Необходимо отобразить данные поиска в виде классической таблицы данных (с использованием элемента SqlDataSource).

Уровень сложности: простой.

Решение

Для начала необходимо подключиться к источнику данных (листинг 19) с возможностью использования фильтров для поиска необходимой информации.

Листинг 19. XAML-верстка, подключение к источнику данных

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:qwe %>"
            SelectCommand="select * from [mbook] order by [mfamily], [mname],
[mmiddlename]" FilterExpression="mfamily like '%{0}%'" OnFiltering =
"SqlDataSource1_Filtering">
            <FilterParameters>
                <asp:ControlParameter ControlID="TextBox1" Type="String"
PropertyName="Text" />
            </FilterParameters>
</asp:SqlDataSource>
```

Затем необходимо правильно подключить элемент управления GridView, указав обязательные свойства DataSourceID (идентификатор SqlDataSource) и DataKeyNames (ключевые поля сущности). В блоке Columns указываются все необходимые для отображения поля с применением стилей (собственных или заимствованных). Важным является поле CommandField, которое позволяет подключать различные дополнительные возможности компонента, например, выбор нужной записи (листинг 20).

Листинг 20. XAML-верстка, настройка GridView

```
<asp:GridView ID="GridView1" runat="server" AllowPaging="True" DataSourceID=
"SqlDataSource1" DataKeyNames="dbid" PageSize="15" AllowSorting="True" Width="75%"
OnPageIndexChanging = "GridView1_PageIndexChanging" EnablePersistedSelection="True"
AutoGenerateColumns="False" AutoGenerateRows="False"
OnSelectedIndexChanged="GridView1_SelectedIndexChanged" >
    <Columns>
        <asp:TemplateField HeaderText="№ пп">
            <ItemTemplate><%# Convert.ToString( Container.DataItemIndex + 1 ) %>
</ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="mfamily" HeaderText="Фамилия">
        </asp:BoundField>
        <asp:BoundField DataField="mname" HeaderText="Имя">
        </asp:BoundField>
        <asp:BoundField DataField="mmiddlename" HeaderText="Отчество">
        </asp:BoundField>
        <asp:BoundField DataField="mbirthday" HeaderText="Дата рождения" HeaderStyle-
Wrap="true">
        </asp:BoundField>
        <asp:BoundField DataField="mbirthplace" HeaderText="Место рождения" HeaderStyle-
Wrap="true">
        </asp:BoundField>
        <asp:BoundField DataField="mdraftplace" HeaderText="Место призыва" HeaderStyle-
Wrap="true">
        </asp:BoundField>
        <asp:BoundField DataField="mdatedeath" HeaderText="Дата смерти" HeaderStyle-
Wrap="true">
        </asp:BoundField>
        <asp:CommandField CausesValidation="False" HeaderText="Карточка"
SelectText="просмотр" ShowHeader="True" ShowSelectButton="True">
        </asp:CommandField>
    </Columns>
    <PagerSettings Mode="NumericFirstLast" />
</asp:GridView>
```

Стили можно указывать как для элемента в целом, так и по отдельности с использованием классов. Если содержимое таблицы достаточно объемно, можно воспользоваться возможностью постраничного разбиения данных (свойства AllowPaging и PageSize).

Событие GridView1_PageIndexChanging (листинг 21) необходимо для организации процесса пролистывания пользователем GridView. Однако если объем информации несколько шире, чем перечень полей в таблице, то при выборе конкретной записи (событие GridView1_SelectedIndexChanged) можно переадресовать пользователя на стороннюю страницу с более подробоной информацией за счет применения GET-запроса.

Листинг 21. Программный код GridView

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        GridView1.PageIndex = e.NewPageIndex;
    }

    public void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
        Server.Transfer("~/people_descr.aspx?people_id=" + people_id);
    }
```

Использование элемента DetailsView для работы с данными (в применении с SQLDataSource)

Формулировка задачи. Необходимо организовать просмотр и редактирование персональных сведений в личном кабинете пользователя (*Рисунок 10*).

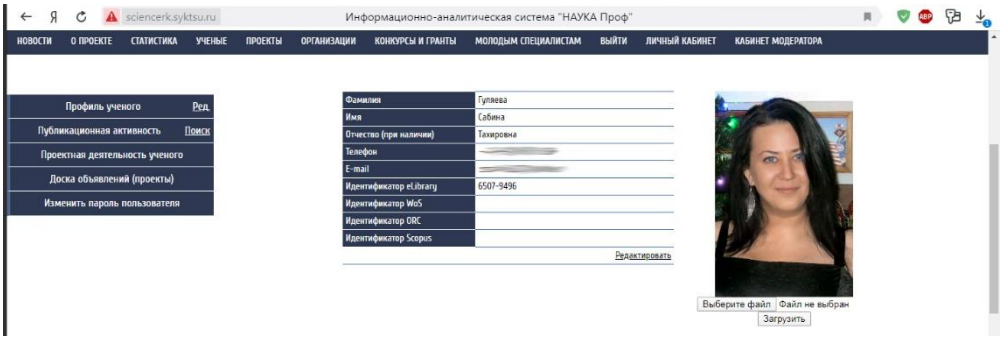


Рисунок 10. Учет персональных данных пользователя, организованный через DetailsView

Уровень сложности: средний.

Решение

Для решения данной задачи не придется использовать программный код, поскольку элемент управления DetailsView поддерживает режим автоматического добавления, изменения и удаления информации, а также позволяет просматривать одну запись за раз.

Подключение к источнику данных целесообразно организовать с использованием компонента SqlDataSource (листинг 22).

Листинг 22. Настройка SqlDataSource

```
<asp:SqlDataSource ID="sci_common" runat="server" ConnectionString="<%$ ConnectionStrings:qwe
%>" OldValuesParameterFormatString="original_{0}" ProviderName="System.Data.SqlClient"
SelectCommand="SELECT [fam], [im], [otch], [telephone], [email], [spin], [researcherid],
[orcid], [scopusid] FROM [scientist] WHERE ([id] = @id)" UpdateCommand="UPDATE scientist SET
fam = @fam, im = @im, otch = @otch, telephone = @telephone, email = @email, spin = @spin,
researcherid = @researcherid, orcid = @orcid, scopusid = @scopusid WHERE (id = @id)">
    <SelectParameters>
        <asp:SessionParameter SessionField="UserAuthentication" Name="id" DefaultValue="0"
Type="Int32" />
    </SelectParameters>
    <UpdateParameters>
        <asp:Parameter Name="fam" />
        <asp:Parameter Name="im" />
        <asp:Parameter Name="otch" />
        <asp:Parameter Name="telephone" />
        <asp:Parameter Name="email" />
        <asp:Parameter Name="spin" />
        <asp:Parameter Name="researcherid" />
        <asp:Parameter Name="orcid" />
        <asp:Parameter Name="scopusid" />
        <asp:SessionParameter SessionField="UserAuthentication" Name="id" DefaultValue="0"
Type="Int32" />
    </UpdateParameters>
</asp:SqlDataSource>
```

Поскольку персональные данные пользователя могут не только просматриваться, но и изменяться, то необходимо помимо SelectCommand использовать также UpdateCommand со списком параметров стандартного типа <asp:Parameter>, кроме первичного ключа таблицы – значение этого параметра хранится в глобальном контейнере, откуда и необходимо его считывать. Таким образом, подключение к БД позволяет за счет настройки SqlDataSource просматривать пользовательские данные и видоизменять их. Настройка компонента DetailsView напоминает отдаленно настройку элемента GridView, главным свойством которого является DataSourceID. Таким образом, верстка элемента DetailsView будет иметь следующий вид (листинг 23).

Листинг 23. XAML-верстка таблицы с персональными данными

```
<asp:DetailsView ID="DetailsView1" runat="server" Width="60%" AutoGenerateRows="False"
DataSourceID="sci_common" HorizontalAlign="Center" AllowPaging="True">
    <Fields>
        <asp:BoundField DataField="fam" HeaderText="Фамилия" ReadOnly="false">
        </asp:BoundField>
        <asp:BoundField DataField="im" HeaderText="Имя" ReadOnly="false">
        </asp:BoundField>
        <asp:BoundField DataField="otch" HeaderText="Отчество" ReadOnly="false">
        </asp:BoundField>
        <asp:BoundField DataField="email" HeaderText="E-mail" ReadOnly="false">
        </asp:BoundField>
        <asp:BoundField DataField="spin" HeaderText="eLibrary" ReadOnly="false">
        </asp:BoundField>
        <asp:BoundField DataField="scopusid" HeaderText="Scopus" ReadOnly="false">
        </asp:BoundField>
        <asp:CommandField ShowEditButton="True" EditText="Редактировать"
UpdateText="Сохранить" CancelText="Отменить">
        </asp:CommandField>
    </Fields>
    <PagerSettings PageButtonCount="5" />
    <PagerStyle BackColor="White" ForeColor="Black" HorizontalAlign="Right" />
</asp:DetailsView>
```

Основной тип полей – это BoundField. В свойстве DataField указывается название атрибута таблицы, в свойстве HeaderText – заголовок поля, отображаемый для пользователя. Также можно воспользоваться тэгом HeaderStyle для организации стилей. Подключение автоматического режима редактирования данных осуществляется в поле CommandField (свойства ShowEditButton, EditText, UpdateText, CancelText). Если компонент настроен верно, при попытке отредактировать данные элемент управления автоматически перейдет в режим редактирования и сохранит введенные пользователем значения (*Рисунок 11*).

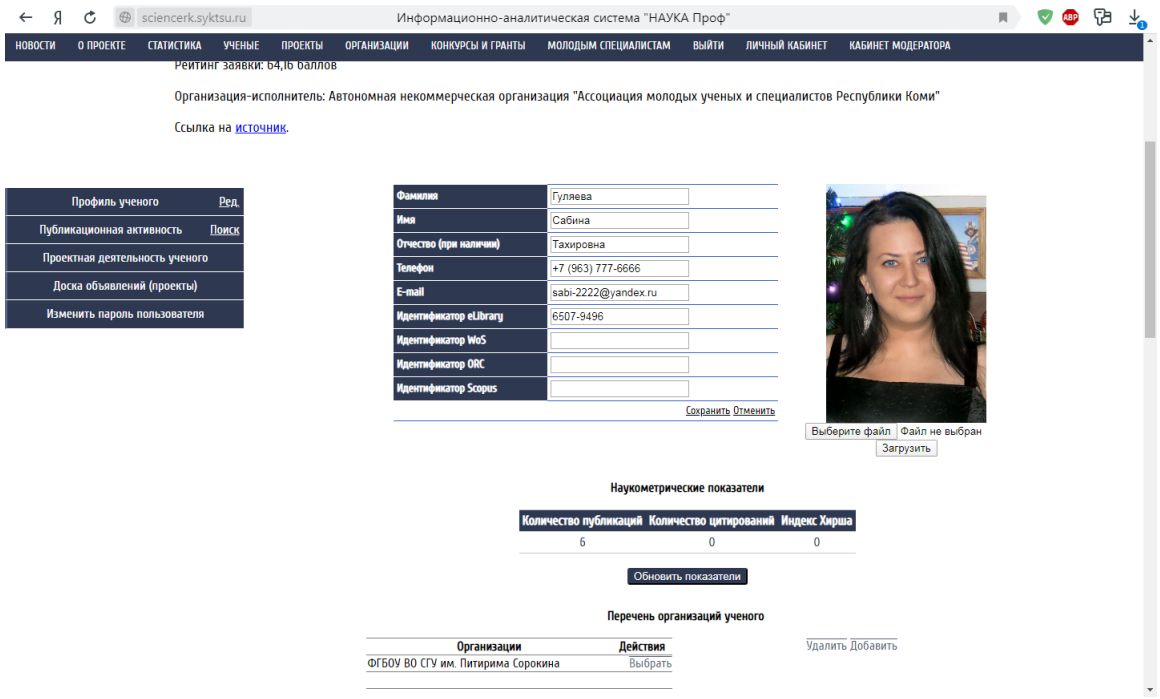


Рисунок 11. Режим редактирования данных элемента DetailsView

Использование элемента FormView для просмотра данных по персоналиям (в применении с SQLDataSource)

Формулировка задачи. Необходимо организовать просмотр данных поиска по персоналиям на базовом уровне.

Уровень сложности: простой.

Решение

Для решения задачи необходимо организовать подключение к источнику данных на примере компонента SQLDataSource (листинг 24).

Листинг 24. XAML-верстка, подключение к источнику данных

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:qwe %>" SelectCommand="SELECT * FROM [mbook] WHERE ([dbid] = @dbid)">
            <SelectParameters>
                <asp:QueryStringParameter QueryStringField="people_id" Name="dbid" />
            </SelectParameters>
</asp:SqlDataSource>
```

Далее необходимо настроить элемент управления FormView, задав при этом собственный стиль отображения сведений (листинг 25).

Листинг 25. Настройка FormView

```
<asp:FormView ID="FormView1" runat="server" CellPadding="4" DataKeyNames="dbid"
DataSourceID="SqlDataSource1" HorizontalAlign="Center" Width="70%">
            <ItemTemplate>
                <div style="padding: 50px;">
                    //использование кода в верстке для проверки данных на пустоту
                    <p class="text-justify"><asp:Label ID="Label1" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mfamily") != DBNull.Value &&
Eval("mfamily").ToString().Trim() != "")? "<strong>Фамилия:</strong> " + Eval("mfamily") :
"" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label2" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mname") != DBNull.Value &&
Eval("mname").ToString().Trim() != "")? "<strong>Имя:</strong> " + Eval("mname") : "" %>'
Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label3" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mmiddlename") != DBNull.Value &&
Eval("mmiddlename").ToString().Trim() != "")? "<strong>Отчество:</strong> " +
Eval("mmiddlename") : "" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label6" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mbirthday") != DBNull.Value &&
Eval("mbirthday").ToString().Trim() != "")? "<strong>Дата рождения:</strong> " +
Eval("mbirthday") : "" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label7" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mbirthplace") != DBNull.Value &&
Eval("mbirthplace").ToString().Trim() != "")? "<strong>Место рождения:</strong> " +
Eval("mbirthplace") : "" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label8" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mdraftplace") != DBNull.Value &&
Eval("mdraftplace").ToString().Trim() != "")? "<strong>Место призыва:</strong> " +
Eval("mdraftplace") : "" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label9" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mlastplace") != DBNull.Value &&
Eval("mlastplace").ToString().Trim() != "")? "<strong>Последнее отмеченное место:</strong>
" + Eval("mlastplace") : "" %>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label10" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mrank") != DBNull.Value &&
Eval("mrank").ToString().Trim() != "")? "<strong>Звание:</strong> " + Eval("mrank") : ""
%>' Font-Size="16px"></asp:Label></p>
                    <p class="text-justify"><asp:Label ID="Label13" runat="server"
style="color: #897862;"
                        Text='<%# (Eval("mdatedeath") != DBNull.Value &&
Eval("mdatedeath").ToString().Trim() != "")? "<strong>Дата смерти:</strong> " +
Eval("mdatedeath") : "" %>' Font-Size="16px"></asp:Label></p>
                </div>
            </ItemTemplate>
            <EmptyDataTemplate>
                <span>По запросу ничего не найдено.</span>
            </EmptyDataTemplate>
</asp:FormView>
```

Как видно из листинга, особый интерес представляет собой повторяющаяся для каждого поля данных альтернативная конструкция условного оператора. Рассмотрим ее на примере атрибута *Дата рождения* (листинг 26).

Листинг 26. Фрагмент XAML-верстки для настройки поля *Дата рождения* элемента FormView

```
<p class="text-justify"><asp:Label ID="Label6" runat="server" style="color: #897862;"
Text='<%# (Eval("mbirthday") != DBNull.Value && Eval("mbirthday").ToString().Trim() !=
"")? "<strong>Дата рождения:</strong> " + Eval("mbirthday") : "" %>' Font-
Size="16px"></asp:Label></p>
```

Если значение атрибута *Дата рождения* из БД для конкретной записи не равно нулю и не является пустой строкой, то данные будут выведены на экран. В противном случае данные не отобразятся.

Если все поля компонента настроены верно, то по результатам поиска сведений по конкретному лицу появится карточка сведений (*Рисунок 12*).

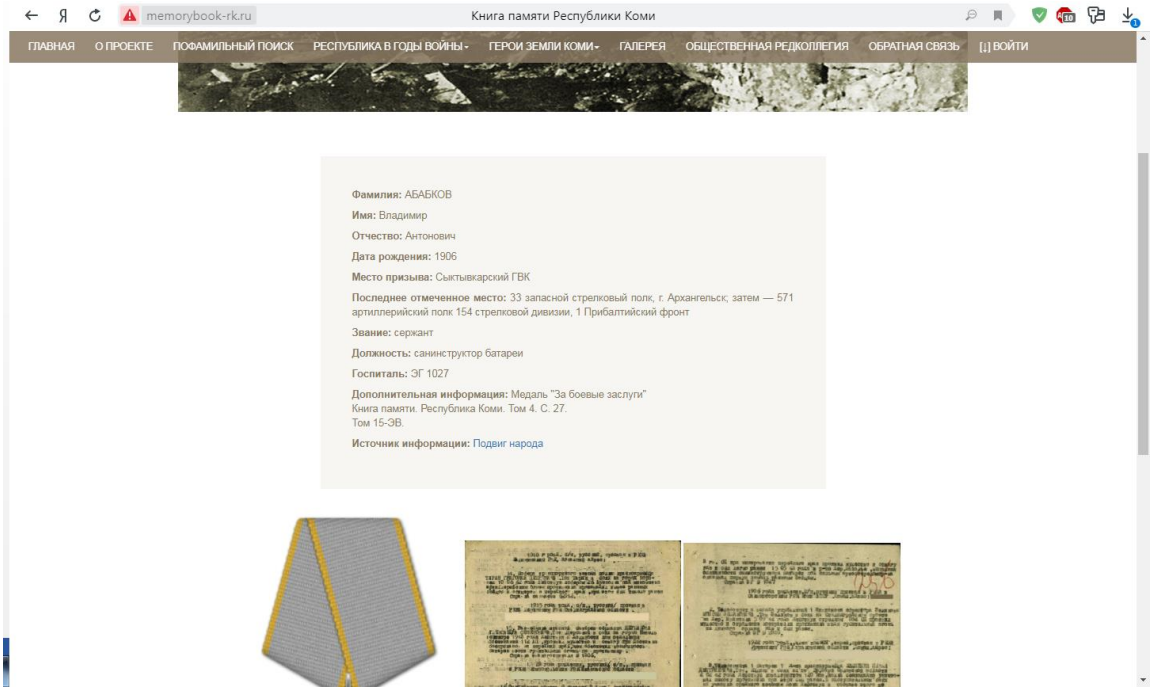


Рисунок 12. Карточка сведений по солдату

Организация модуля для обработки большого массива данных основных сущностей БД с использованием нескольких элементов привязки к данным (GridView, FormView / DetailsView, ListView) в панели администратора

Формулировка задачи. Необходимо реализовать модуль поэтапной обработки большого массива данных основных сущностей БД, а именно:

- Выбор нужной записи (по 2 – 4 атрибутам) в таблице значений (элемент GridView).
- Редактирование данных конкретной записи с использованием удобного шаблона (элемент FormView).
- Прикрепление нескольких дополнительных файлов к конкретной записи с использованием удобного контейнера с функцией постраничного разбиения (элемент ListView).

Уровень сложности: продвинутый.

Решение

Настройка элемента управления GridView является самой простой и менее объемной по сравнению с остальными, включает в себя перечень используемых ранее свойств (листинг 27).

Листинг 27. Настройка GridView и его источника данных

```
Настройка источника данных
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:qwe %>" SelectCommand="select * from [mbook] order by [mfamily], [mname],
[mmiddlename]" FilterExpression="mfamily like '%{0}%'" OnFiltering="SqlDataSource1_
Filtering">
    <FilterParameters>
        <asp:ControlParameter ControlID="TextBox1" Type="String" PropertyName= "Text" />
    </FilterParameters>
</asp:SqlDataSource>

Настройка компонента GridView
<asp:GridView ID="GridView1" runat="server" AllowPaging="True" DataSourceID=
"SqlDataSource1" DataKeyNames="dbid" PageSize="15" AllowSorting="True" CellSpacing="3"
OnPageIndexChanging="GridView1_PageIndexChanging" AutoGenerateColumns="False" AutoGenerate
Rows="False" OnSelectedIndexChanged="GridView1_SelectedIndexChanged">
    <Columns>
        <asp:TemplateField HeaderText="№ пп">
            //пример создания поля-счетчика для количества значений в выборке
            <ItemTemplate>
                <%# Convert.ToString(Container.DataItemIndex + 1) %>
            </ItemTemplate>
        </asp:TemplateField>
        <asp:BoundField DataField="dbid" HeaderText="dbid" Visible="false" />
        <asp:BoundField DataField="mid" HeaderText="mid">
        </asp:BoundField>
        <asp:BoundField DataField="mfamily" HeaderText="Фамилия">
        </asp:BoundField>
        <asp:BoundField DataField="mname" HeaderText="Имя">
        </asp:BoundField>
        <asp:BoundField DataField="mmiddlename" HeaderText="Отчество">
        </asp:BoundField>
        <asp:BoundField DataField="mbirthday" HeaderText="Дата рождения" >
        </asp:BoundField>
        <asp:BoundField DataField="mbirthplace" HeaderText="Место рождения" >
        </asp:BoundField>
        <asp:BoundField DataField="mdraftplace" HeaderText="Место призыва" >
        </asp:BoundField>
        <asp:BoundField DataField="mdatedeath" HeaderText="Дата смерти" >
        </asp:BoundField>
        <asp:CommandField CausesValidation="False" HeaderText="Карточка" SelectText=
"Выбрать" ShowHeader="True" ShowSelectButton="True">
        </asp:CommandField>
    </Columns>
    <PagerSettings Mode="NumericFirstLast" />
    <PagerStyle HorizontalAlign="Center" Height="30px" Width="30px" />
</asp:GridView>
```

При выборе конкретной записи для внесения изменений потребуется запомнить идентификатор записи и организовать карточку для внесения изменений. Следовательно, нужно настроить источник данных SqlDataSource с использованием команд SelectCommand, InsertCommand и UpdateCommand, после чего подключить к нему элемент FormView для удобной работы с данными (листинг 28).

Листинг 28. Настройка SqlDataSource для работы с данными выбранной записи

```
<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$
ConnectionStrings:qwe %>"
      InsertCommand="INSERT INTO [mbook] ([mid], [mentity], [mheader], [mfamily],
[mname], [mmiddlename], [mbirthday], [mbirthplace], [mdraftplace], [mlastplace], [mrank],
[mpost], [mdestiny], [mrelative], [maddinfo], [msource], [verify], [heroes_order],
[heroes_soc], [pop_av], [pop_ak], [pop_zhp], [pop_pr], [pop_sc], [pop_hs]) VALUES (@mid,
@mentity, @mheader, @mfamily, @mname, @mmiddlename, @mbirthday, @mbirthplace,
@mdraftplace, @mlastplace, @mrank, @mpost, @mdestiny, @mrelative, @maddinfo, @msource,
@verify, @heroes_order, @heroes_soc)"
      SelectCommand="SELECT * FROM [mbook] WHERE ([dbid] = @dbid)"
      UpdateCommand="UPDATE [mbook] SET [mid] = @mid, [mentity] = @mentity, [mheader] =
@mheader, [mfamily] = @mfamily, [mname] = @mname, [mmiddlename] = @mmiddlename,
[mbirthday] = @mbirthday, [mbirthplace] = @mbirthplace, [mdraftplace] = @mdraftplace,
[mlastplace] = @mlastplace, [mrank] = @mrank, [mpost] = @mpost, [mdestiny] = @mdestiny,
[mrelative] = @mrelative, [maddinfo] = @maddinfo, [msource] = @msource, [verify] =
@verify, [heroes_order] = @heroes_order, [heroes_soc] = @heroes_soc WHERE [dbid] = @dbid"
>
            <InsertParameters>
                <asp:Parameter Name="mid" Type="String" />
                <asp:Parameter Name="mentity" Type="String" />
                <asp:Parameter Name="mheader" Type="String" />
                <asp:Parameter Name="mfamily" Type="String" />
                <asp:Parameter Name="mname" Type="String" />
                <asp:Parameter Name="mmiddlename" Type="String" />
                <asp:Parameter Name="mbirthday" Type="String" />
                …//далее перечисляются остальные параметры запроса
            </InsertParameters>
            <SelectParameters>
                <asp:ControlParameter ControlID="GridView1" Name="dbid"
PropertyName="SelectedValue" Type="Int32" />
            </SelectParameters>
            <UpdateParameters>
                <asp:ControlParameter ControlID="GridView1" Name="dbid"
PropertyName="SelectedValue" Type="Int32" />
                <asp:Parameter Name="mid" Type="String" />
                <asp:Parameter Name="mentity" Type="String" />
                <asp:Parameter Name="mheader" Type="String" />
                <asp:Parameter Name="mfamily" Type="String" />
                <asp:Parameter Name="mname" Type="String" />
                <asp:Parameter Name="mmiddlename" Type="String" />
                <asp:Parameter Name="mbirthday" Type="String" />
                …//далее перечисляются остальные параметры запроса
            </UpdateParameters>
        </asp:SqlDataSource>
```

Как видно, запросы SelectCommand, InsertCommand и UpdateCommand являются довольно объемными, поэтому в листинге представлен фрагмент кода. В таких случаях обычно удобно применять конфигуратор источника данных (*Рисунок 13*), с использованием которого XAML-верстка формируется автоматически с указанием всех возможных нюансов (выбор stored procedure / columns from a table or view, задание условий Where, сортировка, автоматическая генерация Insert + Update + Delete).

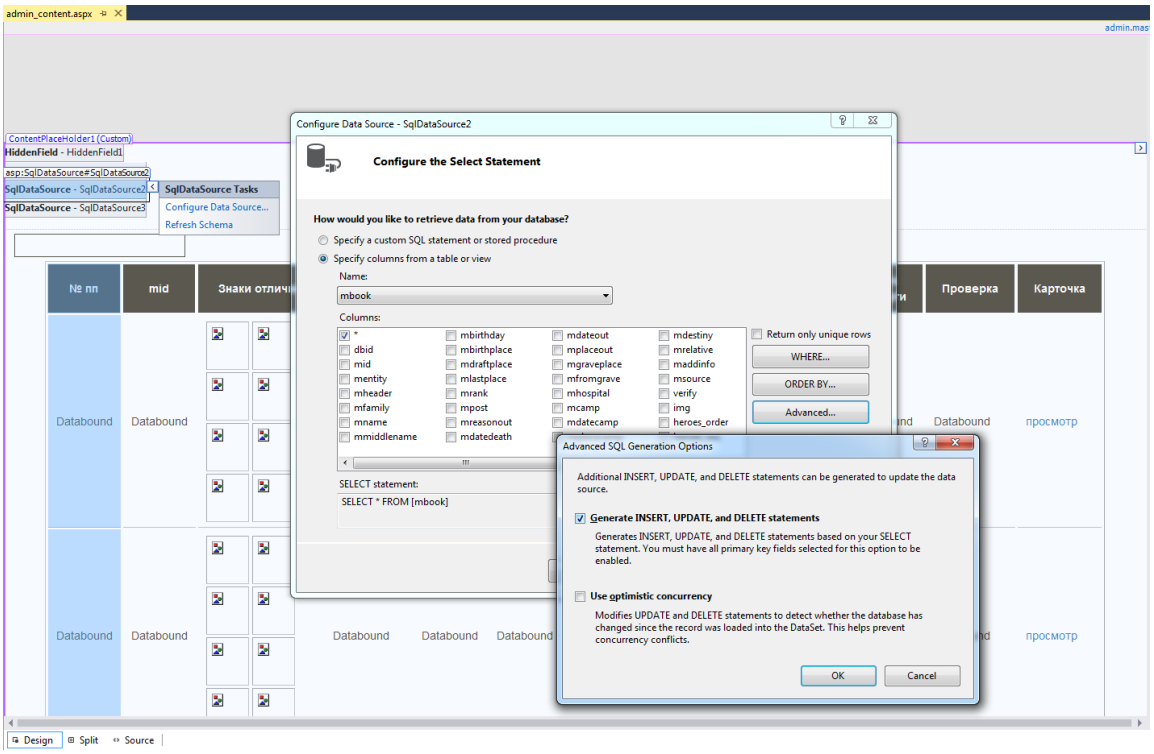


Рисунок 13. Настройка SqlDataSource с помощью конфигуратора источника данных

Настройка элемента FormView (листинг 29) будет включать в себя настройку вспомогательных элементов (если есть необходимость их использования), поскольку используется как универсальная форма для сочетания базовых элементов управления, или настройку самого элемента FormView (в зависимости от разновидности хранимой информации).

Попробуем использовать в качестве вспомогательных элементов компонент DetailsView (листинг 29) для работы с данными выбранной ранее записи и компонент Image для подгрузки изображения. Таким образом, необходимо настроить встроенный элемент DetailsView для работы с данными записи, а именно – настройка тэга TemplateField для каждого атрибута с учетом вариаций EditItemTemplate, InsertItemTemplate и ItemTemplate.

В поле типа CommandField необходимо настроить следующие свойства:

— ShowDeleteButton="True".
— ShowEditButton="True".
— EditText= "Редактировать".
— DeleteText="Удалить".
— UpdateText="Обновить".
— CancelText="Отменить".

Листинг 29. Настройка элементов FormView и DetailsView для работы с данными выбранной записи

```
<asp:FormView ID="FormView1" runat="server" DataSourceID="SqlDataSource2"
RenderOuterTable= "false">
    <ItemTemplate>
        <div id="photo" style="float: left; margin: 5px auto;">
            <asp:Image ID="Image9" runat="server" Style="margin-top: 30px;" ImageUrl='<%#
@"~\heroes\" + Eval("image") %>' /><br />
            <asp:Label ID="Label37" runat="server" Text='<%# Eval("mfamily")+"
"+Eval("mname").ToString().ToUpper()+" "+Eval("mmiddlename").ToString().ToUpper()
%>'></asp:Label>
            <asp:FileUpload ID="FileUpload2" runat="server" />
        </div>
        <div id="descr" style="margin: 5px auto;">
        <asp:DetailsView ID="DetailsView1" runat="server" Height="50px" Width="70%"
AutoGenerateRows="False" DataKeyNames="dbid" DataSourceID="SqlDataSource2"
OnItemDeleted="DetailsView1_ItemDeleted" OnItemInserted="DetailsView1_ItemInserted"
OnItemUpdated="DetailsView1_ItemUpdated" Caption="Карточка: " DefaultMode="Edit">
            <Fields>

                <asp:TemplateField HeaderText="Фамилия" SortExpression="mfamily">
                    <EditItemTemplate>
                      <asp:TextBox ID="TextBox4" runat="server" Text='<%# Bind("mfamily")
%>' Width="100%"></asp:TextBox>
                    </EditItemTemplate>
                    <InsertItemTemplate>
                      <asp:TextBox ID="TextBox4" runat="server" Text='<%# Bind("mfamily")
%>'></asp:TextBox>
                    </InsertItemTemplate>
                    <ItemTemplate>
                      <asp:Label ID="Label5" runat="server" Text='<%# Bind("mfamily")
%>'></asp:Label>
                    </ItemTemplate>
                </asp:TemplateField>

                …//далее аналогично настраиваются остальные атрибуты сущности
               <asp:CommandField ShowDeleteButton="True" ShowEditButton="True" EditText=
"Редактировать" DeleteText="Удалить" UpdateText="Обновить" CancelText="Отменить">
                </asp:CommandField>
            </Fields>
            <EditRowStyle HorizontalAlign="Left" VerticalAlign="Middle" />
            <FooterStyle BackColor="#484231"></FooterStyle>
            <HeaderStyle BackColor="#484231" ForeColor="White" Font-Size="14px"
HorizontalAlign="Left" />
            <PagerStyle BackColor="#484231" ForeColor="White" HorizontalAlign="Center"
Height="30px" Width="30px" />
        </asp:DetailsView>
        </div>
    </ItemTemplate>
</asp:FormView>
```

Для работы с дополнительными материалами к выбранной ранее записи необходимо организовать аналогичным образом источник данных (листинг 30).

Листинг 30. Настройка источника данных SqlDataSource для работы с дополнительными материалами записи

```
<asp:SqlDataSource ID="SqlDataSource3" runat="server" ConnectionString="<%$
ConnectionStrings:qwe %>" ConflictDetection="OverwriteChanges"
       DeleteCommand="DELETE FROM [docs] WHERE [id] = @original_id"
       InsertCommand="INSERT INTO [docs] ([dbid], [path]) VALUES (@dbid, @path)"
       OldValuesParameterFormatString="original_{0}"
       SelectCommand="SELECT * FROM [docs] WHERE ([dbid] = @dbid)">
    <DeleteParameters>
        <asp:ControlParameter ControlID="ListView1" Name="original_id" PropertyName="Selected
Value" Type="Int32" />
    </DeleteParameters>
    <InsertParameters>
        <asp:ControlParameter ControlID="GridView1" Name="dbid" PropertyName="SelectedValue"
Type="Int32" />
        <asp:ControlParameter ControlID="HiddenField1" Name="path" PropertyName="Value"
Type="String" />
    </InsertParameters>
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="dbid" PropertyName="SelectedValue"
Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

Как видно из листинга, компонент SqlDataSource поддерживает возможности удаления, добавления и изменения данных для работы с дополнительными материалами. Поэтому настройка ListView будет значительно упрощена (листинг 31).

Листинг 31. Настройка ListView

```
<asp:ListView ID="ListView1" runat="server" DataKeyNames="id" DataSourceID="SqlDataSource3"
InsertItemPosition="LastItem">
    <EmptyDataTemplate>
         <span>Данные отсутствуют.</span>
    </EmptyDataTemplate>
    <InsertItemTemplate>
        <asp:FileUpload ID="FileUpload1" runat="server" Width="70%" />
        <asp:Button ID="InsertButton" runat="server" Text="Добавить" OnClick="Button1_Click"
UseSubmitBehavior="false" />
    </InsertItemTemplate>
    <ItemTemplate>
        <div>
            <asp:Image ID="Image1" runat="server" Width="30%" ImageUrl='<%#
@"~\docs\"+Eval("path") %>' Style="margin: 15px;" />
            <asp:Button ID="DeleteButton" runat="server" CommandName="Delete" Text="Удалить"
Style="display: block; margin: 0 auto;" />
        </div>
    </ItemTemplate>
    <LayoutTemplate>
        <div id="itemPlaceholderContainer" runat="server">
            <span runat="server" id="itemPlaceholder" />
        </div>
        <div style="width: 70%; margin: 15px auto;">
            <asp:DataPager ID="DataPager1" runat="server" PageSize="3">
                <Fields>
                    <asp:NextPreviousPagerField ButtonType="Button" NextPageText="Вперед"
PreviousPageText="Назад" ShowNextPageButton="false" ShowLastPageButton="false"
ShowFirstPageButton="false" ShowPreviousPageButton="false" />
                    <asp:NumericPagerField />
                </Fields>
            </asp:DataPager>
        </div>
    </LayoutTemplate>
</asp:ListView>
```

Данная верстка сопровождается небольшим программным кодом для организации загрузки дополнительных материалов на сервер (листинг 32).

Листинг 32. Загрузка файлов на сервер

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Random rnd = new Random();
        FileUpload fu = (FileUpload)((Button)sender).Parent.FindControl("FileUpload1");
        try
        {
            if (fu.HasFile)
            {
                HttpPostedFile postedfile = fu.PostedFile;
                HiddenField1.Value = postedfile.FileName;
                var timestamp = (Int32)(DateTime.UtcNow.Subtract(new DateTime(1970, 1,
1))).TotalSeconds;
                string[] splittedName = HiddenField1.Value.Split('.');
                string newName = timestamp.ToString() + rnd.Next(0,100000).ToString() + '.' +
splittedName[splittedName.Length - 1];
                postedfile.SaveAs(Server.MapPath("..") + @"\docs\" + newName);
                HiddenField1.Value = newName;
                Response.Write("<script>alert('Файл добавлен.');</script>");
                ((Button)sender).CommandName = "insert";
            }
            else { Response.Write("<script>alert('Файл не выбран.');</script>");
                ((Button)sender).CommandName = "";
            }
        }
        catch { Response.Write("<script>alert('Что-то пошло не так');</script>");
        }
    }
```

После внесения изменений в запись необходимо обновить данные в таблице GridView, что делается довольно быстро (листинг 33):

Листинг 33. Обновление данных в GridView при добавлениии, обновлении или удалении информации

```
protected void DetailsView1_ItemUpdated(object sender, DetailsViewUpdatedEventArgs e)
    {
        GridView1.DataBind(); //соединение элемента GridView с источником данных
        GridView1.SelectRow(-1); //отсутствие выбранного элемента
    }

protected void DetailsView1_ItemInserted(object sender, DetailsViewInsertedEventArgs e)
    {
        GridView1.DataBind();
        GridView1.SelectRow(-1);
    }

    protected void DetailsView1_ItemDeleted(object sender, DetailsViewDeletedEventArgs e)
    {
        GridView1.DataBind();
        GridView1.SelectRow(-1);
    }
```

Теперь можно с уверенностью сказать, что все элементы привязки к данным работают согласованно и осуществляют поэтапную обработку большого массива данных основных сущностей БД.

Генерация иерархического дерева / меню (в применении с SiteMapDataSource)

Формулировка задачи. Необходимо создать иерархическое меню сайта на основе использования XML-данных.

Уровень сложности: простой.

Решение

Для начала необходимо создать файл web.sitemap (шаблон Site Map) и определить логическую структуру сайта с помощью элементов <siteMap> и <siteMapNode> (листинг 34).

Листинг 34. Пример Web.sitemap

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="~/default.aspx" title="Главная" description="Главная страница сайта">
    <siteMapNode title="Товары" description="Наши товары" url="~/Products.aspx" >
      <siteMapNode title="Комплектующие ПК" description="Процессоры, видеокарты, жесткие
диски"
        url="~/Hardware.aspx" />
      <siteMapNode title="Программы" description="Visual Studio, Expression Blend, Windows
Server"
        url="~/Software.aspx" />
    </siteMapNode>

    <siteMapNode title="Службы" description="Наши службы тех. поддержки, обучения и др."
                 url="~/Services.aspx">
      <siteMapNode title="Обучение" description="Мы научим использовать вас наши программные
продукты"
        url="~/Training.aspx" />
      <siteMapNode title="Консультации" description="Задавайте ваши вопросы"
        url="~/Consulting.aspx" />
      <siteMapNode title="Тех. поддержка" description="Помогаем 24/7"
        url="~/Support.aspx" />
    </siteMapNode>

  </siteMapNode>
</siteMap>
```

Далее необходимо поместить на MasterPage элемент SiteMapDataSource, а также поместить и настроить компонент TreeView (листинг 35).

Листинг 35. Структура MasterPage

```
<body>
    <form id="form1" runat="server">
        <div>
            <table>
                <tr>
                    <td style="width: 226px; vertical-align: top;">
                        <!-- Здесь будут размещаться элементы навигации -->
                    </td>
                    <td style="vertical-align: top; color: green; padding-left: 20px">
                        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server" />
                    </td>
                </tr>
            </table>
        </div>
        <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
        <asp:TreeView ID="TreeView1" runat="server" DataSourceID="SiteMapDataSource1">
        </asp:TreeView>
    </form>
</body>
</html>
```

Вместо элемента TreeView можно также использовать элемент Menu, который по настройкам свойств не сильно отличается от первого.

Поддержка нескольких языков на основе использования БД и ресурсных файлов проекта

Формулировка задачи. Необходимо перевести содержимое сайта на несколько языков (такое требование справедливо для научных журналов).

Теоретическая справка. Данная задача решается с применением ресурсных файлов в качестве словаря для всех элементов управления (*Рисунок 14*), а также использованием БД – содержимое статичных страниц необходимо хранить в БД сразу на нескольких языках (например, на русском и английском – content и content_en).

SABI-PC.culture - dbo.pages

id	name	contentt	contentt_en
1	Информация о журнале	<h1 style="text-align: center;"><span style="color: #...	<h1 style="text-align: center;">Information about journal</stro...
4	Правила рецензирования	<h1 style="text-align: center;">Правила рецензиро...	<h1 style="text-align: center;">PEER-REVI...
5	Перечень требований и условий публикации научных ст...	<header><h1 style="text-align: center;">Перечень ...	<h1 style="text-align: center;">REQUIRE...
6	Требования к оформлению статей	<header><h1 style="text-align: center;">Требовани...	<h1 style="text-align: center;">REQUIRE...
7	Цель и задачи	<header><h1 style="text-align: center;">Цель и зад...	<h1 style="text-align: center;">PURPOSE ...
8	Состав редакционной коллегии	<header><h1 style="text-align: center;">Редакцион...	<h1 style="text-align: center;">EDITORIA...
10	Техническая редакция	<header><h1 style="text-align: center;">Техническ...	<h1 style="text-align: center;">TECHNIC...
11	Этический кодекс	<header><h1 style="text-align: center;">Этический ...	<h1 style="text-align: center;">THE ETHI...
NULL	NULL	NULL	NULL

Рисунок 14. Примерная организация хранения содержимого страниц сайта на уровне БД

Также необходимо наличие MasterPage для целенаправленного управления страницами, а также для формирования конкретной структуры сайта.

Уровень сложности: средний.

Решение

Для организации поддержки сайтом многоязычности необходимо поместить на шапку сайта (блок header элемента MasterPage) элемент управления, отвечающий за выбор пользователем языка сайта (листинг 36).

Листинг 36. Настройка выпадающего списка для выбора языка сайта

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged" >
    <asp:ListItem Text="Выберите язык сайта" ></asp:ListItem>
    <asp:ListItem Text="Русский" Value="ru" ></asp:ListItem>
    <asp:ListItem Text="English" Value="en-us" ></asp:ListItem>
</asp:DropDownList>

Программный код элемента DropDownList
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        Session["lang"] = DropDownList1.SelectedValue;
        Response.Redirect(Request.Url.AbsoluteUri);
    }
```

Находясь в режиме Design элемента MasterPage, необходимо выбрать пункт меню Tools / Generate Local Resource… для создания ресурсного файла *masterr.master.resx* в директории проекта App_LocalResources для языка сайта по умолчанию. После выполнения этих действий каждый элемент управления MasterPage получит свойство *meta:resourcekey* для работы с ресурсными файлами (их может быть несколько). Далее, скопировав данный файл, необходимо разместить дополнительный ресурсный файл для реализации второго языка сайта (в нашем случае это английский язык) с конкретным названием согласно выбранному языку – *masterr.master.en-us.resx* (название ресурсного файла отличается от названия предыдущего

ресурсного файла на префикс en-us, что соответствует языку сайта, для которого он будет задействован). В ресурсном файле конкретного языка необходимо для каждого элемента управления указать правильный перевод текстового значения, поскольку по умолчанию все указанные значения компонентов MasterPage будут даны на русском языке (точнее, на языке сайта по умолчанию).

Для элемента MasterPage также необходимо написать отдельный класс basePage.cs[12] (листинг 37).

Листинг 37. Класс baseP.cs для MasterPage

```
using System.Globalization;
using System.Threading;
using System.Web.UI;

/// <summary>
/// Summary description for baseP
/// </summary>
public class baseP:Page
{
    protected override void InitializeCulture()
    {
        if (Session["lang"] != null)
        {
            Culture = Session["lang"].ToString();
            UICulture = Session["lang"].ToString();
            Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(Session
["lang"].ToString());
            Thread.CurrentThread.CurrentUICulture = CultureInfo.CreateSpecificCulture(Session
["lang"].ToString());
        }
        base.InitializeCulture();
    }

    public baseP()
    {
        //
        // TODO: Add constructor logic here
        //
    }
}
```

Теперь для каждой статичной страницы можно создать пустую страницу с подгружаемым из БД содержимым на нужном языке (листинг 38 приведен на примере страницы default.aspx):

Листинг 38. Содержимое страницы default.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/masterr.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div style="display: inline-block; padding: 15px; margin: 0 auto; width: 100%;"
runat="server" id="content_div">
    </div>
</asp:Content>
```

Программный код к странице default.aspx в данном случае реализует подгрузку выбранного пользователем языка сайта (листинг 39) с указанием созданного ранее класса basePage.

[12] метод InitializeCulture() существует только у класса Page, у класса MasterPage его необходимо определить.

Листинг 39. Программный код к странице:

```
using System;
using System.Configuration;
using System.Data.SqlClient;

public partial class _Default : baseP
{
    protected void Page_Load(object sender, EventArgs e)
    {
        SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["qwe"].
ConnectionString);
        con.Open();
        var s = "";
        if (Session["lang"].ToString() == "ru")
        {
            s = "select contentt from pages where id = 1";
        }
        else
        {
            s = "select contentt_en from pages where id = 1";
        }
        var cmd = new SqlCommand(s, con);
        var reader = cmd.ExecuteReader();
        reader.Read();
        content_div.InnerHtml = reader.GetString(0);
        con.Close();
    }
}
```

Заменить класс Page на baseP в программном коде страницы необходимо у всех статичных страниц, чье содержимое хранится в БД и подгружается автоматически.

Использование элемента управления RegularExpressionValidator

Формулировка задачи. Необходимо осуществлять проверку вводимых пользователем данных на уровне сервера.

Уровень сложности: простой.

Решение

Применение компонента RegularExpressionValidator на практике является довольно примитивным и основывается на навыках работы с регулярными выражениями (листинг 40).

Листинг 40. Применение элемента управления RegularExpressionValidator

```
<div class="form-group">
    <asp:Label ID="label1" for="textbox1" runat="server" Text="Фамилия"></asp:Label>
        <div class="col-sm-7">
            <asp:TextBox ID="textbox1" runat="server" placeholder="Например, Иванов"
ClientIDMode="Static"></asp:TextBox>
            <asp:RegularExpressionValidator ID="CustomValidator1" runat="server"
ErrorMessage="Введены запрещенные символы!" Type="String" ControlToValidate="textbox1"
Display="Dynamic" ValidationExpression='[А-яёЁ 0-9]+'>
            </asp:RegularExpressionValidator>
        </div>
</div>
```

Использование элемента управления CustomValidator для организации проверки данных на клиенте и на сервере

Формулировка задачи. Необходимо организовать проверку вводимой пользователем информации как «на стороне сервера», так и «на клиенте».

Уровень сложности: простой.

Решение

Одним из сложных элементов управления для организации проверки данных на достоверность является CustomValidator. Поэтому для большей наглядности ниже (листинг 41) приводится пример использования данного элемента.

Листинг 41. Пример использования элемента управления CustomValidator

```
Реализация программного кода (на сервере):
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs
args)
{
      try { args.IsValid = (int.Parse(args.Value) % 10 == 0); }
      catch { args.IsValid = false; }
}

JavaScript-процедура проверки на стороне клиента:
function EmpIDClientValidate(ctl, args) {
      args.IsValid = (args.Value % 10 == 0);
}

XAML-верстка (на сервере):
<asp:TextBox runat="server" Width="200px" ID="TextBox1" />
<asp:CustomValidator runat = "server" ID = "CustomValidator1" ControlToValidate =
"TextBox1" ClientValidationFunction="EmpIDClientValidate" ErrorMessage="Неверное значение,
не кратно 10" Display="dynamic" OnServerValidate="CustomValidator1_ServerValidate">*
</asp:CustomValidator>
```

Использование элементов инфографики на основе компонента Chart

Формулировка задачи. Необходимо по нажатию на кнопку отобразить результаты пользовательского запроса в виде отдельных элементов инфографики.

Уровень сложности: средний.

Решение

Для реализации данной задачи необходимо поместить на форму элементы управления Button и Chart без применения настроек, поскольку все необходимые параметры по результатам запроса к БД можно установить программно (листинг 42).

Листинг 42. Реализация настроек элемента Chart на уровне программного кода

```
protected void Button7_Click(object sender, EventArgs e)
    {
        //организация запроса к БД
        SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["qwe"].
ConnectionString);
        con.Open();
        string s = "SELECT publication_type.publication_type as publication_type, count(*) AS
count FROM publication inner join scientist_publication ON publication.id =
scientist_publication.publication inner join scientist ON scientist_publication.scientist =
scientist.id inner join scientist_organization on scientist_organization.scientist =
scientist.id inner join organization on scientist_organization.organization=organization.id
inner join publication_type on publication_type.id = publication.publication_type where
(organization.id like @org) GROUP by publication_type.publication_type";
        SqlDataAdapter adapter = new SqlDataAdapter();
        SqlCommand command = new SqlCommand(s, con);
        command.CommandType = CommandType.Text;
        command.Parameters.Clear();
        command.Parameters.Add("@org", SqlDbType.Int);
        command.Parameters["@org"].Value = Session["id_org"];
        adapter.SelectCommand = command;
        DataTable dt = new DataTable();
        adapter.SelectCommand = command;
        adapter.Fill(dt);
        //настройка элемента Chart
        Chart1.DataSource = dt; //загрузка результатов запроса в компонент Chart
        Chart1.Series[0].XValueMember = dt.Columns[0].ToString();
        Chart1.Series[0].YValueMembers = dt.Columns[1].ToString();

        //настройка стилей элемента Chart
        Chart1.BackColor = Color.Transparent; //прозрачность фона
        Chart1.BorderlineColor = Color.Transparent;
        Chart1.BorderSkin.BackColor = Color.Transparent;
        Chart1.TextAntiAliasingQuality = TextAntiAliasingQuality.Normal; //сглаживание текста
        Chart1.ChartAreas["ChartArea1"].BackColor = Color.Transparent;
        Chart1.ChartAreas["ChartArea1"].AxisX.TitleFont = new Font("Courier New", 12,
FontStyle.Regular); //стиль заголовка оси X
        Chart1.ChartAreas["ChartArea1"].AxisX.IntervalOffset = 1;
        Chart1.ChartAreas["ChartArea1"].AxisX.Interval = 1;
        Chart1.ChartAreas["ChartArea1"].AxisY.TitleFont = new Font("Courier New", 12,
FontStyle.Regular);
        Chart1.ChartAreas["ChartArea1"].AxisX.ArrowStyle = AxisArrowStyle.Triangle;
        Chart1.ChartAreas["ChartArea1"].AxisY.ArrowStyle = AxisArrowStyle.Triangle;
        Chart1.ChartAreas["ChartArea1"].AxisX.LabelAutoFitStyle = LabelAutoFitStyles.WordWrap;
        Chart1.ChartAreas["ChartArea1"].AxisY.LabelAutoFitStyle = LabelAutoFitStyles.WordWrap;
        Chart1.ChartAreas["ChartArea1"].AxisX.LabelStyle.Font = new Font("Courier New", 10,
FontStyle.Bold);
        Chart1.ChartAreas["ChartArea1"].AxisY.LabelStyle.Font = new Font("Courier New", 10,
FontStyle.Bold);
        Chart1.ChartAreas["ChartArea1"].AxisX.MajorGrid.Enabled = false;
        Chart1.ChartAreas["ChartArea1"].AxisY.MajorGrid.Enabled = true;
        //стиль линии
        Chart1.ChartAreas["ChartArea1"].AxisY.MajorGrid.LineDashStyle = ChartDashStyle.Dash;
        Chart1.ChartAreas["ChartArea1"].AxisY.MajorGrid.LineColor = Color.DarkBlue;
        Chart1.ChartAreas["ChartArea1"].AxisX.LabelStyle.Angle = -45;
        Chart1.Series[0].Font = new Font("Courier New", 10, FontStyle.Bold);
        Chart1.Series[0].Color = Color.FromArgb(46, 57, 81);
        Chart1.Series[0].BorderDashStyle = ChartDashStyle.NotSet;
        Chart1.Series[0].BorderColor = Color.Transparent;
        Chart1.ChartAreas["ChartArea1"].Area3DStyle.Enable3D = false;
        con.Close();
    }
```

Результат выполнения программного кода (*Рисунок 15*) отобразится пользователю до трех секунд (удовлетворительное время ожидания ответа на запрос, без задержек).

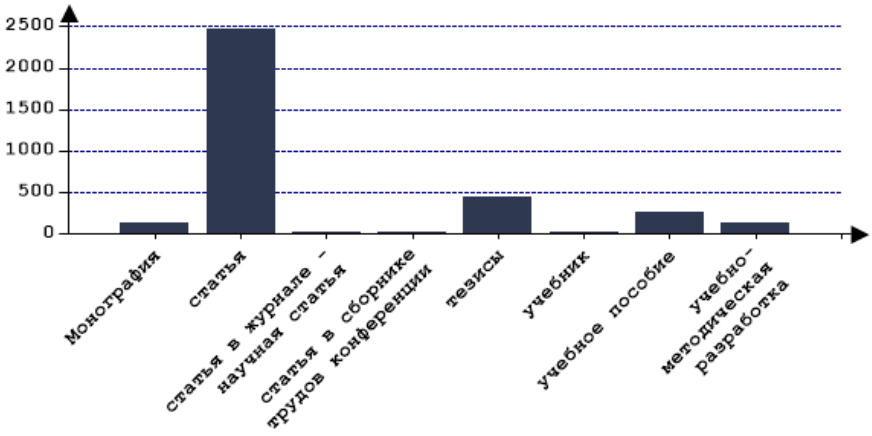


Рисунок 15. Применение диаграмм в проектах ASP.NET

Обработка информации через RSS-канал на примере открытого канала сайта http://syktsu.ru

Формулировка задачи. Необходимо дублировать информацию из новостной ленты сайта университета с использованием RSS-канала.

Уровень сложности: простой.

Решение

По нажатию на значок RSS-канала (https://www.syktsu.ru/news/rss/) в верхней части новостной ленты сайта университета можно пройти по заданной ссылке и ознакомиться со структурой данных в XML-формате. Например, для сайта университета формат данных будет следующим (листинг 43):

Листинг 43. Пример структуры организации данных в XML-формате

```
<rss version="2.0">
<channel>
<title>Новости</title>
<link>https://syktsu.ru</link>
<description/>
<lastBuildDate>Fri, 28 Feb 2020 08:49:40 +0300</lastBuildDate>
<ttl>60</ttl>

<item>
<title>О долге и памяти</title>
<link>https://syktsu.ru/news/30907/</link>
<description>
В СГУ им. Питирима Сорокина прошла встреча, посвященная созранению памяти о героях-
фронтовиках.
</description>
<enclosure url=https://syktsu.ru/upload/iblock/518/IMG_8115.JPG length="210805"
type="image/jpeg"/>
<category>Внеучебная жизнь</category>
<pubDate>Thu, 27 Feb 2020 15:00:00 +0300</pubDate>
</item>
…
</channel>
</rss>
```

Исходя из структуры файла становится понятно, что каждая следующая новость доступна в тэге item. Таким образом, для организации доступа к новостной ленте целесообразно использовать элемент XmlDataSource (листинг 44), настройки которого сводятся к минимуму.

Листинг 44. Настройка вывода новостной ленты университета

```
<asp:xmldatasource id="RSSFeed" runat="server" datafile="http://www.syktsu.ru/news/rss/"
xpath="rss/channel/item">
</asp:xmldatasource>
    <div style="margin: 30px 20px; margin-left: 10%;">
        <asp:DataList ID="FeedList" runat="server" DataSourceID="RSSFeed">
            <ItemTemplate>
                <div style="text-align: left; font-family: GOTHIC;">
                    <h2><a href='<%# XPath("link") %>'>
                        <asp:Label runat="server" ID="TitleLabel" Text='<%# XPath("title") %>'
Style="font-family: VinSansPro;">
                        </asp:Label></a>
                    </h2>
                    <%# XPath ("enclosure.url") %>
                    <asp:Label runat="server" ID="DescriptionLabel" Text='<%#
XPath("description") %>' /><br /><br />
                    <a href='<%# XPath("link") %>' style="">Подробнее в первоисточнике</a><br
/>
                    <i>Опубликовано: <%#XPath("pubDate")%></i>
                    <br /><br />
                </div>
            </ItemTemplate>
        </asp:DataList>
    </div>
```

Обращаясь к соответствующим тэгам раздела Item, можно осуществлять вывод необходимой информации. Стоит также отметить, что для реализации данной задачи не нужно производить дополнительных манипуляций в части программного кода.

Использование SVG-файлов в проектах ASP.NET

Формулировка задачи. Создать интерактивный объект векторной графики (карту Республики Коми, разделенную на административные округа, подсвечивающиеся при наведении на них курсором) и разместить его на сайте.

Теоретическая справка. Чтобы графический объект (иконки, простые рисунки, стикеры) можно было масштабировать (т. е. увеличивать/уменьшать) без потери качества, применяется векторная графика. Для работы с векторной графикой можно использовать один из следующих графических редакторов: *Adobe Illustrator*, *CorelDRAW*, *InkSkape*, *Asymptote* и др.

Структуру файлов векторной графики (например, в формате SVG, который, по сути, представляет собой обычный XML-файл) можно просматривать в текстовом редакторе. Только что созданный SVG-файл без какого-либо содержания с рабочей областью 1920 на 1080 пикселей, созданный в одной из последних версий Adobe Illustrator, выглядит следующим образом (листинг 45):

Листинг 45. Содержимое пустого SVG-файла

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 23.0.6, SVG Export Plug-In . SVG Version: 6.00
Build 0)  -->
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
      viewBox="0 0 1920 1080" style="enable-background:new 0 0 1920 1080;"
xml:space="preserve">
</svg>
```

℟ Если открыть таким же образом файлы, созданные в более ранних версиях Adobe Illustrator, легко заметить, что в них содержится гораздо больше служебной информации (например, объявление DOCTYPE), которую можно удалить так же, как и первые три строки в листинге 45, – это никак не скажется на самом файле, браузер способен определять данную информацию самостоятельно.

Все важные атрибуты располагаются в корневом элементе svg. К ним относятся:

— Version, где указана версия svg.
— Xmlns, где указано пространство имен, чтобы избежать конфликта с языком разметки HTML или другими XML-документами. Адрес http://www.w3.org/2000/svg, указанный в пространстве имён, является идентификатором для браузера, в реальности такого адреса нет. Адрес http://www.w3.org/1999/xlink" указывается во избежание конфликта с ссылками HTML href.
— ViewBox, где указаны размеры рабочей области (или «холста»).
— Enable-background, который служит для указания фона.

Если в Adobe Illustrator нарисовать графические примитивы, то в самом файле (листинг 46) можно заметить, что информация о них помещается в специальные теги:

— <rect />, который хранит координаты одной из вершин прямоугольника, его ширину и высоту.
— <polygon />, который хранит координаты вершин многоугольника.
— <circle />, который хранит координаты центра окружности и ее радиус.
— <path />, который хранит информацию о точках и форме соединяющих их линий (позволяет рисовать незамкнутые фигуры).

Листинг 46. Содержимое SVG-файла, содержащего четыре графических примитива

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 23.0.6, SVG Export Plug-In . SVG Version: 6.00
Build 0)  -->
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
       viewBox="0 0 1920 1080" style="enable-background:new 0 0 1920 1080;"
xml:space="preserve">
<style type="text/css">
      .st0{fill:#333347;stroke:#000000;stroke-width:10;stroke-miterlimit:10;}
      .st1{fill:#4747BF;stroke:#333333;stroke-width:8;stroke-miterlimit:10;}
      .st2{fill:#1658B7;stroke:#888888;stroke-width:6;stroke-miterlimit:10;}
      .st3{fill:#8DA2C9;stroke:#AAAAAA;stroke-width:4;stroke-miterlimit:10;}
</style>
<rect x="264" y="226" class="st0" width="374" height="203"/>
<polygon class="st1" points="893.3,131.1 768.7,131.1 706.4,239 768.7,346.9
893.3,346.9 955.6,239 "/>
<path class="st2" d="M980.7,632L879.5,516.4l88.6-115.5c0,0,115.5-
1.6,115.5,115.5S980.7,632,980.7,632z"/>
<circle class="st3" cx="1276.7" cy="408.8" r="123.4"/>
</svg>
```

Также для удобства фигурам в соответствие устанавливаются классы (здесь от st0 до st3), для каждого из которых в теге <style> прописывается CSS-стиль. В данном случае (по умолчанию) стиль содержит в себе такие свойства, как цвет заливки (fill), цвет обводки (stroke), ее толщина (stroke-width), размер скоса, под которым в каждой точке соединяются линии (stroke-miterlimit), и др. Один стиль может соответствовать как одной, так и сразу нескольким фигурам.

Для группировки логически связанных между собой элементов применятся тег <g>, что позволяет объединить несколько разных элементов в одну группу или подгруппу. Данному тегу

можно присваивать свой идентификатор id. Соответственно, стили, применяемые к элементу <g> с назначенным идентификатором, наследуются всеми потомками.

Уровень сложности: средний.

Решение

Таким образом, чтобы изобразить карту, поделенную на несколько областей, можно фоном разместить на рабочей области настоящую карту и на новом слое поверх нее при помощи инструмента Pen (Перо) с необходимой плотностью расставить опорные точки, обрисовав внешние границы (Рисунок 16). При этом заливку для удобства рекомендуется оставить прозрачной, оставив только границу.

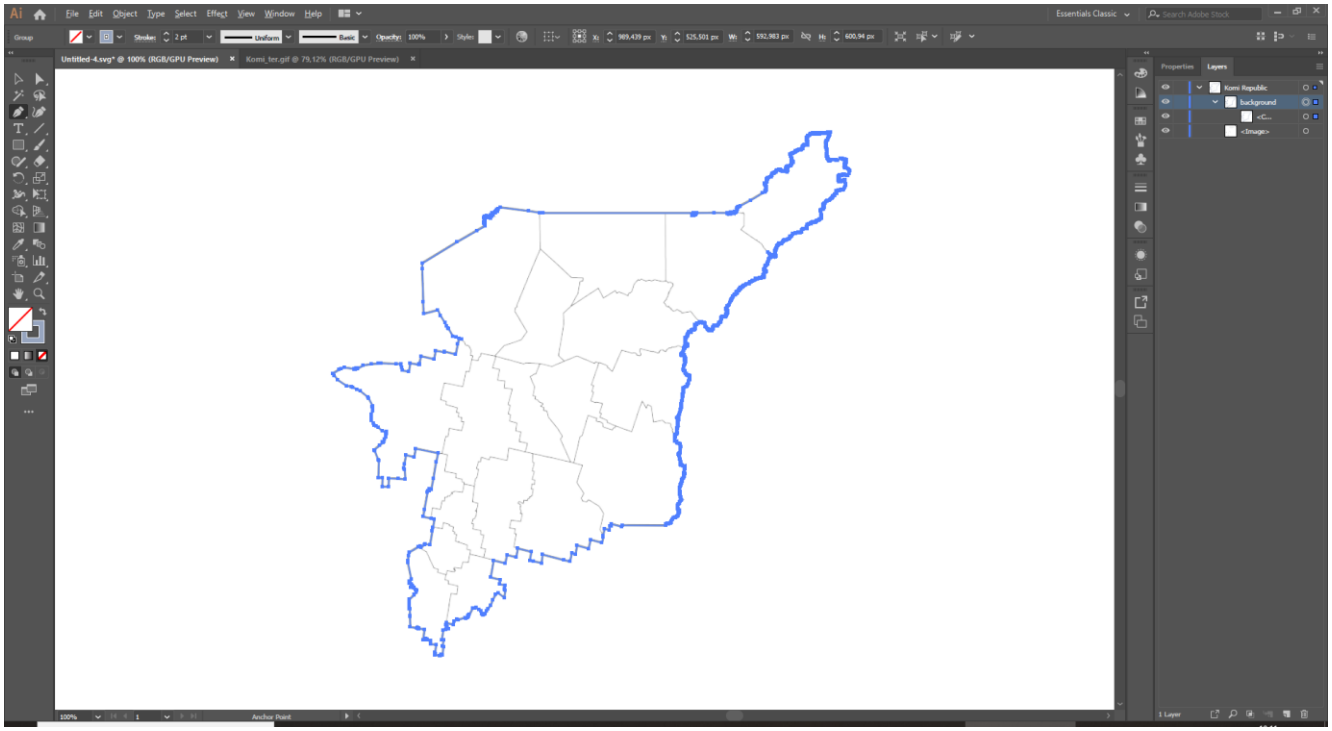


Рисунок 16. Контур карты в Adobe Illustrator CC 2019

Далее, чтобы выделить районы, необходимо выполнить следующие шаги (Рисунок 17):

— Создать копию слоя с картой: выделить его, нажать сочетание клавиш Ctrl+C, затем Ctrl+Shift+V, чтобы вставить на ту же позицию.
— Так же, как и в самом начале, обрисовать одну внутреннюю границу (автоматически создастся новый отдельный слой) и замкнуть фигуру, обрисовав контур вокруг района таким образом, чтобы он целиком оказался внутри новой фигуры.
— Зажав Ctrl, выделить на палитре слоев два слоя: с картой и с новой фигурой (они отметятся кружками).
— Выбрать инструмент Shape Builder (активируется сочетанием клавиш Shift+M).
— Зажать Alt (рядом с курсором появится знак «минус») и нажать левой кнопкой мыши на карте за районом, окруженным фигурой (эта область окрасится в серый цвет).

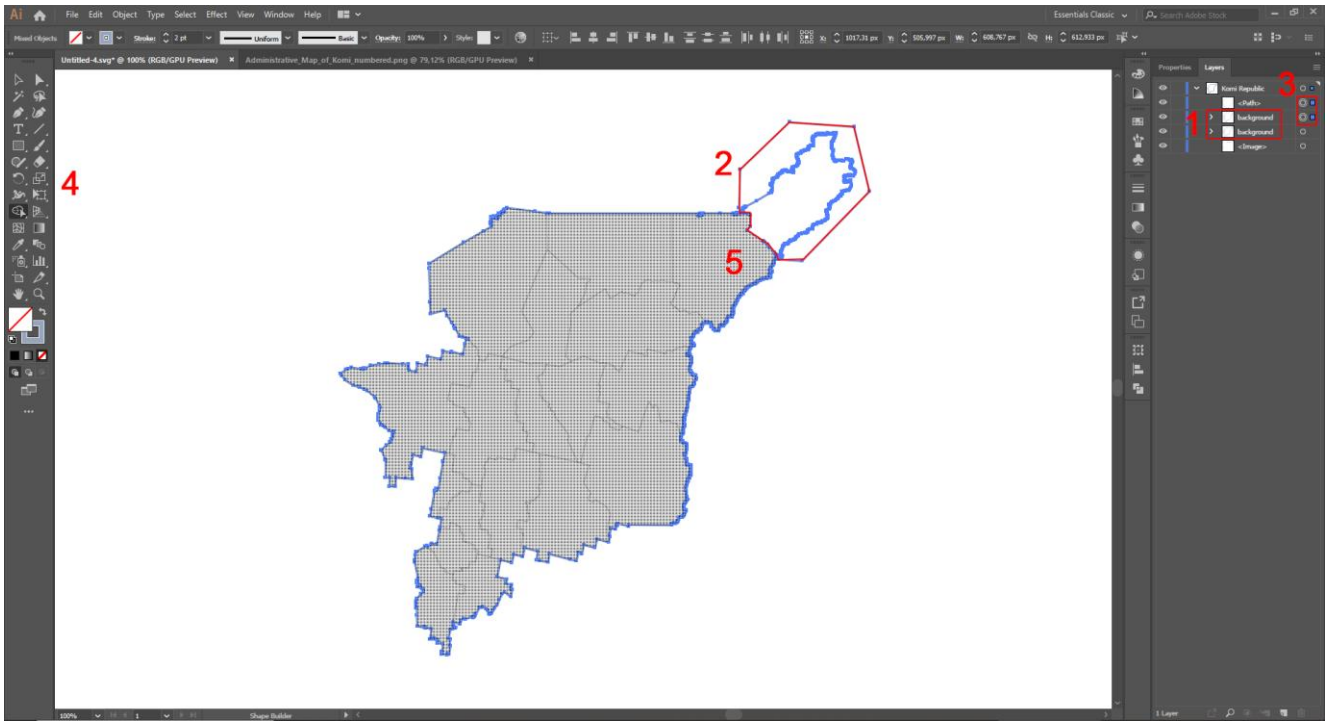


Рисунок 17. Последовательность действий при создании отдельных районов на карте

Таким образом, за исключением фонового изображения имеется три слоя: фигура, окружающая район, сам район отдельно от карты и непосредственно карта. На слое с картой отделенный район остался нетронутым. Чтобы его удалить, необходимо повторить алгоритм: выделить слои с картой и фигурой (не районом), выбрать инструмент Shape Builder, зажать Alt и нажать левой кнопкой мыши на районе и области фигуры вне района.

℟ В результате применения инструмента Shape Builder могут появляться лишние слои с отдельными участками границ. Их необходимо удалить вручную. Проверить их наличие можно, отключая все слои по очереди (иконка глаза слева от изображения слоя на палитре).

Описанную последовательность действий необходимо повторить для каждого района на карте. После этого можно добавить обводку, чтобы выделить границы.

Для того чтобы при наведении курсора подсвечивалась соответствующая область, необходимо открыть SVG-файл в текстовом редакторе и добавить соответствующий стиль. Ниже (листинг 47) приведен пример с одной областью, которой назначен класс distr. Ему соответствуют два стиля, второй (hover) активирует подсветку (постепенную смену цвета заливки) при наведении курсора на область.

Листинг 47. Структура SVG-файла с изображением Ижемского района

```
<svg version="1.1" id="Komi_Republic" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
	 viewBox="0 0 1920 1080" style="enable-background:new 0 0 1920 1080;"
xml:space="preserve">
<style type="text/css">
	.distr
		{
			fill:#55678D;
			fill-opacity:1;
			stroke:white;
			stroke-opacity:1;
			stroke-width:1;
		}

		.distr:hover
		{
			fill:#97A5BF;
			transition:fill 0.5s ease-in-out;
			cursor:pointer;
		}

		.descr
		{
			stroke:none;
		}

		.descr>text
		{
			fill: black;
		}
<g id="IZ">
	<g class="distr">
		<path d="M200.25,275l10.751-12l19.624-6.626l6.626-9.375l-8.75-14.625l-
3.375-5.875
		l-1.875-2.874l0.875-4.25l1.999-0.999l-0.499-1.626l16.751-
40.875l8.749-20.374l1.5-3.375l22.125,19.999l5.374,5.125l8.126,7.625
		l13.376,0.501l0.124,1.624l-0.499,1.25l0.124,0.375l-2.876,0.75l-
0.874,1l0.251,0.626l-0.251,0.75l0.375,0.375l-0.499,1.374
		L295.751,203l-8.125,14.873L279,225.875l0.625,11.498l-
6.124,37.125l6.624,6.251l-14.123,6L252,283.624l-15.375-0.876l-3.375,1.125
		l-28.125-7.749L200.25,275L200.25,275z"/>
		<g class="descr" id="IZ-city">
			<radialGradient id="Izhma_2_" cx="308.3042" cy="206.8027"
r="7.8384" gradientTransform="matrix(1.216 0 0 1.216 -116.9968 -14.0741)"
gradientUnits="userSpaceOnUse">
				<stop offset="0" style="stop-color: #FFFFFF"/>
				<stop offset="0.25" style="stop-color: #FFFFFF"/>
				<stop offset="1" style="stop-color: #FFFFFF;stop-
opacity:0"/>
			</radialGradient>
			<circle id="Izhma_1_" fill="url(#Izhma_2_)" cx="257.899"
cy="237.399" r="9.531"/>
			<text opacity="1" transform="matrix(1 0 0 1 244.877 255.1309)"
font-size="14">Ижма</text>
		</g>
	</g>
</g>
</svg>
```

Стоит отметить, что внутри класса distr размещен класс descr для отображения метки (окружность <circle />, залитая полупрозрачным круговым градиентом <radialGradient>) и

названия столицы округа. Чтобы они не наследовали стиль класса descr, к ним применяются отдельные стили (отменяющий обводку и заливающий текст черным цветом).

Финальный вариант карты изображен ниже (Рисунок 18).

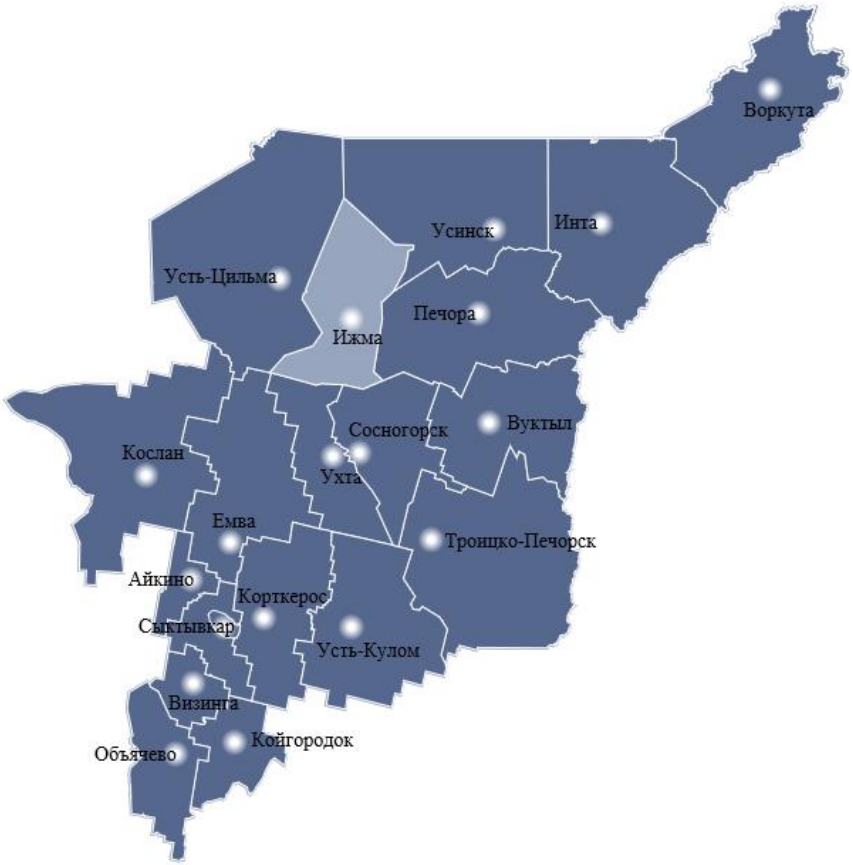


Рисунок 18. Интерактивная карта Республики Коми

Организация отправки писем на электронный адрес администрации сайта

Формулировка задачи. Организовать на стороне пользователя отправку научных публикаций для их последующего размещения на сайте журнала.

Уровень сложности: простой.

Решение

Для решения данной задачи необходимо реализовать пользовательский интерфейс согласно принятым требованиям для научных журналов (Рисунок 19):

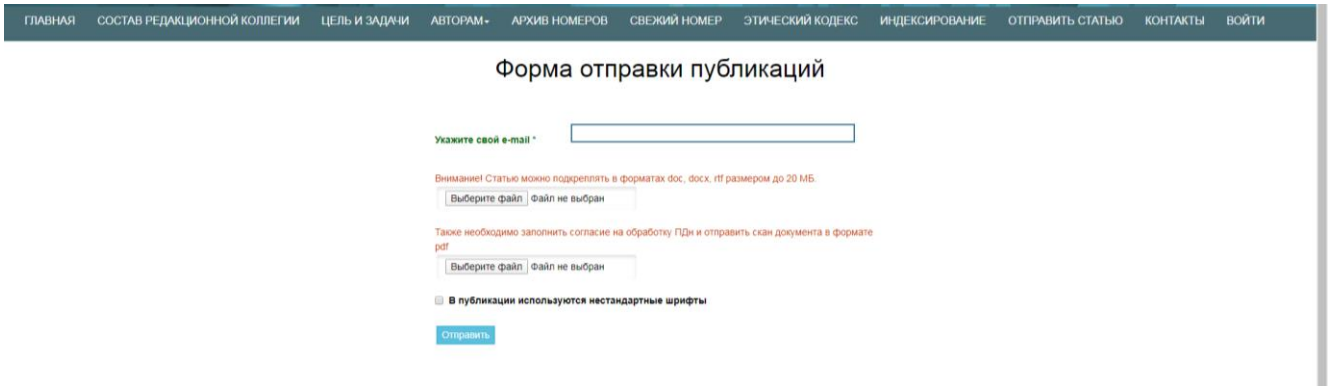


Рисунок 19. Интерфейс страницы для отправки материалов администрации журнала

Подобный интерфейс включает в себя стандартные лэйблы, кнопку, текстовое поле для ввода e-mail автора статьи и пару специфичных элементов FileUpload для загрузки материалов (листинг 48).

Листинг 48. XAML-верстка формы отправки публикаций

```
<h1 style="text-align: center">Форма отправки публикаций</h1>
    <br />
    <div style="display: block; margin: 0 auto; padding: 30px; width: 50%;">
        <table class="nav-justified" style="width: 100%;">
            <tr>
                <td style="width: 30%;">
                    <asp:Label ID="Label1" runat="server" Text="Укажите свой e-mail *"
style="color: #006600; font-weight: 700"></asp:Label></td>
                <td><asp:TextBox ID="TextBox1" runat="server" Width="90%" CssClass="text-info"
AutoCompleteType="Email" BorderStyle="Solid"></asp:TextBox>
                    <asp:RegularExpressionValidator ID="regexEmailValid" runat="server"
ValidationExpression="\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
ControlToValidate="TextBox1" ErrorMessage="Неверный формат электронной почты!">
</asp:RegularExpressionValidator>
                </td>
            </tr>
        </table><br />
        <asp:Label ID="Label2" runat="server" Text="Внимание! Статью можно подкреплять в
форматах doc, docx, rtf размером до 20 МБ." style="color: #CC3300"></asp:Label>
        <asp:FileUpload ID="FileUpload1" runat="server" BorderStyle="Solid" CssClass="btn
active" />
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ControlToValidate="FileUpload1" ErrorMessage="Only *.pdf"
ValidationExpression=".*\.([Pp][Dd][Ff])">
        </asp:RegularExpressionValidator>
        <br />
        <a href="imgs/Согласие-на-обработку-персональных-данных.doc" target="_blank">
            <asp:Label ID="Label3" runat="server" Text="Также необходимо заполнить согласие на
обработку ПДн и отправить скан документа в формате pdf" style="color: #CC3300">
            </asp:Label>
        </a>
        <asp:FileUpload ID="FileUpload2" runat="server" BorderStyle="Solid" CssClass="btn
active" />
        <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server"
ControlToValidate="FileUpload2" ErrorMessage="Only *.pdf"
ValidationExpression=".*\.([Pp][Dd][Ff])">
        </asp:RegularExpressionValidator>
        <br />
        <asp:CheckBox ID="CheckBox1" runat="server" Text="В публикации используются
нестандартные шрифты" CssClass="checkbox-inline" /><br /><br />
        <asp:Button ID="Button1" runat="server" Text="Отправить" CssClass="btn-info"
OnClick="Button1_Click" BorderStyle="Solid" />
        <br /><br /><br /><br />
    </div>
```

Естественно, для ввода пользователем информации в текстовое поле, а также для компонента FileUpload необходимо использовать компонент `RegularExpressionValidator` для организации валидации данных (работа компонента базируется на использовании регулярных выражений). Программный код для реализации отправки формы является достаточно понятным при подключении библиотек `System.Net.Mail` и `System.Web.UI` (листинг 49).

Листинг 49. Программный код для реализации отправки публикаций администрации сайта

```
using System.Net.Mail;
using System.Web.UI;
…

protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (TextBox1.Text != "" && FileUpload1.HasFile && FileUpload2.HasFile)
            {
                MailMessage mm = new MailMessage();
                mm.From = new MailAddress("вспомогательная почта для отправки писем от
клиентов");
                mm.To.Add(new System.Net.Mail.MailAddress("email администратора сайта"));
                mm.Subject = "Публикация статей в журнале ЧКО";
                if (CheckBox1.Checked == true) mm.Body = "Имеются нестандартные шрифты";
                mm.Body += "\n e-mail автора: " + TextBox1.Text;
                string publ = Path.GetFileName(FileUpload1.PostedFile.FileName);
                Attachment attachment = new Attachment(FileUpload1.PostedFile.InputStream,
publ);
                string PDn = Path.GetFileName(FileUpload2.PostedFile.FileName);
                Attachment attachment_2 = new Attachment(FileUpload2.PostedFile.InputStream,
PDn);
                mm.Attachments.Add(attachment);
                mm.Attachments.Add(attachment_2);
                SmtpClient client = new System.Net.Mail.SmtpClient("smtp.yandex.ru");
                client.EnableSsl = true;
                client.Credentials = new System.Net.NetworkCredential("логин", "пароль");
                client.Send(mm);
                mm.Dispose();
                ScriptManager.RegisterStartupScript(this, this.GetType(), "message",
"alert('Ваша публикация отправлена, спасибо!');", true);
            }
        }
        catch (Exception ex)
        {
            Response.Write("<script>alert('Что-то пошло не так.');</script>");
            ScriptManager.RegisterStartupScript(this, this.GetType(), "message", "alert('" +
ex.Message + "');", true);
        }

    }
```

После создания экземпляра класса `MailMessage` можно указать отправителя письма, получателя, а также перечень прикрепленных файлов для размещения публикации. После отправки письма `client.Send(mm);` необходимо в обязательном порядке разорвать соединение `mm.Dispose()`. Конструкция try-catch используется для предотвращения ошибок и реализации безопасного кода.

ЧАСТЬ 3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Лабораторная работа 1. Работа с серверными элементами управления Standard для создания статичной страницы

Цель работы – формирование навыков работы с серверными элементами управления Standard в процессе верстки статичных страниц сайта.

Количество времени: 2 часа.

Задание. Реализовать процесс заполнения специалистом резюме, используя серверные элементы управления категории Standard с последующей генерацией XML-документа (Рисунок 20).

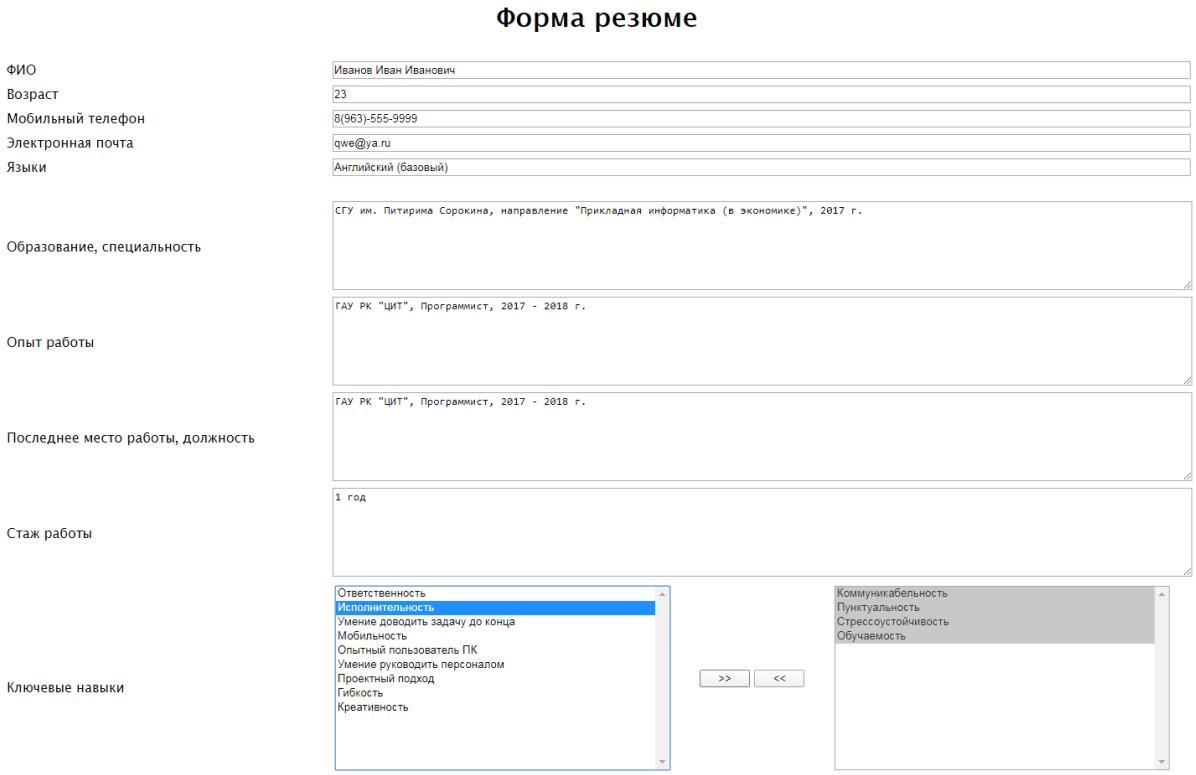


Рисунок 20. Форма заполения резюме с перечнем обязательных полей

Требования к заданию:

— При формировании формы резюме задействовать не менее 12 пунктов, среди которых обязательные разделы представлены на рисунке 20.

— Осуществить валидацию вводимых пользователем данных средствами специальных серверных компонентов.

— Организовать раздел «Ключевые навыки» на основе автоматического переноса пунктов между ListBox-ами по клику.

Лабораторная работа 2. Разработка простого многостраничного сайта с использованием шаблона MasterPage

Цель работы – изучить и применить на практике особенности проектирования и разработки статичных web-страниц на основе использования технологии ASP.NET.

Количество времени: 2 часа.

Задание. Разработать простой web-сайт по конкретному макету (Рисунок 21), включающий в себя от 2 до 5 статичных страниц с использованием блочной верстки на основе технологии ASP.NET по тематике «Минералы Севера». Для реализации требуемой структуры сайта необходимо использовать шаблоны MasterPage и SiteMap.

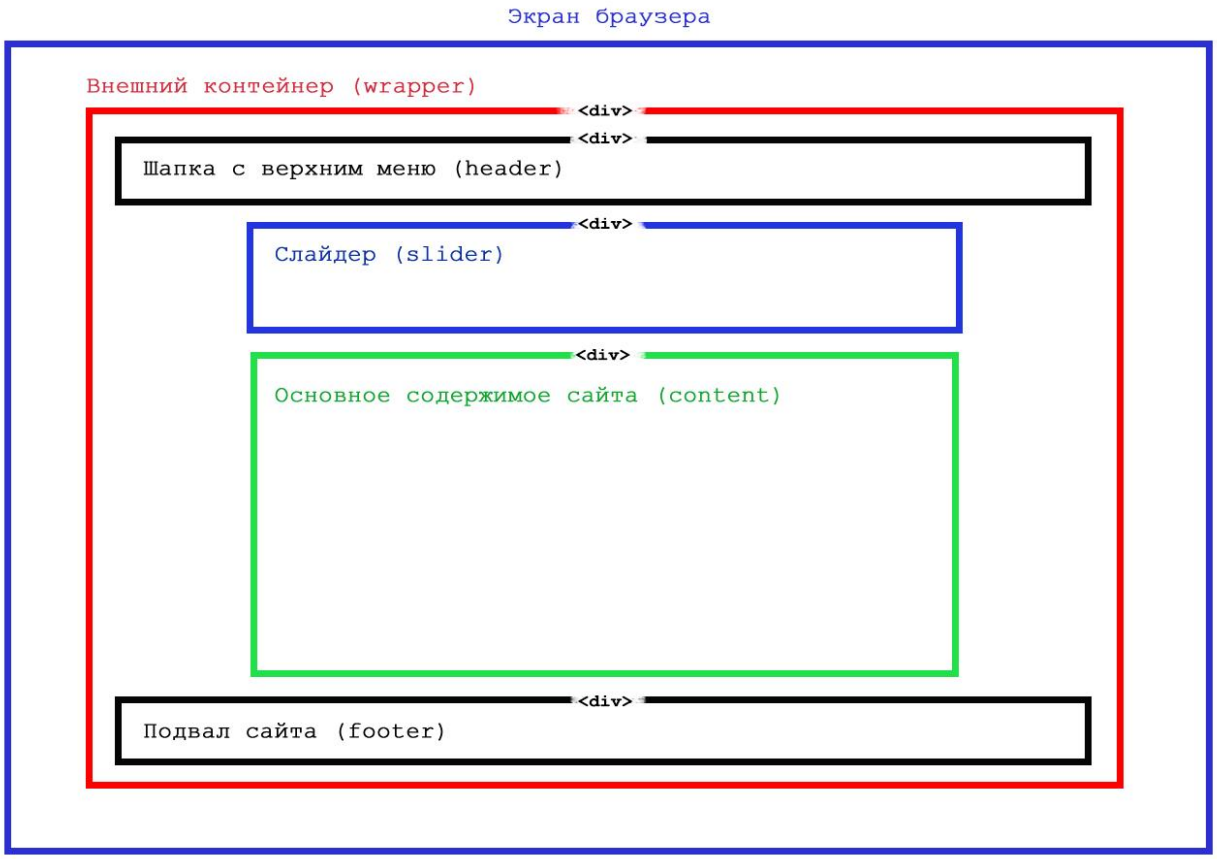


Рисунок 21. Требуемая структура сайта

Замечание. Для проектирования и разработки статичной страницы разрешается использовать серверные элементы управления вкладки Standard, а также стандартные тэги HTML5.

Лабораторная работа 3. Разработка web-приложения с использованием СУРБД MS SQL Server

Цель работы – применение технологий взаимодействия с БД (компоненты вкладки Data) при разработке web-приложений.

Количество времени: 8 часов.

Задание. Разработать несложный web-интерфейс (web-приложение категории Help Desk) для автоматизации процесса регистрации, учета и исполнения заявок с использованием СУРБД MS SQL Server (Express Edition) согласно прилагаемой структуре (Рисунок 22).

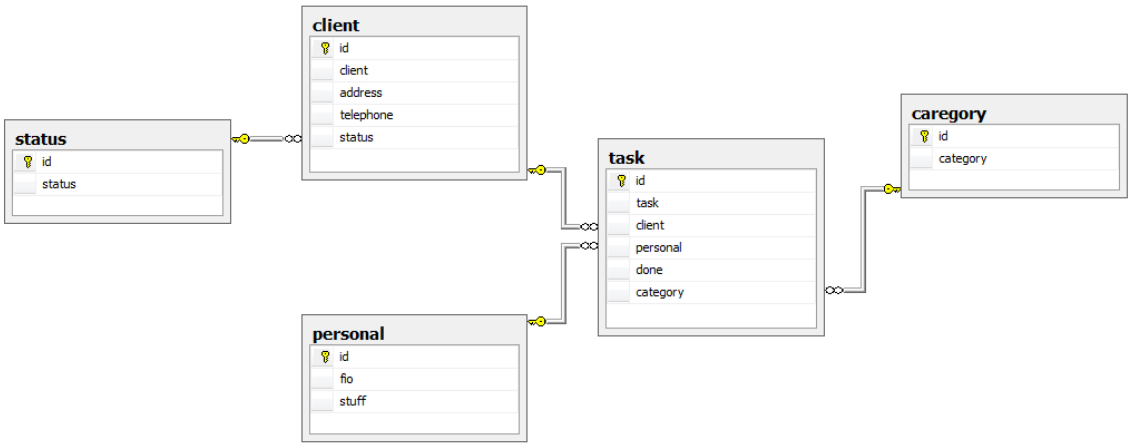


Рисунок 22. Требуемая структура БД

Требования к заданию:

- Написать скрипт на языке T-SQL для генерации структуры требуемой БД и связей.
- Задействовать два типа ролей – сотрудник IT-отдела и клиент.
- Реализовать регистрацию и авторизацию пользователей в системе.
- Реализовать личный кабинет для пользователей всех ролей.
- Организовать регистрацию и учет заявок в системе.
- Организовать рассылку уведомлений о статусе заявки по электронной почте.

Внимание! В качестве CSS-стилей использовать в проекте готовую библиотеку bootstrap последней версии (скачать с сайта разработчика). Верстка сайта должна быть адаптирована под различные устройства (ноутбуки, стационарные компьютеры, планшеты, мобильные телефоны).

Лабораторная работа 4. Календарь событий с использованием модальных окон

Цель работы – формирование навыков работы с серверным элементом управления Calendar на основе взаимодействия с XML-файлами в качестве хранилища данных, а также применение навыков работы с JavaScript.

Количество времени: 6 часов.

Задание. Разработать простой календарь событий студента для повышения активности участия в мероприятиях института и вуза.

Требования к заданию:

- Разработать панель администратора сайта для размещения студенческих мероприятий.
- Использовать XML-файл (или JSON-файл) в качестве хранилища данных.
- Организовать удобный интерфейс для просмотра пользователем календаря событий.
- Использовать JavaScript для реализации модальных окон с целью детального просмотра событий на конкретную дату.
- Реализовать возможность предварительной записи студента на мероприятие.

Лабораторная работа 5. Разработка SOAP Web-сервисов на основе WCF/ASMX

Цель работы – формирование навыков разработки web-сервисов для многозвенных АИС.

Количество времени: 4 часа.

Задание. Разработать SOAP web-сервис (на основе WCF/ASMX) для получения данных по студенческим мероприятиям на конкретную дату в формате dd.MM.yyyy. Выявить особенности разработки web-сервисов с использованием WCF и на основе ASMX.

Требования к заданию: выполнить реализацию AJAX.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ASP.NET [Электронный ресурс]. URL: http://professorweb.ru/my/ASP_NET/base/level1/aspnet_info.php (дата обращения: 10.05.2020).
2. ASP.NET (Материал из Википедии — свободной энциклопедии) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/ASP.NET (дата обращения: 15.05.2020).
3. LESS (язык стилей) (Материал из Википедии — свободной энциклопедии) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/LESS_(язык_стилей) (дата обращения: 17.05.2020).
4. Sass (Материал из Википедии — свободной энциклопедии) [Электронный ресурс]. URL: https://Ru.Wikipedia.Org/Wiki/Sass (дата обращения: 05.05.2020).
5. А. Фримен. ASP.NET 4.5 с примерами на C# 5.0 для профессионалов. 5-е изд. М.: ООО «И.Д. Вильямс», 2014. 1120 с.
6. Документация по .NET [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/?view=netframework-4.7.2 (дата обращения: 13.05.2020).
7. Изучение сессии в ASP.NET – все об IT и программировании [Электронный ресурс]. URL: http://www.cyberguru.ru/microsoft-net/asp-net/aspnet-session-exploration.html?showall=1 (дата обращения: 13.05.2020).
8. Какие типы сайтов бывают — полная классификация с примерами [Электронный ресурс]. URL: https://altblog.ru/vidy_sajtov/ (дата обращения: 20.05.2020).
9. Какое программирование самое востребованное в 2019 году [Электронный ресурс]. URL: https://skillbox.ru/media/code/kakoe_programmirovanie_vostrebovanno (дата обращения: 17.05.2020).
10. Каталог API (Microsoft) и справочных материалов [Электронный ресурс]. URL: https://msdn.microsoft.com/library (дата обращения: 14.05.2020).
11. Руководство по .NET Framework [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/framework/index (дата обращения: 19.05.2020).
12. Руководство по языку C# [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 18.05.2020).
13. Сайт о программировании METANIT.COM [Электронный ресурс]. URL: https://metanit.com/ (дата обращения: 12.05.2020).
14. Самые востребованные IT-навыки в 2020 [Электронный ресурс]. URL: https://alexstaff.agency/ru/demand-skills-2020 (дата обращения: 07.05.2020).
15. Самые высокооплачиваемые профессии в IT-сфере в 2020 [Электронный ресурс]. URL: https://zen.yandex.ru/media/cyberstar/samye-vysokooplachivaemye-professii-v-itsfere-v-2020-5e260e90f73d9d00af63dc91 (дата обращения: 08.05.2020).
16. Типы файлов ASP.NET [Электронный ресурс]. URL: http://wiki.it-wiki.org.ua/doku.php/asp.net_type_file (дата обращения: 13.05.2020).
17. Что выбрать: ASP.NET или PHP [Электронный ресурс]. URL: https://skillbox.ru/media/code/chto_vybrat_asp_net_ili_php/ (дата обращения: 11.05.2020).

ПРИЛОЖЕНИЯ

Таблица 14. Базовые классы веб-элементов управления [1]

Объявление дескриптора ASP.NET	Генерируемая HTML-разметка	Ключевые члены
<asp:Button>	<input type="submit"/> или	Text, CausesValidation, PostBackUrl, ValidationGroup, событие Click
<asp:CheckBox>	☐	AutoPostBack, Checked, Text, TextAlign, событие CheckedChanged
<asp:FileUpload>	<input type="file">	FileBytes, FileContent, FileName, HasFile, PostedFile, SaveAs()
<asp:HiddenField>	<input type="hidden">	Value
<asp:HyperLink>	<a>...	ImageUrl, NavigateUrl, Target, Text
<asp:Image>	<img/>	AlternateText, ImageAlign, ImageUrl
<asp:ImageButton>	<input type="image"/>	CausesValidation, ValidationGroup, событие Click
<asp:ImageMap>	<map>	HotSpotMode, HotSpots (коллекция), AlternateText, ImageAlign, ImageUrl
<asp:Label>	...	Text, AssociatedControlID
<asp:LinkButton>	<a> <img /> 	Text, CausesValidation, ValidationGroup, событие Click
<asp:Panel>	...	BackImageUrl, DefaultButton, GroupingText, HorizontalAlign, Scrollbars, Wrap
<asp:RadioButton>	☐	AutoPostBack, Checked, GroupName, Text, TextAlign, событие CheckedChanged
<asp:Table>	<table>...</table>	BackImageUrl, CellPadding, Cellspacing, GridLines, HorizontalAlign, Rows (коллекция)
<asp:TableCell>	<td>...</td>	ColumnSpan, HorizontalAlign, RowSpan, Text, VerticalAlign, Wrap
<asp:TableRow>	<tr>...</tr>	Cells (коллекция), HorizontalAlign, VerticalAlign
<asp:TextBox>	____ или <textarea>...</textarea>	AutoPostBack, Columns, MaxLength, Readonly, Rows, Text, TextMode, Wrap, событие TextChanged

Таблица 15. Типы столбцов GridView [6]

Столбец	Описание
BoundField	Этот столбец отображает текст поля источника данных
ButtonField	Этот столбец отображает кнопку для каждого значения в списке
CheckBoxField	Этот столбец отображает флажок для каждого элемента в списке Используется автоматически для полей true/false (в SQL Server это поля, использующие битовый тип данных)
CommandField	Этот столбец предоставляет кнопки выделения или редактирования
HyperlinkField	Этот столбец отображает свое содержимое (поле из источника данных или статический текст) как гиперссылку

Столбец	Описание
ImageField	Этот столбец отображает графические данные из двоичного поля (предполагая, что они могут быть успешно интерпретированы как поддерживаемый графический формат)
TemplateField	Этот столбец позволяет указывать множество полей, пользовательские элементы управления и произвольную HTML-разметку, используя специальный шаблон. Предоставляет наибольший контроль, но также требует максимум работы

Таблица 16. Типы столбцов DetailsView [6]

Тип поля столбца	Описание
BoundField	Отображает значение поля в источнике данных в виде текста.
ButtonField	Отображает кнопку команды в элементе управления DetailsView. Это позволяет отобразить строку с пользовательским элементом управления "Кнопка", например, "Добавить" или "Удалить"
CheckBoxField	Отображает флажок в элементе управления DetailsView. Этот тип поля строки обычно используется для вывода полей с логическим значением
CommandField	Отображает встроенные командные кнопки для выполнения операций изменения, вставки или удаления в элементе управления DetailsView
HyperLinkField	Отображает значение поля в источнике данных в виде гиперссылки. Этот тип поля строки позволяет привязать второе поле к URL-адресу гиперссылки
ImageField	Отображает изображение в элементе управления DetailsView
TemplateField	Отображает определяемое пользователем содержимое для строки в элементе управления DetailsView в соответствии с заданным шаблоном. Этот тип поля строки позволяет создать настраиваемое поле строки

Таблица 17. Стили GridView [6]

Описание	Стиль
HeaderStyle	Конфигурирует внешний вид строки заголовка, содержащей заголовки столбцов, если включено их отображение (т.е. ShowHeader установлено в true)
RowStyle	Конфигурирует внешний вид каждой строки данных
AlternatingRowStyle	Если установлен, применяет дополнительное форматирование к каждой второй строке. Это форматирование действует в дополнение к форматированию RowStyle. Например, если установить шрифт с использованием RowStyle, он также применяется для чередующихся строк, если только в AlternatingRowStyle явно не указан другой шрифт
SelectedRowStyle	Конфигурирует внешний вид текущей выбранной строки
EditRowStyle	Конфигурирует внешний вид строки, находящейся в режиме редактирования
EmptyDataRowStyle	Конфигурирует стиль, используемый для одной пустой строки в специальном случае, когда привязанный объект данных не содержит строк
FooterStyle	Конфигурирует внешний вид строки нижнего колонтитула GridView, если включено его отображение (т.е. ShowFooter установлено в true)

Описание	Стиль
PagerStyle	Конфигурирует внешний вид строки со ссылками на страницы, если включено постраничное разбиение (т.е. AllowPaging установлено в true)

Таблица 18. Шаблоны GridView [6]

Шаблон	Описание
HeaderTemplate	Определяет внешний вид и содержимое ячейки заголовка
FooterTemplate	Определяет внешний вид и содержимое ячейки нижнего колонтитула
ItemTemplate	Определяет внешний вид и содержимое каждой ячейки данных (если не используется AlternatingItemTemplate) или каждой нечетной ячейки (если используется)
AlternatingItemTemplate	Используется в сочетании с ItemTemplate для различного форматирования четных и нечетных строк
EditItemTemplate	Определяет внешний вид и элементы управления, используемые в режиме редактирования
InsertItemTemplate	Определяет внешний вид и элементы управления, используемые при вставке новой записи

Таблица 19. Шаблоны ListView [6]

Режим	Описание
ItemTemplate	Устанавливает содержимое каждого элемента данных (если вы не используете AlternatingItemTemplate) или каждой нечетной ячейки (если используете)
AlternatingItemTemplate	Применяется в сочетании с ItemTemplate для различного форматирования четных и нечетных строк
ItemSeparatorTemplate	Устанавливает содержимое разделителя, размещаемого между элементами
SelectedItemTemplate	Устанавливает содержимое элемента, выбранного в данный момент. Можно использовать то же содержимое, что и ItemSeparatorTemplate, но с другим форматированием, или же выбрать отображение расширенного вида с дополнительными деталями для выбранного элемента
EditItemTemplate	Устанавливает элементы управления, используемые для элемента в режиме редактирования
InsertItemTemplate	Устанавливает элементы управления, используемые для вставки нового элемента
LayoutTemplate	Устанавливает разметку, обертывающую ваш список элементов
GroupTemplate	Устанавливает разметку, обертывающую каждую группу элементов, если используется средство группирования
GroupSeparatorTemplate	Устанавливает содержимое разделителя групп элементов
EmptyItemTemplate	Устанавливает содержимое, используемое для заполнения пустых значений в последней группе, если применяется группирование. Например, если создаются группы из 5 элементов, а источником данных является коллекция из 13 объектов, то в последней группе будет не хватать 2 элементов
EmptyDataTemplate	Устанавливает разметку, используемую в случае пустого привязанного объекта данных (т.е. не содержащего записей или объектов)

Таблица 20. Шаблоны FormView [6]

Тип шаблона	Описание
EditItemTemplate	Определяет содержимое для строки данных, когда элемент управления FormView находится в режиме редактирования. Этот шаблон обычно содержит элементы управления вводом и командные кнопки, с помощью которых пользователь может изменять существующую запись
EmptyDataTemplate	Определяет содержимое для пустой строки данных, отображаемой, когда FormView элемент управления привязан к источнику данных, который не содержит записей. Этот шаблон обычно содержит содержимое, чтобы предупредить пользователя о том, что источник данных не содержит записей
FooterTemplate	Определяет содержимое для строки нижнего колонтитула.Этот шаблон обычно содержит любое дополнительное содержимое, которое вы хотите отобразить в строке нижнего колонтитула. **Примечание.** В качестве альтернативы можно просто указать текст, отображаемый в строке нижнего колонтитула, задав свойство FooterText
HeaderTemplate	Определяет содержимое для строки заголовка. Этот шаблон обычно содержит любое дополнительное содержимое, которое вы хотите отобразить в строке заголовка. **Примечание.** В качестве альтернативы можно просто указать текст, отображаемый в строке заголовка, задав свойство HeaderText
ItemTemplate	Определяет содержимое для строки данных, если элемент управления FormView находится в режиме только для чтения.Этот шаблон обычно содержит содержимое для вывода значений существующей записи
InsertItemTemplate	Определяет содержимое для строки данных, когда элемент управления FormView находится в режиме вставки. Этот шаблон обычно содержит элементы управления вводом и командные кнопки, с помощью которых пользователь может добавить новую запись
PagerTemplate	Определяет содержимое строки пейджера, отображаемой при включенной функции разбиения по страницам (если свойство AllowPaging имеет значение true). Этот шаблон обычно содержит элементы управления, с помощью которых пользователь может переходить к другой записи. **Примечание.** Элемент управления FormView имеет встроенный пользовательский интерфейс строки пейджера. Шаблон пейджера необходимо создать только в том случае, если вы хотите создать собственную настраиваемую строку пейджера

Таблица 21. Кнопки и функции, распознаваемые FormView [6]

Кнопка	Описание
Отмена	Используется в операциях обновления или вставки для отмены операции и для отмены значений, вводимых пользователем. Затем элемент управления FormView возвращается в режим, заданный свойством DefaultMode
Удаление	Используется в операциях удаления для удаления отображаемой записи из источника данных. Вызывает события ItemDeleting и ItemDeleted

Кнопка	Описание
Правка	Используется в операциях обновления для помещения элемента управления FormView в режим редактирования. Содержимое, указанное в свойстве EditItemTemplate, отображается для строки данных
Вставить	Используется в операциях вставки, чтобы попытаться вставить новую запись в источник данных, используя значения, предоставленные пользователем. Вызывает события ItemInserting и ItemInserted
Создать	Используется в операциях вставки для помещения элемента управления FormView в режим вставки. Содержимое, указанное в свойстве InsertItemTemplate, отображается для строки данных
Страница	Используется в операциях подкачки для представления кнопки в строке пейджера, которая выполняет разбиение на страницы.Чтобы указать операцию разбиения по страницам, задайте для свойства CommandArgumentкнопки значение "Далее", "назад", "первый", "последний" или индекс страницы, к которой осуществляется переход. Вызывает события PageIndexChanging и PageIndexChanged. **Примечание.** Этот тип кнопки обычно используется только в шаблоне страничного навигатора
Обновить	Используется в операциях обновления, чтобы попытаться обновить отображаемую запись в источнике данных значениями, предоставленными пользователем. Вызывает события ItemUpdating и ItemUpdated

Таблица 22. Шаблоны ListView [6]

Тип шаблона	Описание
LayoutTemplate	Корневой шаблон, определяющий объект-контейнер, например, table, div или элемент span, который будет содержать содержимое, определенное в шаблоне ItemTemplate или GroupTemplate. Он также может содержать объект DataPager
ItemTemplate	Определяет содержимое, привязанное к данным, которое будет отображаться для отдельных элементов
ItemSeparatorTemplate	Определяет содержимое для отображения между отдельными элементами
GroupTemplate	Определяет объект-контейнер, например, строку таблицы (tr), div или элемент span, который будет содержать содержимое, определенное в шаблонах ItemTemplate и EmptyItemTemplate. Количество элементов, отображаемых в группе, задается свойством GroupItemCount
GroupSeparatorTemplate	Определяет содержимое для отображения между группами элементов
EmptyItemTemplate	Определяет содержимое, которое будет отображаться для пустого элемента при использовании шаблона GroupTemplate. Например, если свойство GroupItemCount имеет значение 5, а общее число элементов, возвращаемых из источника данных, равно 8, последняя группа данных, отображаемая элементом управления ListView, будет содержать три элемента, указанные в шаблоне ItemTemplate, и два элемента, как указано в шаблоне EmptyItemTemplate
EmptyDataTemplate	Определяет содержимое для отображения, если источник данных не возвращает данные

Тип шаблона	Описание
SelectedItemTemplate	Определяет содержимое, отображаемое для выбранного элемента данных, чтобы отличать выбранный элемент от других элементов
AlternatingItemTemplate	Определяет содержимое, отображаемое для чередующихся элементов, чтобы облегчить различение последовательных элементов
EditItemTemplate	Определяет содержимое, отображаемое при редактировании элемента. Шаблон EditItemTemplate отображается вместо шаблона ItemTemplate для редактируемого элемента данных
InsertItemTemplate	Определяет содержимое, которое будет отображено для вставки элемента. Шаблон InsertItemTemplate отображается вместо шаблона ItemTemplate в начале или в конце элементов, отображаемых элементом управления ListView. Вы можете указать, где будет подготавливаться шаблон InsertItemTemplate, используя свойство InsertItemPosition элемента управления ListView

Таблица 23. Кнопки и функции, распознаваемые ListView [6]

Кнопка	Описание
Отмена	Отменяет операцию изменения или вставки.Порождает событие ItemCanceling
Удаление	Удаляет текущую запись из источника данных. Вызывает события ItemDeleted и ItemDeleting
Выбрать	Задает для свойства SelectedIndex значение свойства DisplayIndex элемента.Подготавливает к просмотру шаблон SelectedItemTemplate для элемента. Вызывает события SelectedIndexChanging и SelectedIndexChanged
Правка	Помещает элемент в режим редактирования.Подготавливает к просмотру шаблон EditItemTemplate для элемента. Порождает событие ItemEditing
Вставить	Вставляет привязанные значения из шаблона InsertItemTemplate в источник данных. Вызывает события ItemInserting и ItemInserted
Обновить	Обновляет текущую запись в источнике данных связанными значениями из шаблона EditItemTemplate. Вызывает события ItemUpdating и ItemUpdated
Сортировать	Сортирует столбцы, перечисленные в свойстве CommandArgument кнопки. Вызывает события Sorting и Sorted

Учебное издание

С.Т. Гуляева, В.В. Миронов, Н.О. Котелина

ASP.NET Web Forms в задачах и примерах

Учебное пособие

Выполнено с использованием программы MS Office Word

Системные требования:
ПК не ниже класса Pentium III; 256 Мб RAM; не менее 1,5 Гб на винчестере; Windows XP с пакетом обновления 2 (SP2); Microsoft Office 2003 и выше; видеокарта с памятью не менее 32 Мб; экран с разрешением не менее 1024 × 768 точек; 4-скоростной дисковод (CD-ROM) и выше; мышь.

Редактор *Е.М. Насирова*
Техническое редактирование – *Д.В. Богачук,* преподаватель кафедры радиофизики и электроники Института точных наук и информационных технологий СГУ им. Питирима Сорокина
Выпускающий редактор *Л.Н. Руденко*

7.08 Мб. 1 компакт-диск, пластиковый бокс, вкладыш.
Подписано к использованию 07.12.2020 г. Заказ № 136. Тираж 100 экз.
Издательский центр СГУ им. Питирима Сорокина
167982, Сыктывкар, ул. Коммунистическая, 23Б
Тел. (8212)390-472, 390-473.

e-mail: ipo@syktsu.ru
http://www.syktsu.ru

Титул Об издании Производственно-технические сведения Оглавление