Доступ к данным при помощи ADO.NET

ASP.NET MVC 4.0

2013

Цель

- Узнать о базовом доступе к данным при помощи классов ADO.NET.
- Познакомиться с моделью данных, сохраняемых в базе.
- Разработать простое приложение CRUD.

Классы ADO.NET

- Классы конкретных поставщиков данных (находятся в пространствах имен System.Data.OleDb, System.Data.SqlClient, и т.п.)
- Классы для автономной работы с данными (находятся в пространствах имен System.Data и System.Data.Common)

Классы поставщиков данных

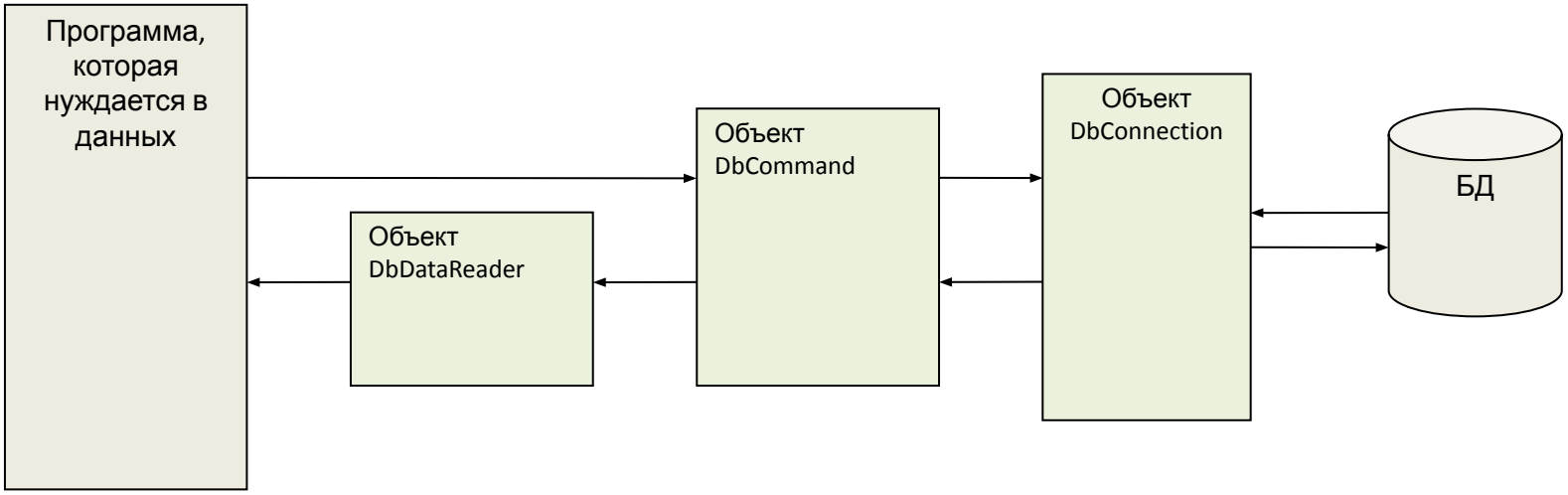
Для поставщика SqlClient это:

- SqlConnection,
- SqlCommand,
- SqlDataReader,
- SqlDataAdapter.

Для поставщика SqlServerCe это:

- SqlCeConnection,
- SqlCeCommand,
- SqlCeDataReader,
- SqlCeDataAdapter.

Все классы поставщиков наследуют абстрактных предков, например,
DBConnection □ SqlConnection, DBCommand □ SqlCommand



Класс Connection

- Устанавливает соединение с хранилищем данных методом *Open()*. Основное свойство – *ConnectionString* – строка соединения.

- Позволяет начать транзакцию методом *BeginTransaction()* (завершение или откат транзакции выполняются методами объекта Transaction, который возвращает *BeginTransaction()*).

```
string conStr = "Data Source=|DataDirectory|GB.sdf";
SqlCeConnection connection = new SqlCeConnection(conStr);
```

На сайте http://www.connectionstrings.com/ можно найти примеры разнообразных строк соединения

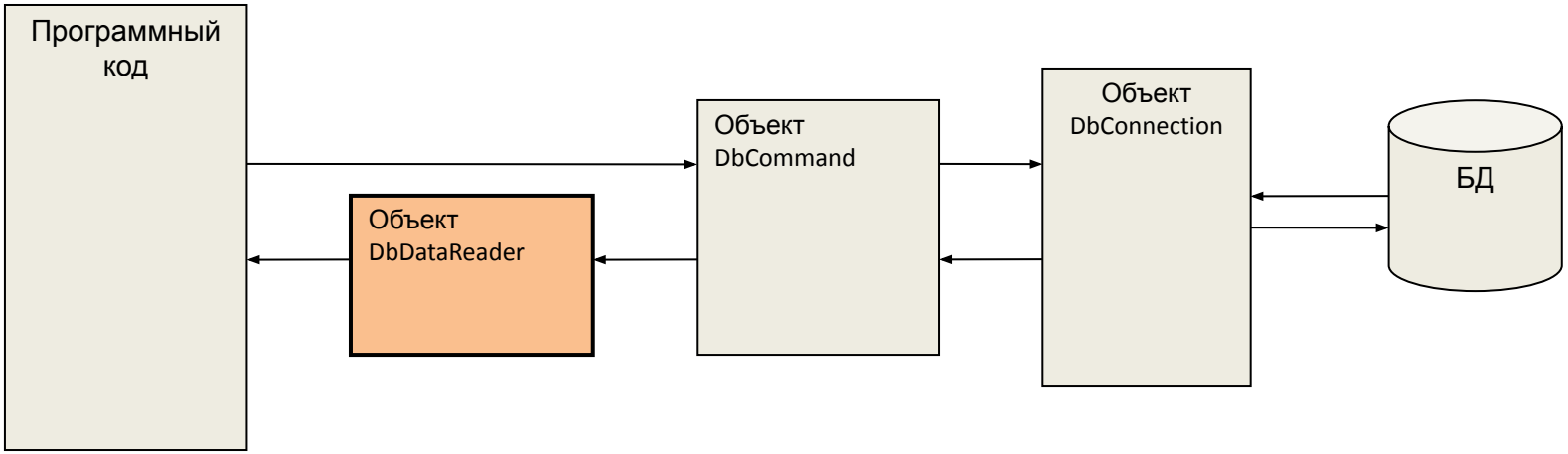
Класс Command

- Представляет собой SQL-оператор или хранимую процедуру.
- Sql-команда формируется свойствами *CommandText* и *CommandType*.
- Параметры команды устанавливаются коллекцией *Parameters*.
- Объект Command связывается с объектом соединения свойством *Connection*.
- Чтобы выполнить команду, нужно использовать один из методов:
  - *ExecuteNonQuery()* – для изменения данных,
  - *ExecuteReader()* – создает поток *DataReader* для чтения данных из хранилища,
  - *ExecuteScalar()* – создает поток *DataReader*, из которого читает единственное значение – первый столбец первой строки.

```
string commandString = "SELECT Id, Text, Author, RecordDate FROM Records";
SqlCeCommand command = new SqlCeCommand(commandString, connection);
```

Класс DataReader

Позволяет только читать данные из хранилища и только в одном направлении. Собственно чтение выполняет метод Read().




```
using (SqlCeDataReader reader = command.ExecuteReader(CommandBehavior.CloseConnection))
{
    while (reader.Read()) {
        Id = (int)reader["Id"];
    }
}
```

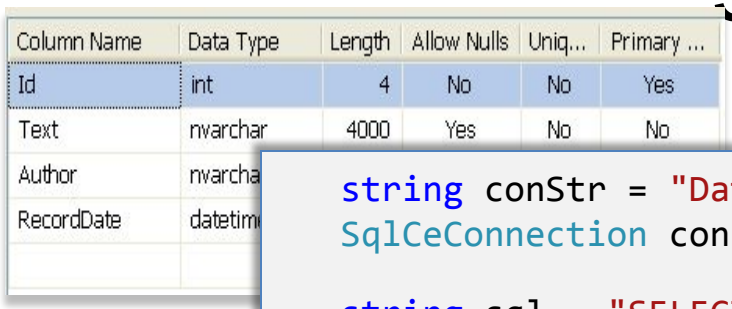
После открытия потока данных указатель находится *перед первой* записью.

Пример

В базе данных есть таблица Records

Column Name	Data Type	Length	Allow Nulls	Uniq...	Primary ...
Id	int	4	No	No	Yes
Text	nvarchar	4000	Yes	No	No
Author	nvarchar	4000	Yes	No	No
RecordDate	datetime	8	No	No	No

Получить коллекцию записей



Column Name	Data Type	Length	Allow Nulls	Uniq...	Primary ...
Id	int	4	No	No	Yes
Text	nvarchar	4000	Yes	No	No
Author	nvarcha				
RecordDate	datetim				

```
string conStr = "Data Source=|DataDirectory|GB.sdf";
SqlCeConnection connection = new SqlCeConnection(conStr);

string sql = "SELECT Id, Text, Author, RecordDate FROM Records";

SqlCeCommand command = new SqlCeCommand(sql, connection);
connection.Open();

List<Record> result = new List<Record>();

using (SqlCeDataReader reader =
       command.ExecuteReader(CommandBehavior.CloseConnection))
{
     while (reader.Read()) {
         Record record = new Record() {
            Id = (int)reader["Id"],
            Text = (string)reader["Text"],
            Author = (string)reader["Author"],
            RecordDate = (DateTime)reader["RecordDate"],
         };
         result.Add(record);
     }
}
```

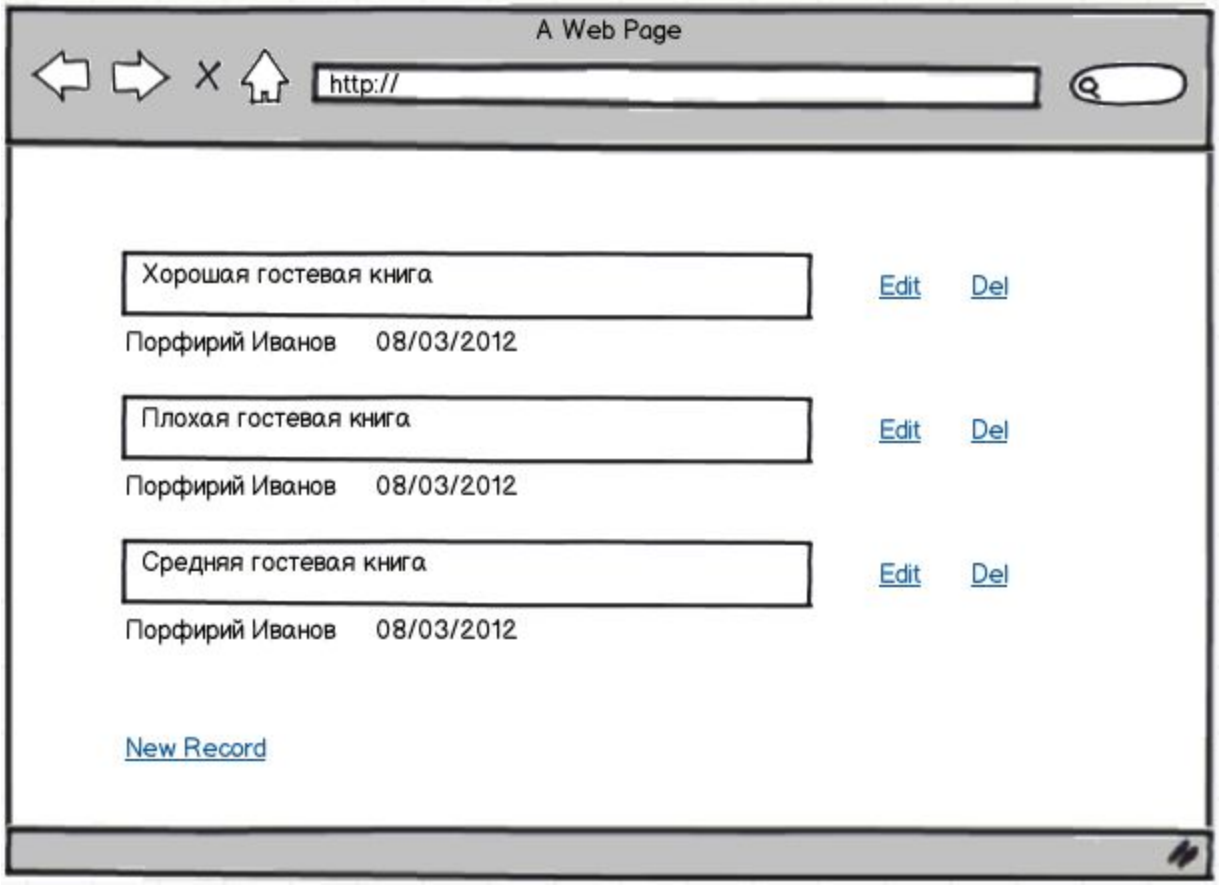
Приложение GuestBook – гостевая книга

Разработать приложение «Гостевая книга»

- Каждый желающий может сделать запись в гостевой книге, нужно только ввести текст записи и свое имя. Все могут читать внесенные записи. Гостевая книга хранится в базе данных.

- Дополнительно: записи можно упорядочивать и фильтровать по вхождению заданной подстроки в имя гостя или в текст записи.

GuestBook. Главная страница



Создадим приложение MVC 4 (Basic). Назовем его GuestBook.

Модель

Модель гостевой книги представляет собой коллекцию объектов Record.

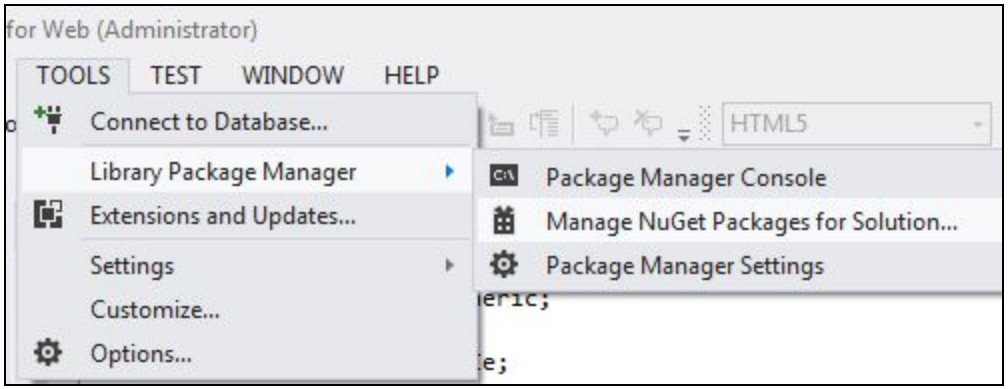
```
public class Record
{
    public int Id { set; get; }
    public string Text { set; get; }
    public string Author { set; get; }
    public DateTime RecordDate { set; get; }
}
```

В качестве СУБД используем SqlServer CE 4.0 или SqlExpress.

SqlServer CE 4.0

- SqlServer CE 4.0 представляет собой исполняемую программу, которая просто копируется в каталог bin веб-приложения, никакого другого развертывания не требуется.
- Сервер поддерживает многопоточные запросы, поэтому может использоваться в небольших ASP.NET приложениях.
- SqlServer CE допускает тот же доступ к данным, что и полная версия (ADO.NET, EF), но не имеет хранимых процедур и представлений.
- Бесплатен.

Загрузка пакета через NuGet



Microsoft SQL Server Compact Edition

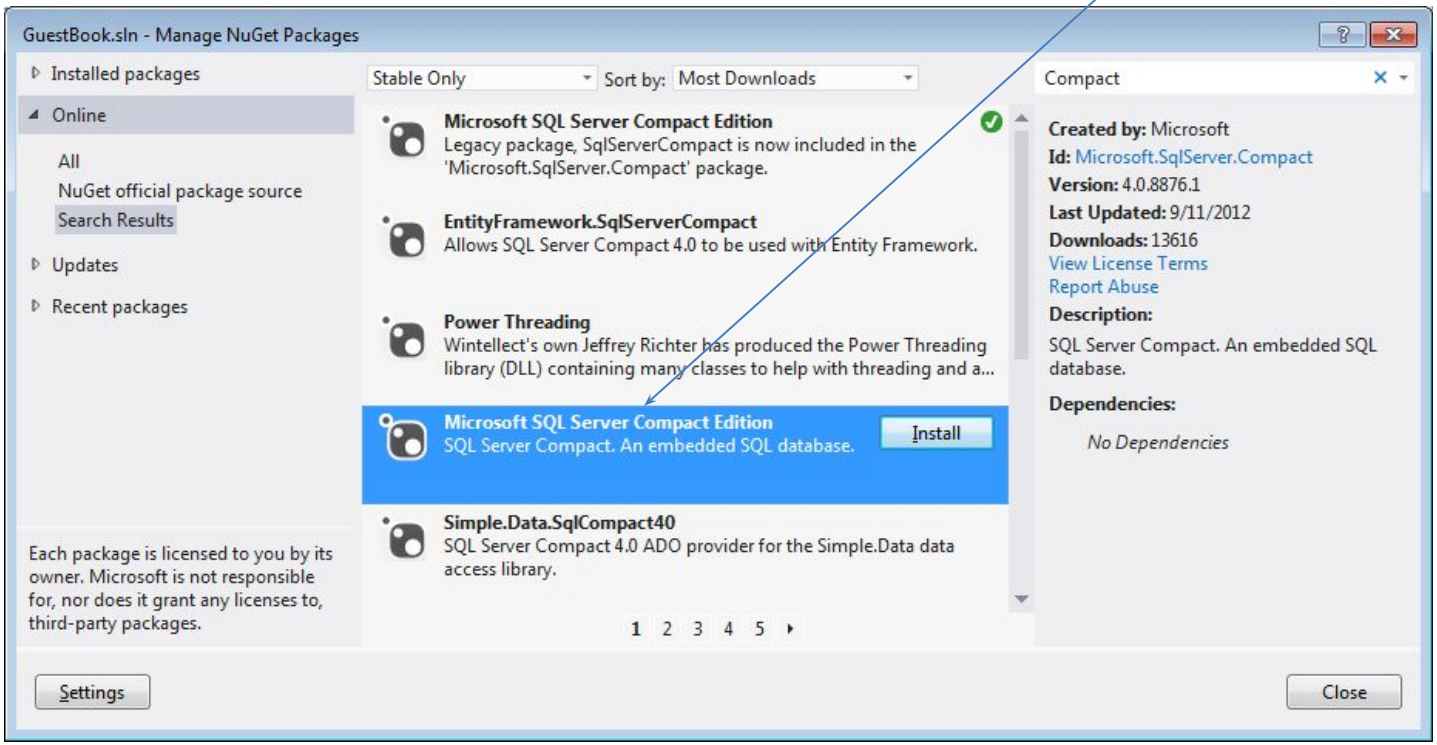
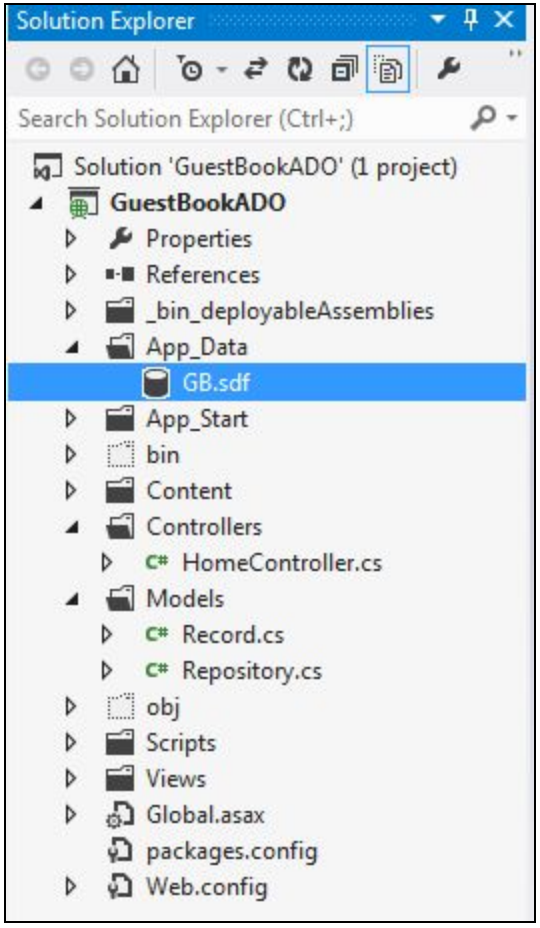
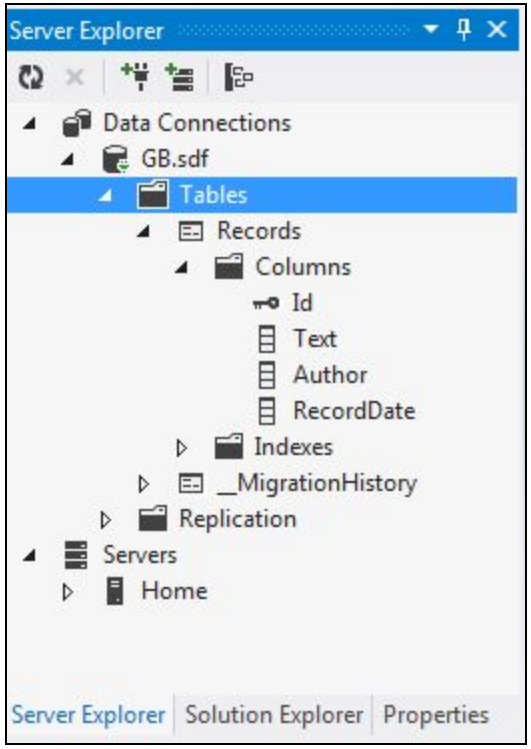


Схема данных





Добавим в каталог App_Data базу данных в формате .sdf .

Многослйная архитектура

Одним из важнейших аспектов проектирования приложения является выбор архитектуры.

Классическим примером является **многослйная архитектура**.

- На самом верху – **уровень представления** (Presentation Layer):

  Предназначен для взаимодействия с пользователем.

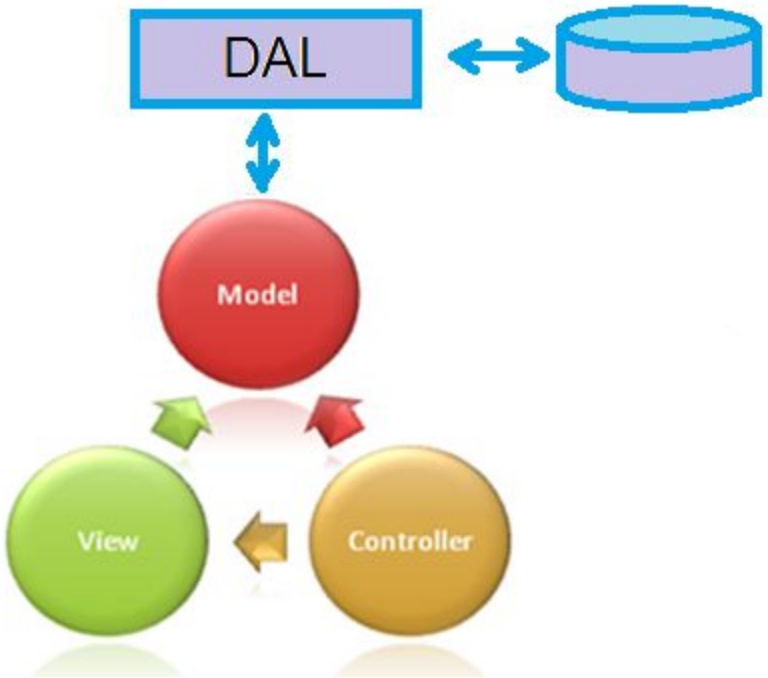
- За уровнем представления следует **уровень бизнес логики** (Business Logic Layer):

  Осуществляет логическую обработку поступающих команд и бизнес правил.

- В самом низу располагается **уровень доступа к данным** (Data Access Layer):

  Осуществляет доступ к хранилищу данных в двустороннем направлении.

Слой доступа к данным



В слое доступа всю техническую работу по общению с базой будут выполнять классы ADO.NET

Слой доступа к данным

```
public class Repository
{
   public string conStr = "Data Source=|DataDirectory|GB.sdf";

   public List<Record> GetRecords() {
      SqlCeConnection connection = new SqlCeConnection(conStr);
      string commandString = "SELECT Id, Text, Author, RecordDate FROM Records";
      SqlCeCommand command = new SqlCeCommand(commandString, connection);
      connection.Open();
      //
      List<Record> result = new List<Record>();
      //
      using (SqlCeDataReader reader =
             command.ExecuteReader(CommandBehavior.CloseConnection))
      {
         while (reader.Read()) {
            Record record = new Record() {
               Id = (int)reader["Id"],
               Text = (string)reader["Text"],
               Author = (string)reader["Author"],
               RecordDate = (DateTime)reader["RecordDate"],
            };
            result.Add(record);
         }
      }
      return result;
   }
}
```

Слой доступа выполнен по шаблону Repository – класс без состояния с интерфейсом, подобным коллекции.

Слой доступа к данным

Метод Create()

```
public void CreateRecord(Record record)
{
    string commandString = @"INSERT INTO Records (Text, Author, RecordDate)
                             VALUES (@Text, @Author, @RecordDate) ";

    using (SqlCeConnection connection = new SqlCeConnection(conStr))
    {
        SqlCeCommand command = new SqlCeCommand(commandString, connection);
        command.Parameters.AddWithValue("@Text", record.Text);
        command.Parameters.AddWithValue("@Author", record.Author);
        command.Parameters.AddWithValue("@RecordDate", record.RecordDate);
        connection.Open();
        command.ExecuteNonQuery();
    }
}
```

Класс Command имеет коллекцию параметров.
Параметризованные команды выполняются быстрее и позволяют избежать sql-инъекций.

SQL-инъекции

Если вместо параметризованной формы команды

```
"INSERT INTO Records (Text) VALUES (@Text)"
```

использовать конкатенацию строк

```
"INSERT INTO Records (Text) VALUES (" + text + ")"
```

то при определенных значениях строки `text`

```
text = "'a'); DELETE * FROM Records--"
```

может получиться вредоносный код

```
"INSERT INTO Records (Text) VALUES ('a'); DELETE * FROM Records--)"
```

`;` - граница начала нового оператора
`--` - коментирует оставшуюся часть оператора.

Строка соединения в web.config

Хранение строки соединения в web.config, позволяет изменять параметры соединения без перекомпиляции приложения.

```
<connectionStrings>
   <add name="GB"
      connectionString="Data Source=|DataDirectory|GB.sdf"
      providerName="System.Data.SqlServerCe.4.0" />
 </connectionStrings>
```

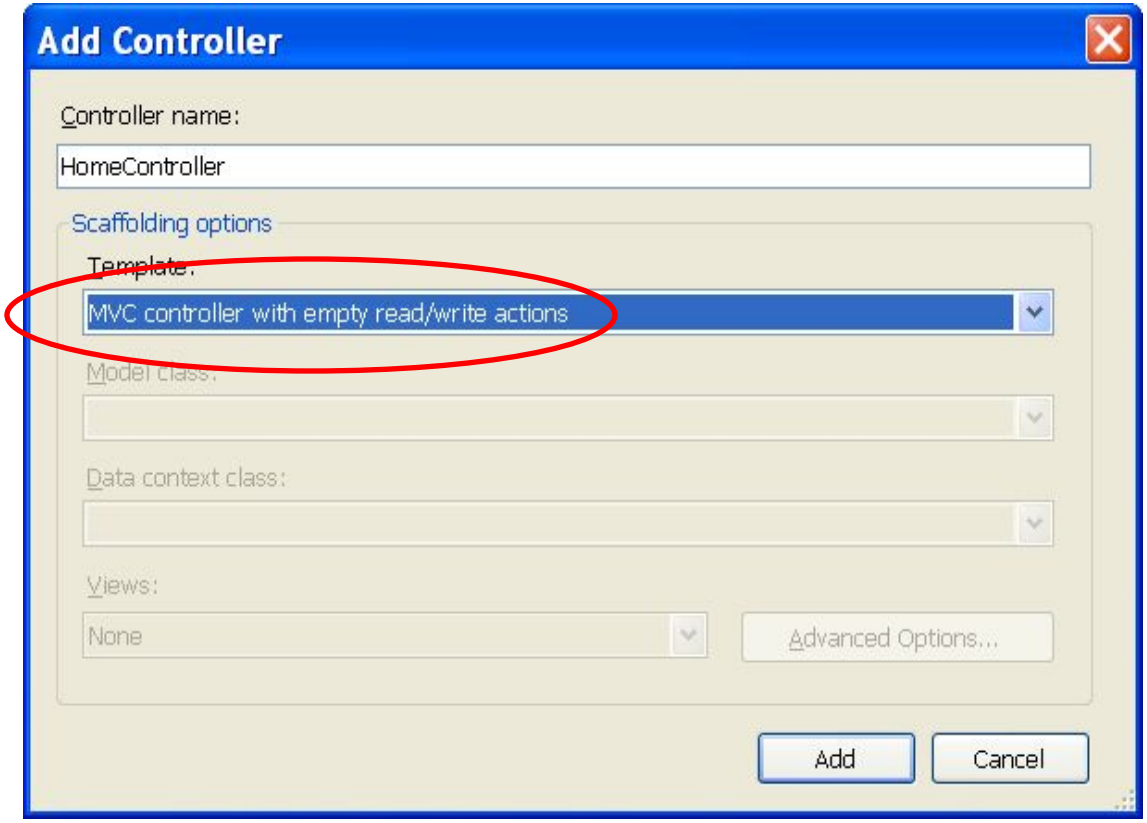
Изменение в классе Repository:

```
public string conStr = "Data Source=|DataDirectory|GB.sdf";

public string conStr =
    WebConfigurationManager.ConnectionStrings["GB"].ConnectionString;
```

WebConfigurationManager – класс для работы с файлами конфигурации web приложения.

Контроллер Home

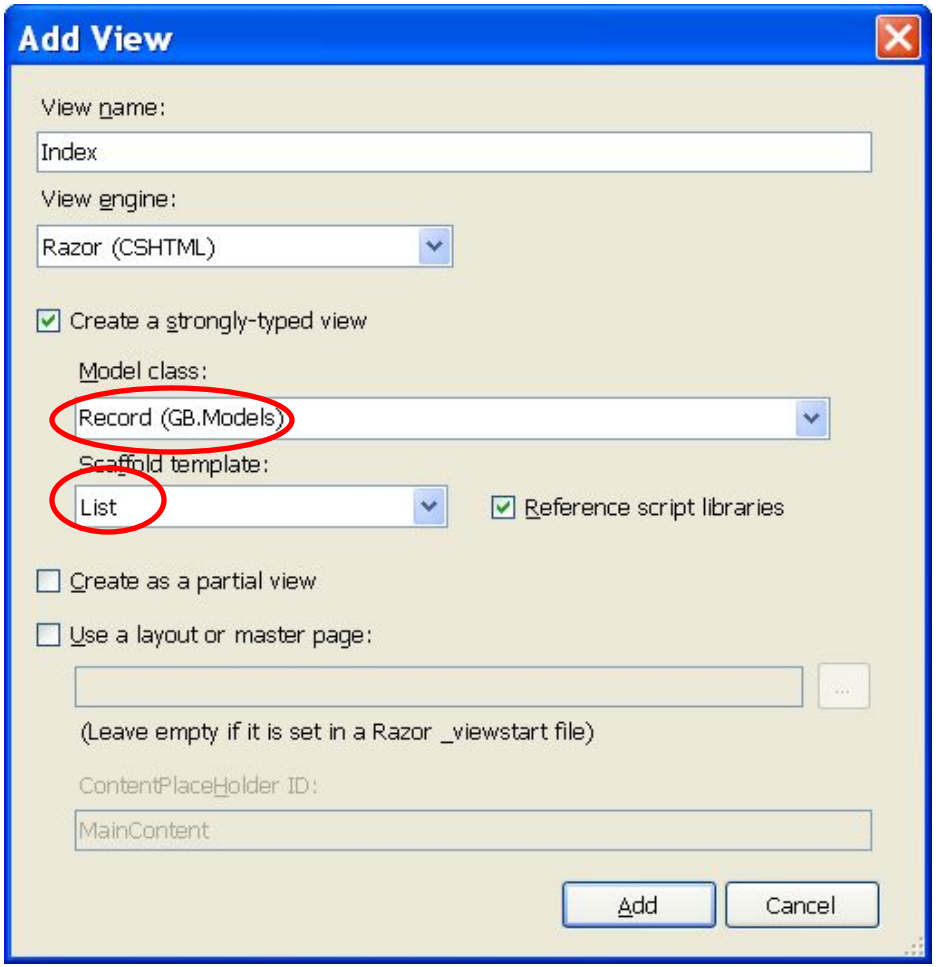



```
        Repository repository = new Repository();

        public ActionResult Index()
        {
            return View(repository.Read);
        }
```

Представление Home/Index

```
public ActionResult Index()
{
    return View(new Repository().Read());
}
```



Методы HomeController.Create()

Два метода Create() в HomeController

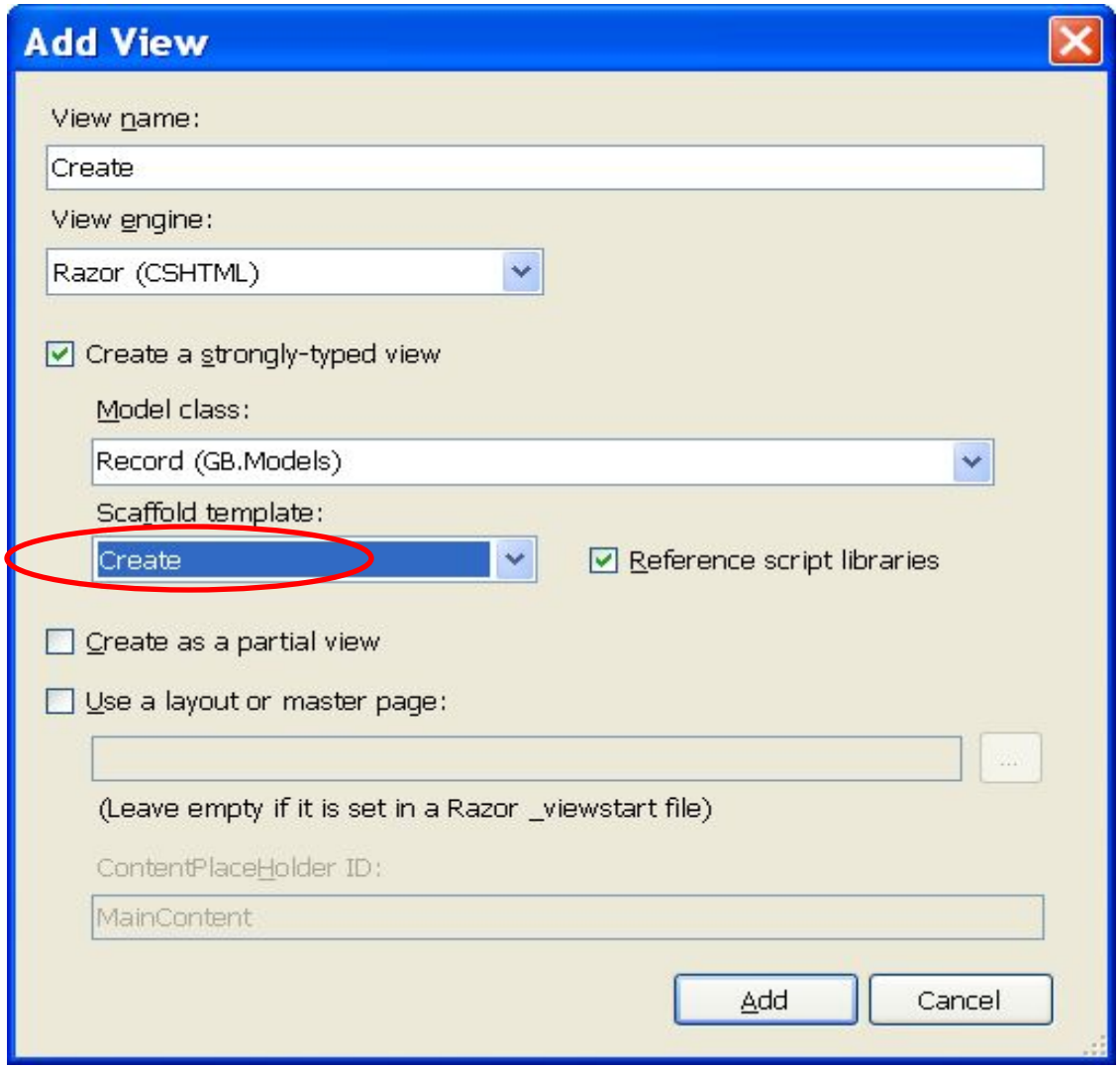
```
//
// GET: /Home/Create
public ActionResult Create()
{
    return View();
}

//
// POST: /Home/Create
[HttpPost]
public ActionResult Create(Record record)
{
    try
    {
        repository.Create(record);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Различие между GET и POST

GET	POST
В спецификации HTTP/1.1 говорится, что метод **лишен действий** – только получение запрашиваемой информации	**Может выполнять** некоторые **действия**
Запрос **может кэшироваться** браузером	Запрос **не кэшируется** браузером
Ограниченная длина отправка данных- **8 К**	Лимит ограничен сервером – около **2 М**
Можно увидеть переданные данные в адресной строке	Данные **нельзя увидеть** воочию
Может передать **только ASCII** символы	Может передавать **любые данные**, в том числе файлы

Представление Home/Create



Типизированное представление дает возможность сгруппировать данные, внесенные в форму пользователем, в объект Record и передать его во второй метод Create().

Самостоятельно

Закончить гостевую книгу, сделав:

- Удаление записей.
- Изменение записей.
- Проверку ввода.