Лабораторная работа № 4

Создание Web-интерфейса для работы с распределённой базой данных

Цель работы

Получить навыки создания Web-интерфейса для работы с распределённой базой данных. Научиться применять технологию AJAX для обмена данными между сервером и Web-страницей.

Создание простого ASP.NET приложения

Сначала нужно запустить Microsoft Visual Studio и выбрать создание нового Web-сайта (см. рисунок 1).

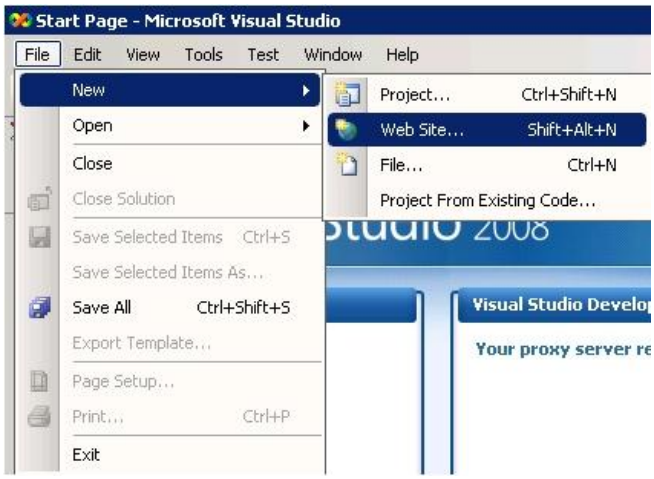

Рисунок 1 – Создание нового проекта

Откроется диалоговое окно (см. рисунок 2), в котором необходимо указать следующее:

1. Шаблон проекта – «ASP.NET Web Site».
2. Location – любой каталог на общедоступном ресурсе, например «**`D:\MyWebSite`**».
3. Язык программирования – возможен выбор любого из предложенных языков, рекомендуется Visual C# (см. рисунок 3).

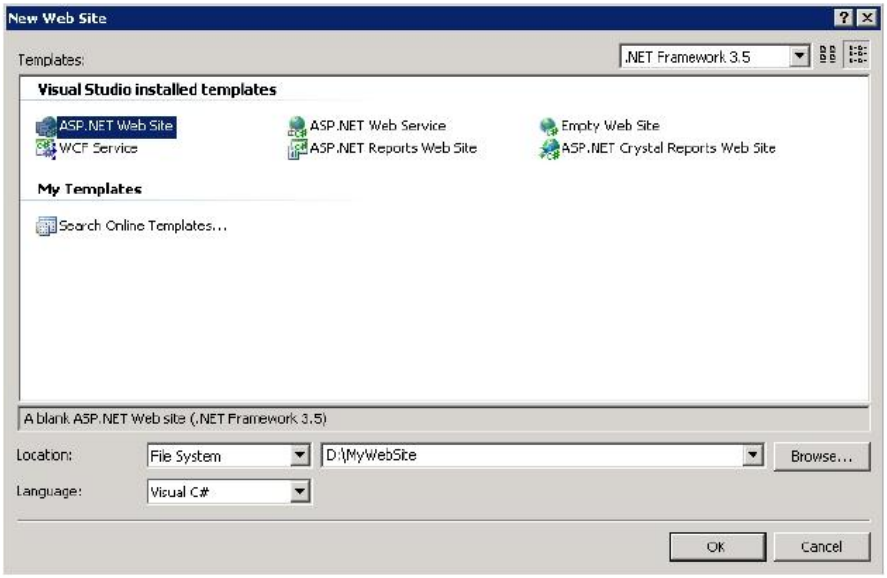

Рисунок 2 – Выбор шаблона проекта

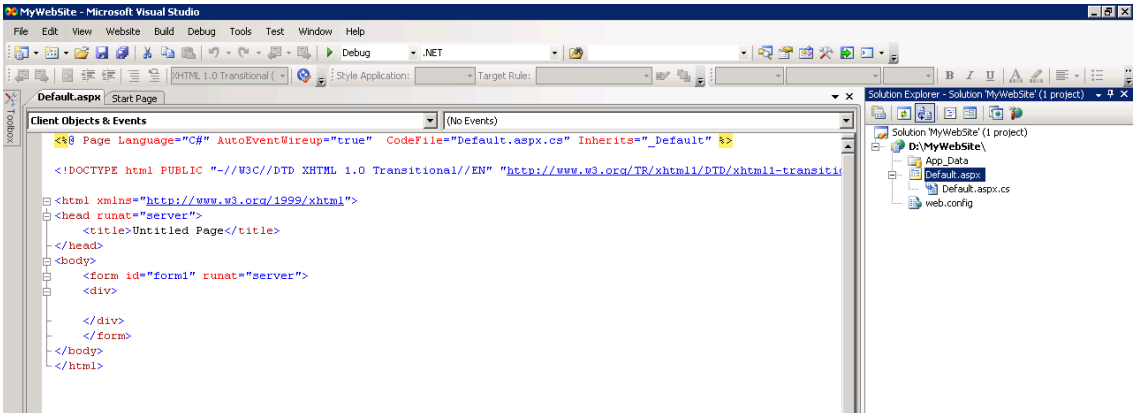

Рисунок 3 – Код нового проекта

В меню «**Solution Explorer**» (см. рисунок 4) описывается структура созданного сайта, который состоит:

- из страницы «**Default**»;
- из конфигурационного файла **web.config**.

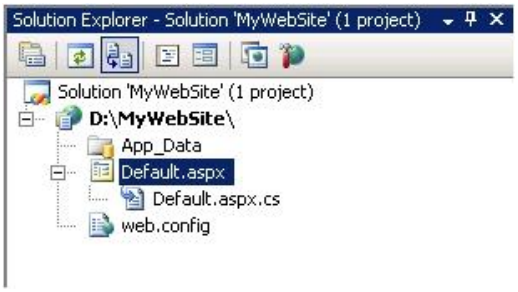

Рисунок 4 – Меню «Solution Explorer»

Концепция разработки ASP.NET чётко разделяет представление (внешний вид) и логику (обработчики событий) приложения: представление страницы находится в файле с расширением **aspx**, а код – в файле с расширением **aspx.cs**.

Теперь созданное приложение представляет собой пустую Web-страницу.

Чтобы добавить на неё несколько элементов, нужно открыть файл **Default.aspx** в режиме **Design**. Теперь можно перенести любой компонент с панели **Toolbox** на страницу (см. рисунки 5 и 6). Перетащив компоненты **Label** (метка), **TextBox** (поле для ввода текста) и **Button** (кнопка), можно получить примерно следующую страницу (см. рисунок 7).

Рисунок 5 – Панель элементов

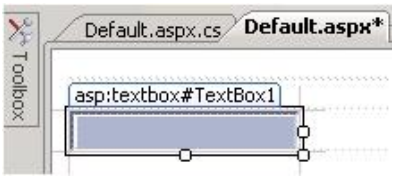

Рисунок 6 – Текстовое поле TextBox

Рисунок 7 – Основные компоненты формы

Настроить параметры каждого компонента можно, щёлкнув по нему правой кнопкой мыши и выбрав меню **Properties**. В появившемся окне можно задать все возможные параметры компонента (см. рисунок 8). После задания настроек компонентов получается страница следующего вида (см. рисунок 9):

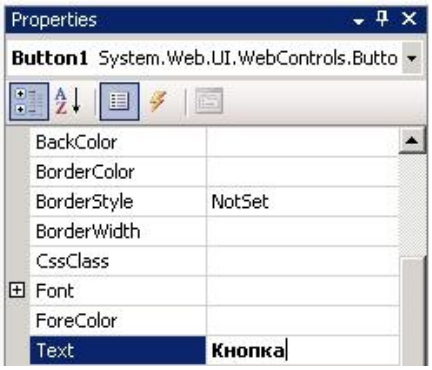

Рисунок 8 – Свойства кнопки

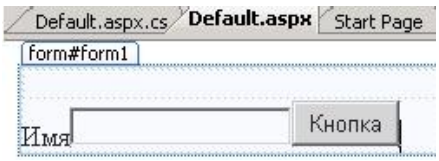

Рисунок 9 – Итоговая форма

На этом создание внешнего вида страницы можно считать завершённым. Теперь можно посмотреть получившийся результат, нажав на кнопку **F5** (**Debug → Start Debugging**). При этом в нижнем правом углу экрана (в списке запущенных программ) должен появиться следующий значок (см. рисунок 10). Этот значок свидетельствует о том, что на компьютере был запущен Web-сервер с созданным Web-приложением. Кроме того, после нажатия кнопки **F5** будет открыт браузер, в котором уже будет набран URL для доступа к созданному приложению (см. рисунок 11).

Рисунок 10 – Значок запущенного приложения

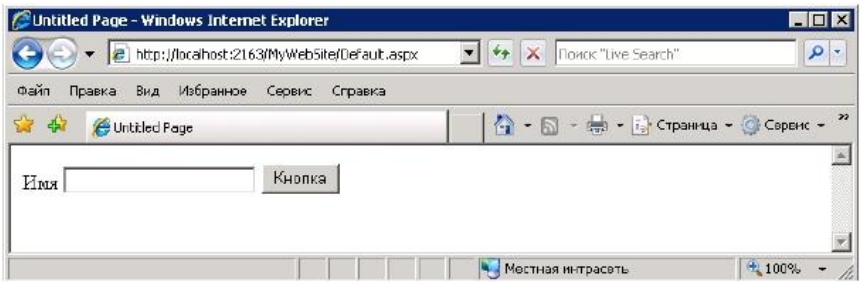

Рисунок 11 – Работающее приложение

Теперь можно добавить обработку нажатия на кнопку. Для этого нужно сделать двойной щелчок на созданной кнопке. Это приведёт к открытию файла **Default.aspx.cs** и созданию тела функции-обработчика нажатия на эту кнопку (см. рисунок 12).

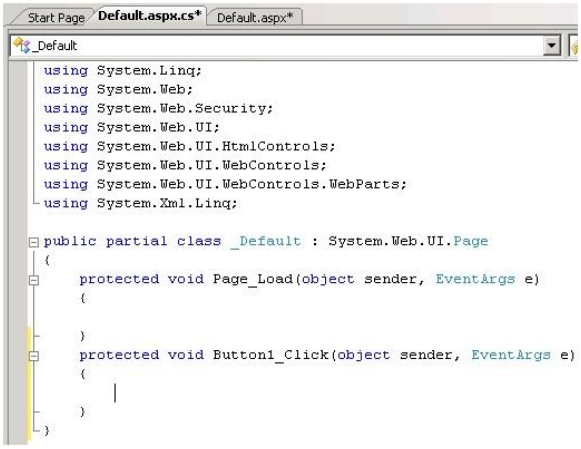

Рисунок 12 – Код приложения на языке C#

В тело функции можно добавить код, подставляющий в начало поля ввода текста **TextBox1** строку «**Привет,** »:

```
protected void Button1_Click(object sender, EventArgs e)
{
 TextBox1.Text = "Привет, " + TextBox1.Text;
}
```

Если запустить созданное приложение и набрать в поле ввода текст «Студент», то получится следующее (см. рисунок 13):

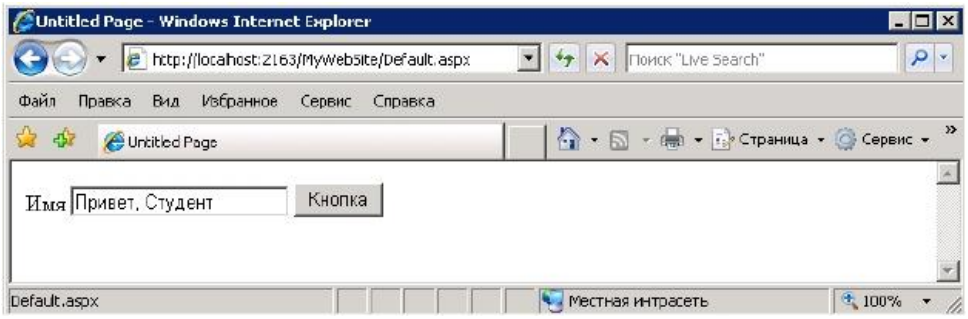

Рисунок 13 – Результат нажатия кнопки

Получившийся пример является своего рода программой «Hello, world» для технологии ASP.NET.

Работа с СУБД

Подключение к базе данных напрямую из кода приложения зачастую невозможно. Причина этого заключается в том, что язык программирования не может поддерживать протоколы работы СУБД всех производителей из-за их большого количества. Возможное (но не единственное) решение данной проблемы заключается в использовании технологий-«адаптеров», позволяющих абстрагироваться от конкретной версии используемой СУБД посредством предоставления интерфейса для работы с общим для всех версий СУБД функционалом. Конечно, такой подход имеет и отрицательные стороны: издержки времени на работу «адаптеров», невозможность использовать уникальный функционал конкретной версии СУБД и так далее. Однако часто эти неудобства являются приемлемой платой за независимость представления от типа СУБД.

Создание источника данных

На текущий момент существует довольно большое количество технологий-«адаптеров». Выбор той или иной технологии зачастую зависит от языка программирования, на котором разрабатывается приложение (для языка Java преимущественно используется технология JDBC), и версии самой СУБД (для СУБД MSSQL преимущественно используется технология ADO[.NET]).

В настоящих методических указаниях рассматривается технология ODBC. Эта технология, в общем случае, не самая быстрая для работы с большинством СУБД. Однако она поддерживает очень большой процент существующих на данный момент СУБД и, что очень важно, проверена временем.

В отличие от операционных систем семейства *NIX, где настройка параметров ODBC происходит с помощью файла **`.odbc.ini`**, операционная система Windows осуществляет настройку средствами системных утилит. Для этого нужно выполнить следующие действия:

1. Прежде всего необходимо, чтобы в системе был установлен драйвер ODBC для используемой СУБД (в нашем случае для PostgreSQL). Этот драйвер можно найти на официальном сайте [1].
2. Теперь необходимо зарегистрировать источник данных в системе. Для этого нужно открыть меню **`Control Panel`** → **`Performance and Maintenance`** → **`Administrative Tools`** → **`Data Sources (ODBC)`** (путь приведён для операционной системы Windows XP). В появившемся окне нужно открыть вкладку **`Drivers`** и убедиться в наличии установленного драйвера ODBC для СУБД PostgreSQL (см. рисунок 14):

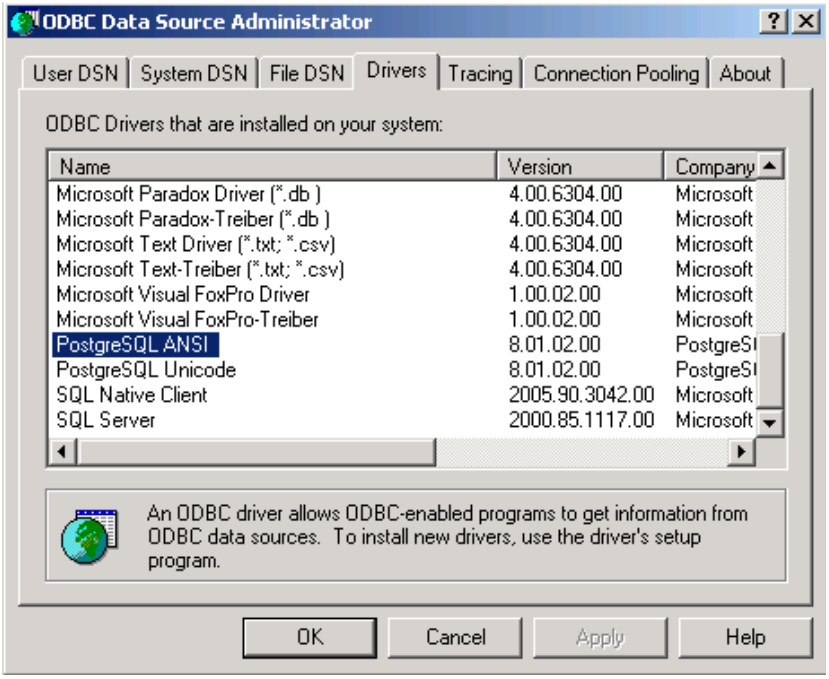

Рисунок 14 – Список драйверов ODBC

3. Выбрать вкладку **`User DSN`**, на которой будет представлен список доступных источников данных (см. рисунок 15):

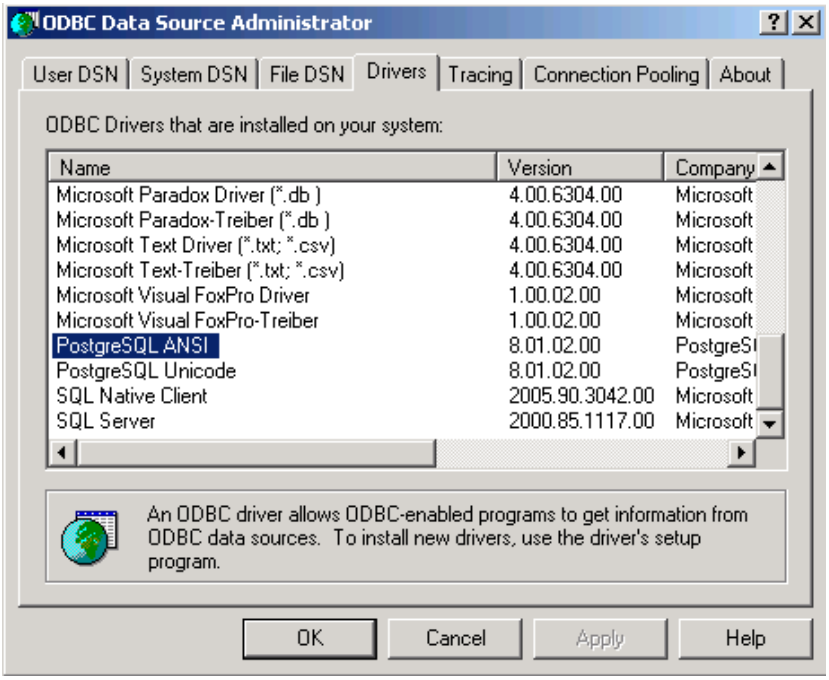

Рисунок 15 – Список доступных источников данных

4. Для того чтобы добавить новый источник данных, следует нажать кнопку **`Add`** и выбрать драйвер СУБД «**`PostgreSQL ANSI`**» (см. рисунок 16):

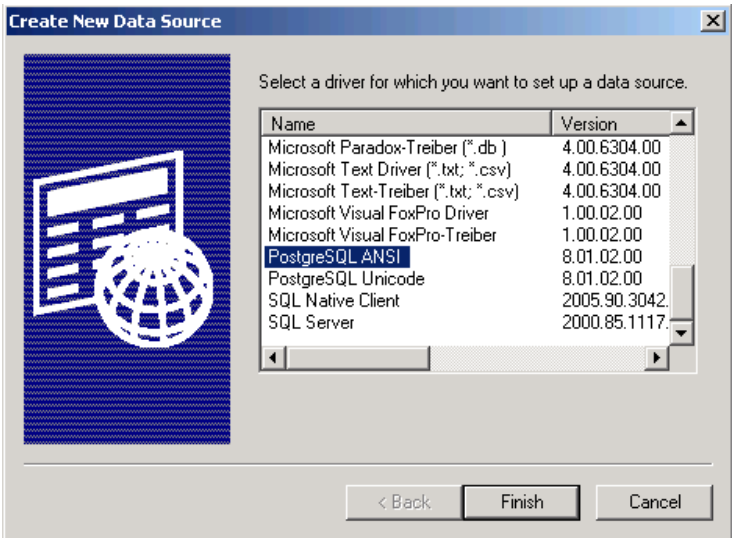

Рисунок 16 – Новый источник данных

5. Теперь осталось лишь указать настройки соединения и нажать кнопку **Save** (см. рисунок 17):

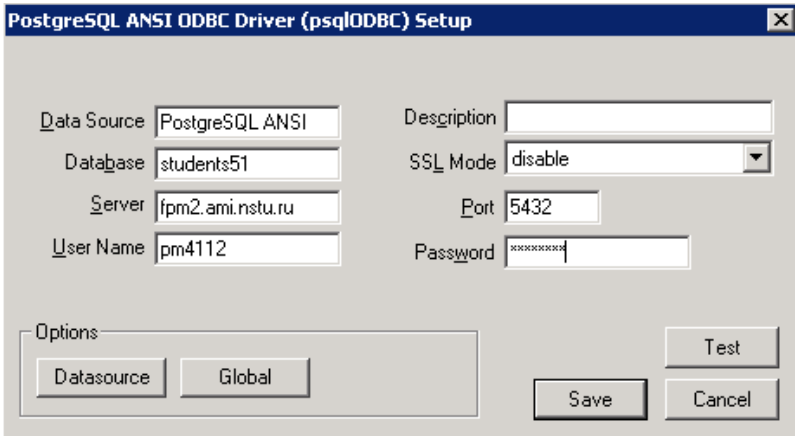

Рисунок 17 – Настройки соединения

6. Источник данных создан и готов к использованию локальными приложениями.

Стоит заметить, что приведённые выше действия необходимы лишь для удобства при подключении к базе данных через ODBC из приложения: теперь в качестве параметра подключения будет достаточно лишь ввести имя созданного источника данных – заданное значение поля **Data Source**, показанного на рисунке 17 (**PostgreSQL ANSI**). В качестве альтернативы можно было бы описать все настроенные параметры подключения прямо в коде приложения.

Подключение к БД напрямую через ODBC

Для создания программного соединения с базой данных с использованием технологии ODBC можно использовать класс **System.Data.Odbc.OdbcConnection**. Параметры соединения задаются с помощью публичного атрибута класса **Connection.String**. Пример строки соединения с использованием источника данных:

```
oConn.ConnectionString = "Dsn=PostgreSQL ANSI";
```

Пример строки соединения без использования источника данных (в минимальном виде):

```
oConn.ConnectionString = "DRIVER=PostgreSQL ANSI; SERVER=<Сервер>; PORT=5432;
  DATABASE=<База данных>; UID=<Имя пользователя>; PWD=<Сервер>";
```

Подключение с использованием компонента Data:SqlDataSource

В качестве альтернативного варианта подключения к базе данных можно использовать компонент **Data:SqlDataSource** (см. рисунок 18):

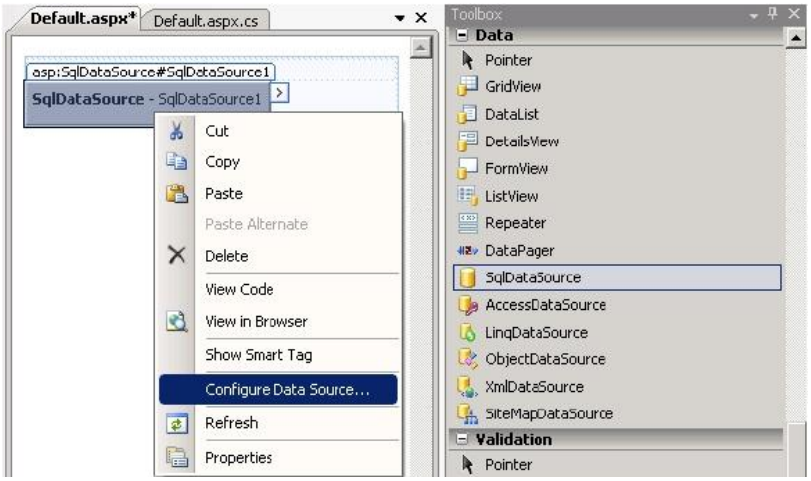

Рисунок 18 – Компонент SqlDataSource

После перетаскивания компонента на страницу нужно выбрать позицию «**Configure Data Source**» и указать в качестве ресурса созданный выше источник данных (см. рисунок 19):

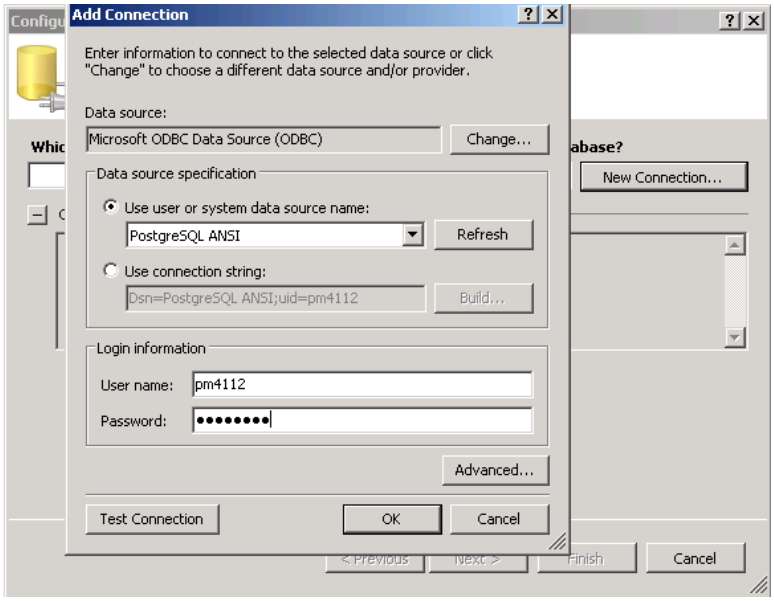

Рисунок 19 – Настройка подключения SqlDataSource

Далее нужно сохранить сформированную строку подключения (DSN-строку) и оформить SQL-запрос, на основе которого компонент **Data:SqlDataSource** будет формировать данные (см. рисунки 20, 21, 22)

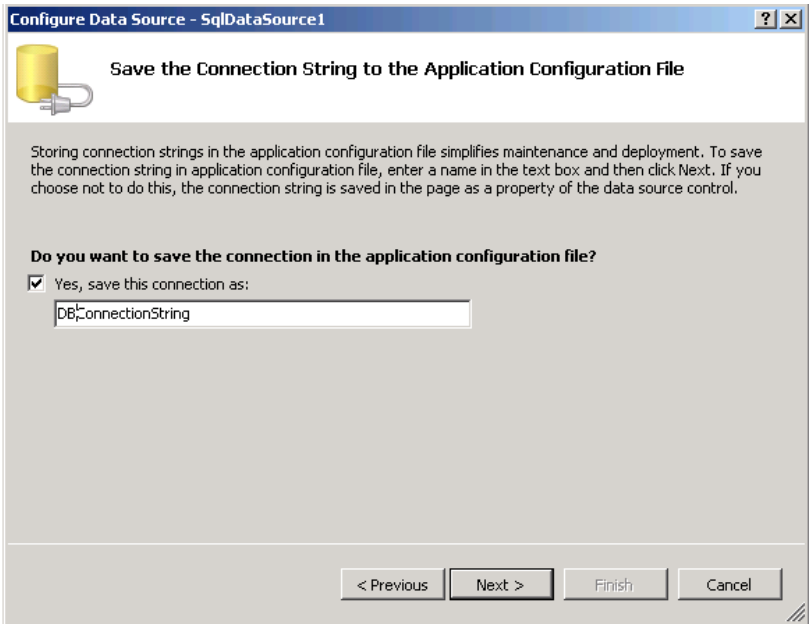

Рисунок 20 – Ввод названия строки подключения

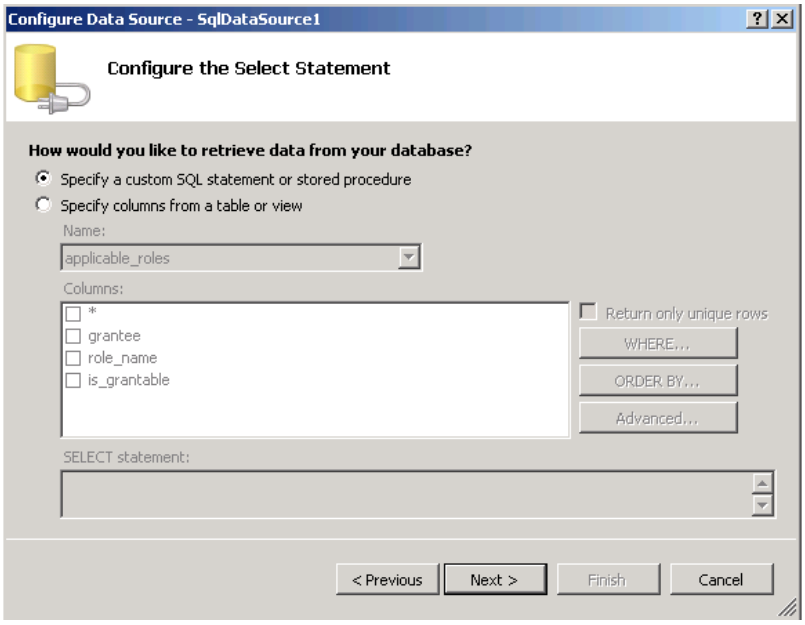

Рисунок 21 – Выбор способа формирования запросов

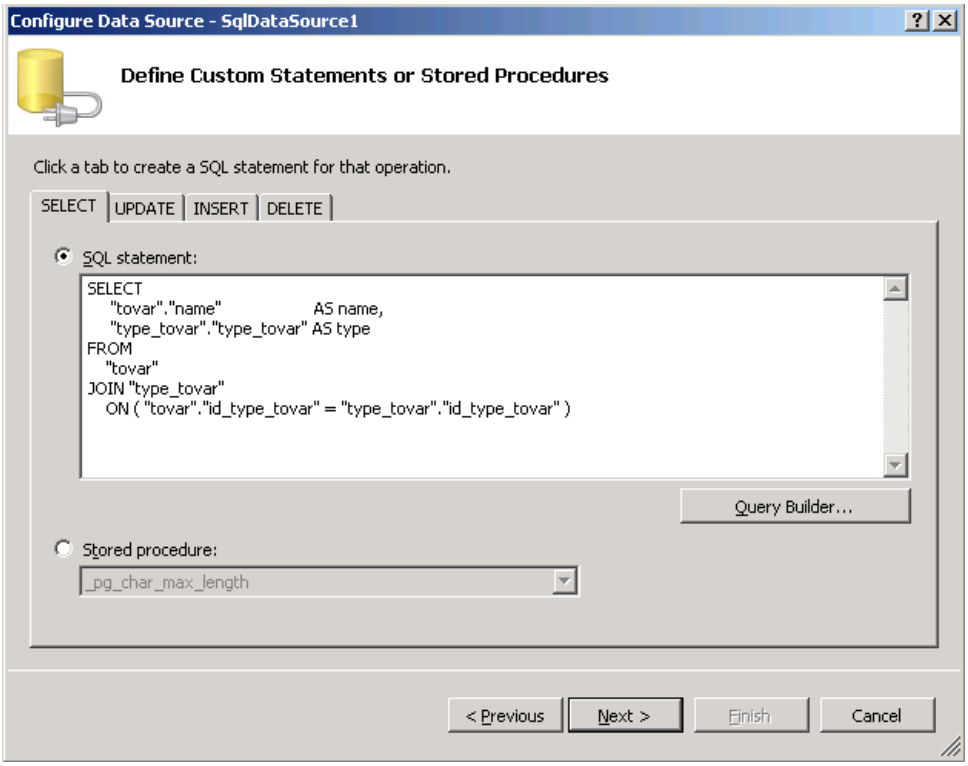

Рисунок 22 – Пример запроса Select

Компонент **`Data:SqlDataSource`** готов к использованию в качестве источника данных. Описание его использования будет приведено далее.

Выбор способа подключения

Стоит заметить, что последний способ подключения через компонент **`Data:SqlDataSource`**, по сути, является «обёрткой» для подключения напрямую через ODBC без использования источника данных. Основным отличием в этом случае будет тот факт, что подключение напрямую через ODBC можно использовать в коде написанных функций для работы с СУБД, в то время как подключение через компонент **`Data:SqlDataSource`** используется встроенными компонентами ASP.NET, например такими, как **`Data:GridView`**.

Ещё один немаловажный вопрос – это необходимость использовать источник данных. Его использование, безусловно, является большим плюсом, т.к. позволяет разработчику полностью абстрагироваться от типа СУБД. Однако стоит понимать, что источник данных – лишь часть конфигурации операционной системы. Для разработчика это означает, что он должен позаботиться о создании такового на компьютере пользователя. Обычно для этого используются программы-установщики, но, поскольку Web-сайты не являются обычным приложением, такой подход к ним

малоприменим. Поэтому рекомендуется для соединения с базой данных использовать подключение через ODBC без источника данных.

Выборка данных

Для вывода данных из базы данных на страницу используется компонент **Data:GridView** (см. рисунок 23).

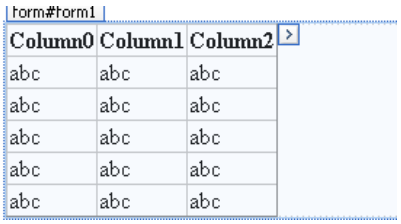

Рисунок 23 – Компонент Data:GridView

Чтобы для компонента **Data:GridView** указать источник данных, нужно нажать на стрелку в правой верхней части рисунка 23, выбрать пункт **Выбрать источник данных** и указать в нём созданный ранее **Data:SqlDataSource** (см. рисунок 24).

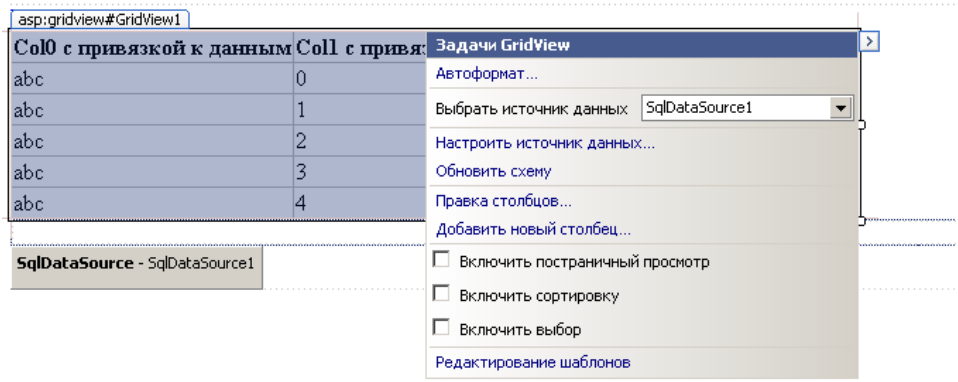

Рисунок 24 – Диалоговое меню компонента Data:GridView

После этого появится возможность указать, будут ли возможны сортировка по столбцам, постраничная навигация, выбор конкретной строки. Также имеется возможность переименования столбцов и редактирования шаблона таблицы.

Выполнение запросов

После того как соединение с базой данных установлено, можно приступать к выполнению запросов. Стоит отметить, что запросы к базе данных можно разделить на две основные группы: запросы выборки и запросы модификации данных. Для каждого из этих типов существуют свои классы исполнения запросов: класс **System.Data.Odbc.OdbcDataAdapter** с методом **Fill** для выборки данных и класс **System.Data.Odbc.OdbcCommand** с методом **ExecuteNonQuery** для модификации данных.

Шаблоном для выполнения запроса любого типа может служить следующий код:

```
System.Data.Odbc.OdbcConnection oConn = new System.Data.Odbc.OdbcConnection();
oConn.ConnectionString = "<Строка подключения>";
try
{
 // Открывается подключение
 oConn.Open();

 /*
    Здесь выполняется запрос
 */

 // Закрывается подключение
 oConn.Close();
}
```

```
catch (System.InvalidOperationException)
{
 ErrorText.Text = "Ошибка запроса к БД";
}
catch (System.Data.Odbc.OdbcException)
{
 ErrorText.Text = "Ошибка соединения с БД";
}
finally
{
 oConn.Close();
}
```

Выборка данных

Для выборки данных достаточно при создании **SqlDataSource** на этапе, приведённом на рисунке 22, указать запрос, который будет выполнять выборку данных.

Редактирование данных

После нажатия на стрелку в правой верхней части рисунка 23 нужно выбрать пункт появившегося меню **Правка столбцов**, выделить каждый столбец и щёлкнуть на ссылку **Преобразовать это поле в TemplateField** (см. рисунок 25).

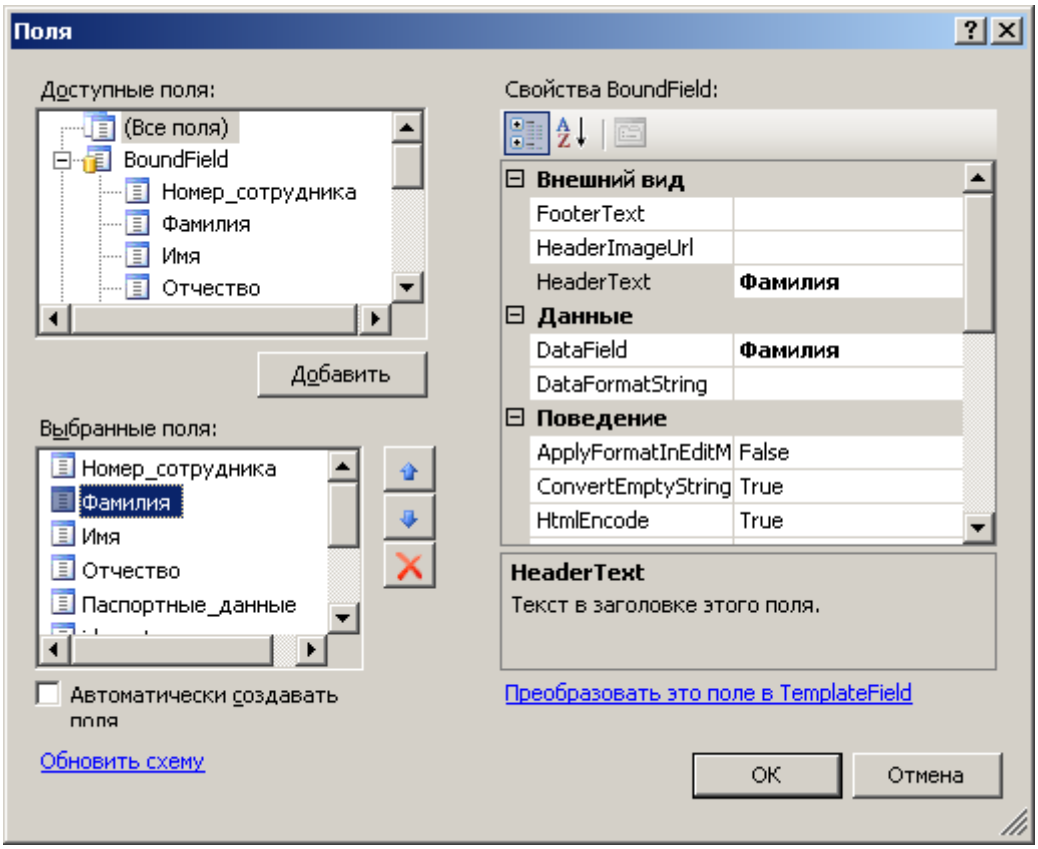

Рисунок 25 – Меню «Правка столбцов»

Далее нужно зайти в редактирование шаблонов. При этом появится следующее окно (см. рисунок 26):

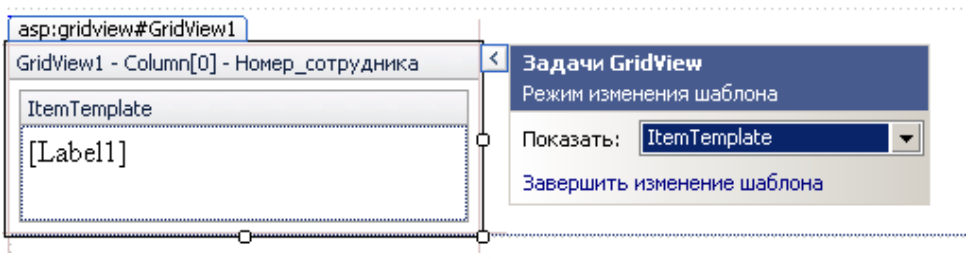

Рисунок 26 – Редактирование шаблона для поля Номер_сотрудника

В списке «**Показать**» следует выбрать нужный столбец и указать **EditItemTemplate**, который будет отображаться при редактировании записи (см. рисунок 27). По умолчанию ему соответствует стандартный компонент **Label**. Для **id** (как и в данном примере) изменять его не рекомендуется для того, чтобы пользователь не мог изменять идентификатор записи в таблице.

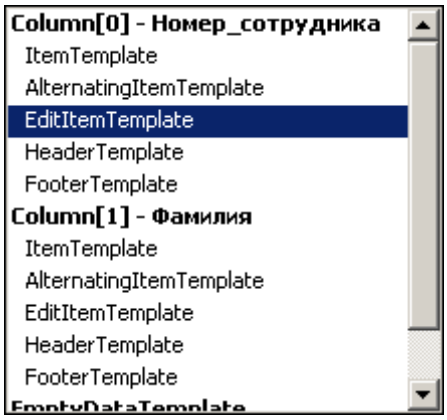

Рисунок 27 – Список столбцов, доступных для редактирования

Для поля **Фамилия** для удобства работы с данными следует изменить этот компонент на выпадающий список **DropDownList**, чтобы можно было выбирать фамилии только существующих сотрудников. Чтобы связать этот список с источником данных, поместим сюда **SqlDataSource** (см. рисунок 28).

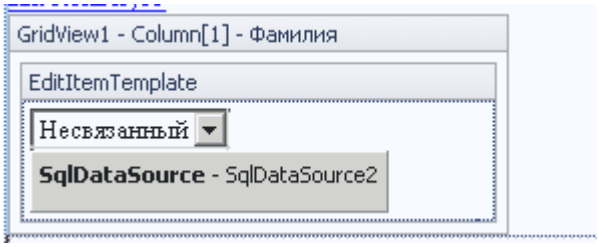

Рисунок 28 – Редактирование шаблона для поля Фамилия

Настройка **SqlDataSource** подробно была описана выше. Здесь достаточно лишь уточнить, что запрос на выборку данных получает **id** (идентификатор записи) и **surname** (поле с фамилиями, которое необходимо вывести в список). Далее нужно связать список с созданным источником данных, как это было сделано для **GridView**. После нажатия **Настроить источник данных** появляется окно, в котором требуется выбрать нужный источник данных (**SqlDataSource2**) и поля данных, которые являются отображаемым элементом списка и соответствующим ему значением (поле данных для отображения используется для вывода информации пользователю (фамилии), а поле значений используется для хранения соответствующих идентификаторов выведенных записей, которые не видны пользователю (**id**)). Выбор нужных полей приведён на рисунке 29:

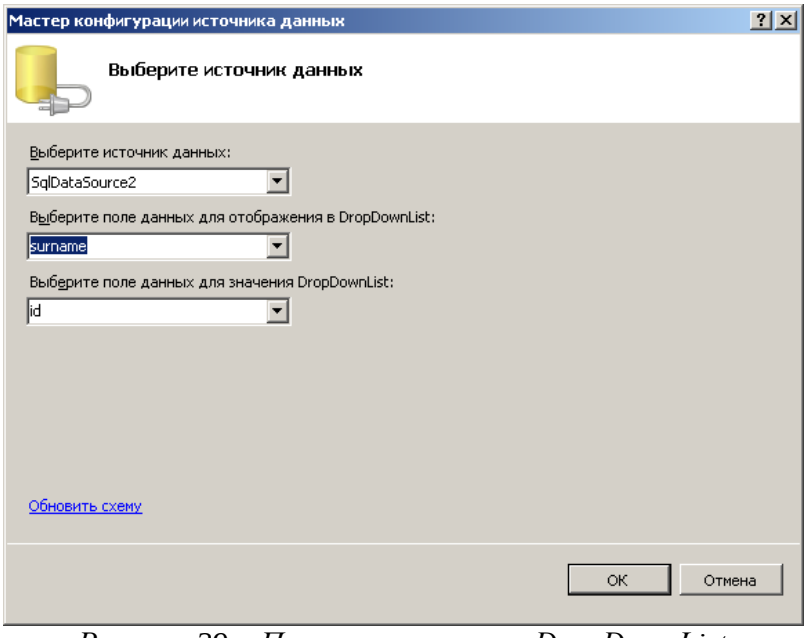

Рисунок 29 – Параметры списка DropDownList

Чтобы первым элементом в списке отображалось текущее значение редактируемого поля (фамилии), нужно связать список с соответствующим полем из **GridView**. Делается это следующим образом: нужно нажать на стрелку в правом верхнем углу списка и выбрать пункт **Правка DataBindings** (см. рисунок 30).

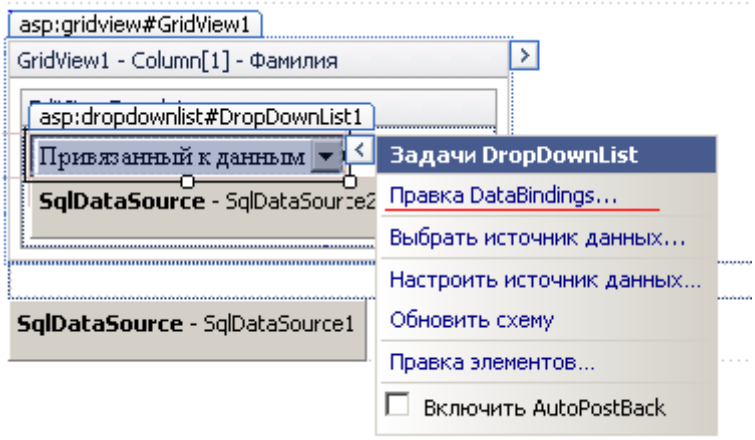

Рисунок 30 – Связывание списка с полем из GridView

Далее свойство **SelectedValue** нужно привязать к полю **Фамилия** следующим образом (см. рисунок 31).

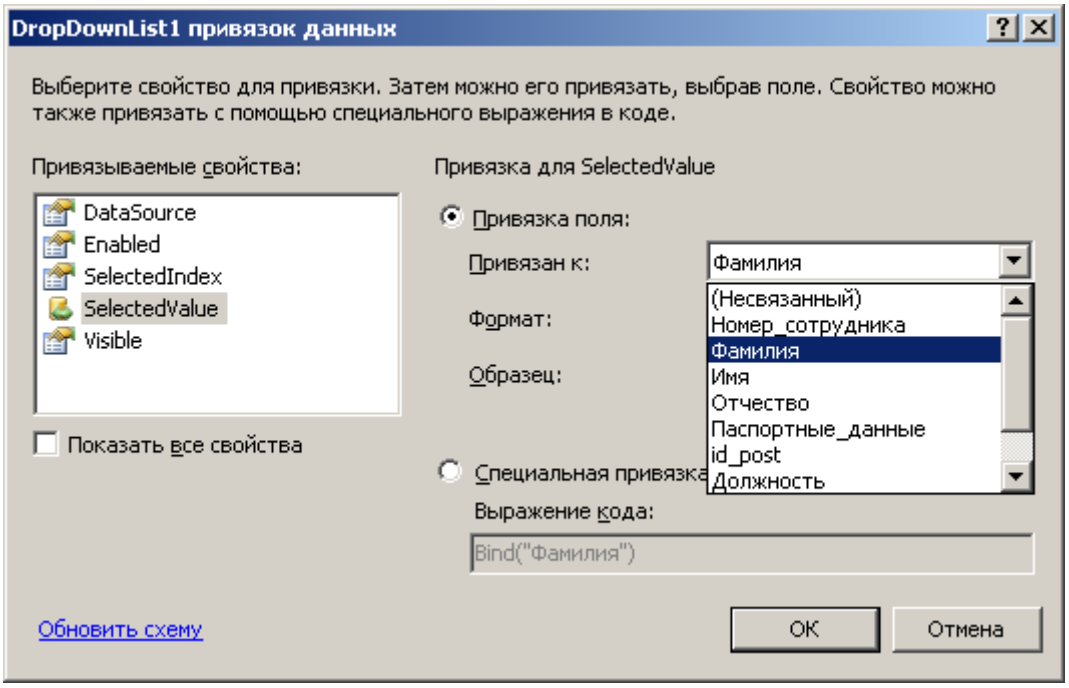

Рисунок 31 – Привязка к поля Фамилия

Теперь редактирование шаблона завершено, для выхода и применения изменений необходимо нажать **Завершить изменение шаблона** (см. рисунок 32).

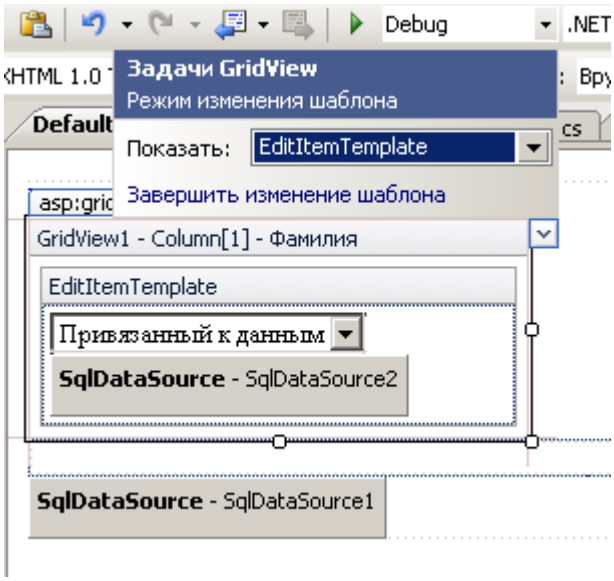

Рисунок 32 – Завершение изменения шаблона

Теперь для каждой строки необходимо добавить кнопки модификации (**Правка**, **Удалить**). Для этого свойства **AutoGenerateDeleteButton** и **AutoGenerateEditButton** объекта **GridView** нужно выставить в **true**. В результате **GridView** примет вид, показанный на рисунке 33:

asp:gridview#GridView1

	Номер_сотрудника	Фамилия	Имя	Отчество	Паспор
Правка Удалить	0	abc	abc	abc	abc
Правка Удалить	1	abc	abc	abc	abc
Правка Удалить	2	abc	abc	abc	abc
Правка Удалить	3	abc	abc	abc	abc
Правка Удалить	4	abc	abc	abc	abc

Рисунок 33 – Настроенный GridView

Осталось обработать нажатие кнопки **Правка**. Для этого нужно зайти в события (**Events**) компонента **GridView** и на строке **RowEditing** сделать двойной щелчок левой кнопкой мыши (см. рисунок 34):

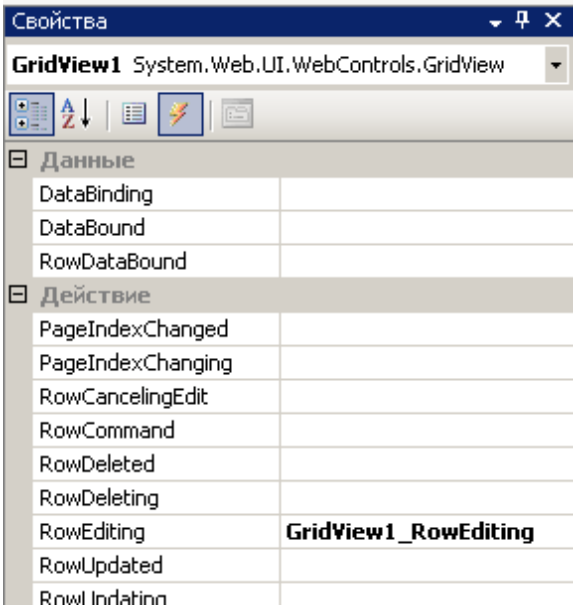

Рисунок 34 – Свойства GridView

Появится следующий код обработчика:

```
protected void GridView1_RowEditing(object sender,
                                    GridViewEditEventArgs e)
{
}
```

В обработчик следует поместить код:

```
// Выделение выбранной строки
GridView1.EditIndex = e.NewEditIndex;
```

Чтобы обработать нажатие кнопки **Обновить**, нужно зайти в события (**Events**) компонента **GridView** и на строке **RowUpdating** сделать двойной щелчок левой кнопкой мыши. В появившийся обработчик следует вставить следующий код:

```
protected void GridView1_RowUpdating(object sender,
                                     GridViewUpdateEventArgs e)
{
 // Создание объекта типа строка GridView
 // и присваивание ему выбранной строки
 GridViewRow row = GridView1.Rows[GridView1.EditIndex];

 // Получение id, выбранного из списка сотрудника
 string id_work = ((DropDownList) row.FindControl("DropDownList1")
                  as DropDownList).SelectedValue.ToString();
 // Получение id выбранной записи
 string ID = ((Label) row.FindControl("Label3") as Label).Text;
 // Команда обновления источника данных SqlDataSource
 SqlDataSource1.UpdateCommand =
   "select public.dblink_connect('con1',
      'dbname=students52 user=pm**** password=********');"
   + "select public.dblink_exec('con1',
       'update photo52_2.sale set id_worker=" + id_work
   + " where id=" + ID + "');"
   + "select public.dblink_disconnect('con1');";
 // Выполнение команды
 SqlDataSource1.Update();
}
```

Удаление записи

Для удаления выбранной записи необходимо обработать событие **RowDeleting**:

```
protected void GridView1_RowDeleting(object sender,
                                     GridViewDeleteEventArgs e)
{
 // Создание объекта типа строка GridView
 // и присваивание ему выбранной строки
 GridViewRow row = GridView1.Rows[e.RowIndex];
 // Получение id выбранной записи
 string ID = ((Label) row.FindControl("Label1") as Label).Text;
 // Команда удаления
 SqlDataSource1.DeleteCommand =
   "select public.dblink_connect('con1',
      'dbname=students52 user=pm**** password=********');"
   + "select public.dblink_exec('con1',
        'delete from photo52_2.sale where id=" +ID+ "'); "
   + "select public.dblink_disconnect('con1');";
 // Выполнение команды
 SqlDataSource1.Delete();
}
```

Применение AJAX

AJAX (от англ. *Asynchronous Javascript and XML* – асинхронный JavaScript и XML) – подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в

«фоновом» обмене данными браузера с веб-сервером. В результате при обновлении данных веб-страница не перезагружается полностью и веб-приложения становятся более быстрыми и удобными.

Чтобы применить подход AJAX к созданному ранее Web-сайту, нужно найти в панели инструментов на вкладке AJAX-расширения элемент **`ScriptManager`** и поместить его на форму перед всеми остальными элементами. Затем в этой же вкладке найти элемент **`UpdatePanel`** и поместить его на форму, после чего все элементы, требующие обновления данных, перенести внутрь данной панели. Теперь **`GridView`** при модификации данных будет обновляться без перезагрузки страницы (см. рисунок 35).

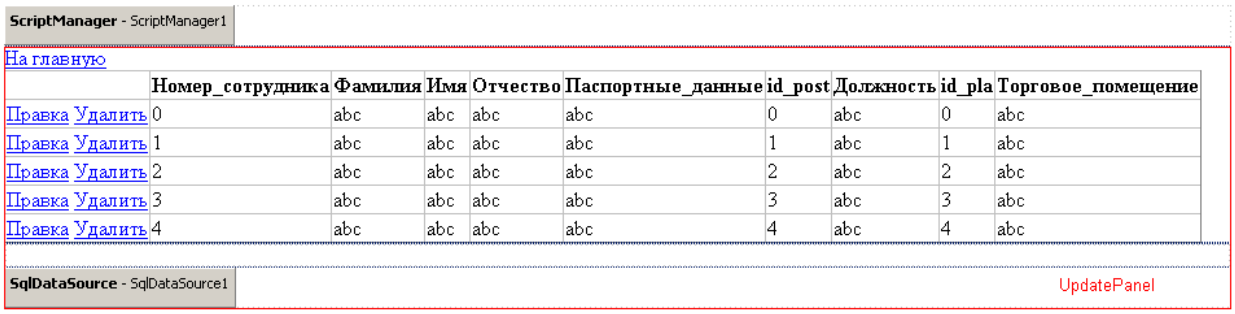

Рисунок 35 – Применение элемента ScriptManager

Задание

Для созданной в лабораторной работе № 1 базы данных с оптимальным размещением таблиц по двум узлам написать Web-интерфейс (портал) для работы с этой базой. Портал должен уметь корректно обрабатывать вводимые данные, делать выборку данных из таблиц, вставлять, удалять и изменять данные в таблицах, расположенных в различных базах данных, сохраняя целостность распределённой базы данных.

При реализации интерфейса использовать технологию AJAX для передачи данных на страницу без её перезагрузки.

Примечание: При реализации портала ***запрещается*** использовать язык PHP, рекомендуется использовать ASP.NET.

Требования к оформлению отчёта

Отчёт по лабораторной работе должен включать в себя:

- титульный лист;
- краткое описание предметной области, для которой разработана база данных;
- ER-диаграммы баз данных для каждого узла;
- описание разработанного портала;
- программный код, написанный непосредственно студентами;
- тестирование портала.

Отчёт не должен содержать орфографических, пунктуационных и смысловых ошибок. Все его разделы должны быть выдержаны в едином стиле оформления.

Критерии оценивания качества работы

1. Количество корректно обрабатываемых полей таблиц:
   1 – приложение обрабатывает данные в ***10*** полях таблиц, при этом ***6*** из этих полей служат внешними ключами для других таблиц, а ***3*** из них – внешними ключами для таблиц, размещённых в другой базе;
   0 – приложение обрабатывает данные в ***6*** полях таблиц, при этом ***3*** из этих полей служат внешними ключами для других таблиц, а ***1*** из них является внешним ключом для таблицы, размещённой в другой базе;
   л.р. не принимается – иначе.

Примечание: Сложность выполнения лабораторной работы рассчитывается в количестве полей, которые обрабатываются, суммарно со всех использованных таблиц. Например, если необходима обработка 10 полей, можно взять одну таблицу, включающую в себя 5 полей, одну таблицу, включающую в себя 3 поля, и одну таблицу, включающую в себя 2 поля. В сумме это даст 10 полей. Из них 3 должны быть внешними ключами в локальной базе данных (ссылаются из student51 в student51), а 3 других – внешними ключами ко второй локальной базе данных (ссылаются из student51 в student52).

2. Применение технологии AJAX:
   1 – портал использует технологию AJAX более чем на 1 странице;
   0 – портал использует технологию AJAX только на 1 странице;
   л.р. не принимается – ни одна страница портала не использует технологию AJAX.
3. Содержание отчёта:
   1 – отчёт удовлетворяет всем требованиям;
   0 – отчёт удовлетворяет не всем требованиям.
4. Обработка ошибок:
   1 – все возможные ошибки и нестандартные ситуации (например, неудачная попытка открытия файла) обрабатываются программой, которая выдаёт соответствующее сообщение;
   0 – не все возможные ошибки обрабатываются программой.
5. Применение принципов структурного программирования:
   1 – все повторяющиеся либо логически целостные фрагменты программы выделены в качестве функций; работа каждой функции полностью определяется её параметрами (т.е. не используются глобальные переменные, все данные, нужные функции для работы, передаются ей через параметры); программа позволяет без перекомпиляции изменять все параметры, от которых зависит её работа; в тексте программы отсутствуют числовые константы (все необходимые константы объявляются как поименованные);
   0 – иначе (не выполняется что-либо из перечисленного).
6. Наличие комментариев в исходных кодах:
   1 – комментариев достаточно для документирования исходных кодов;
   0 – комментариев недостаточно.
7. Глубина понимания материала лабораторной работы каждым членом бригады:
   1 – быстрые и правильные ответы на все вопросы;
   0 – не на все вопросы ответы правильные и быстрые;
   л.р. не принимается – на половину вопросов ответы неправильные.

Список литературы

1. СУБД PostgreSQL [Электронный ресурс]. – Режим доступа: http://www.postgresql.org/.