ГЛАВА 9

Службы WCF Data Services

Доступность: .NET Framework 3.5 SP1 (ограниченная функциональность) и выше

Службы данных WCF (WCF Data Services, ранее "Astoria" и ADO.NET Data Services) позволяют модифицировать и поставлять данные по интерфейсу HTTP REST. Службы WCF Data Services (WDS) включают богатый язык запросов и могут быть легко доступны через автоматически сгенерированные прокси-классы либо посредством низкоуровневых запросов HTTP.

Службы WCF Data Services поддерживают возврат данных во множестве популярных форматов, в том числе XML, AtomPub и JSON, и потенциально очень полезны для сценариев интеграции и приложений, которые не поддерживают прямых соединений с базой данных, например, Silverlight.

На заметку! Пока шла работа над этой главой, Microsoft изменила название ADO.NET Data Services на WCF Data Services. Однако имена шаблонов VS пока не изменены, поэтому в примерах настоящей главы используются имена шаблонов ADO.NET Data Services.

Введение в WCF Data Services

Прежде чем приступать к изучению WDS, понадобятся какие-то данные, чтобы было с чем поработать. Если это еще не сделано, обратитесь к введению и установите базу данных примеров. В этой главе будет использоваться SQL Server 2008, но не думайте, что вы ограничены использованием только SQL Server, поскольку WDS работает со всем, что поддерживает Entity Framework (см. главу 8).

Чтобы представить данные с помощью этой службы, необходимо выполнить четыре следующих действия.

1. Создать классы Entity Framework для данных, которые должны быть представлены.
2. Создать приложение-хост ADO.NET для службы WDS.
3. Создать службу WDS.
4. Сконфигурировать правила доступа к службе.

Итак, приступим. Откройте Visual Studio и создайте новый веб-сайт ASP.NET; измените значение в раскрывающемся списке Web location (Веб-расположение) на HTTP и введите местоположение `http://localhost/Chapter9/`.

Внимание! Хостинг служб WCF Data Services в IIS на одной машине постоянно приводил к ошибке HTTP 500 при попытке запроса данных. Я так и не смог докопаться до причин, поэтому если у вас получится то же самое, попробуйте вместо этого поработать с локальным веб-сервером.

Entity Framework

Службы WDS должны знать, как структурированы и связаны данные, которые мы хотим представить. Для предоставления этой информации используется Entity Framework.

1. Добавьте в проект новую сущностную модель данных ADO.NET.
2. Назовите эту новую сущностную модель данных ADO.NET как `Chapter9Model.edmx` (рис. 9.1).

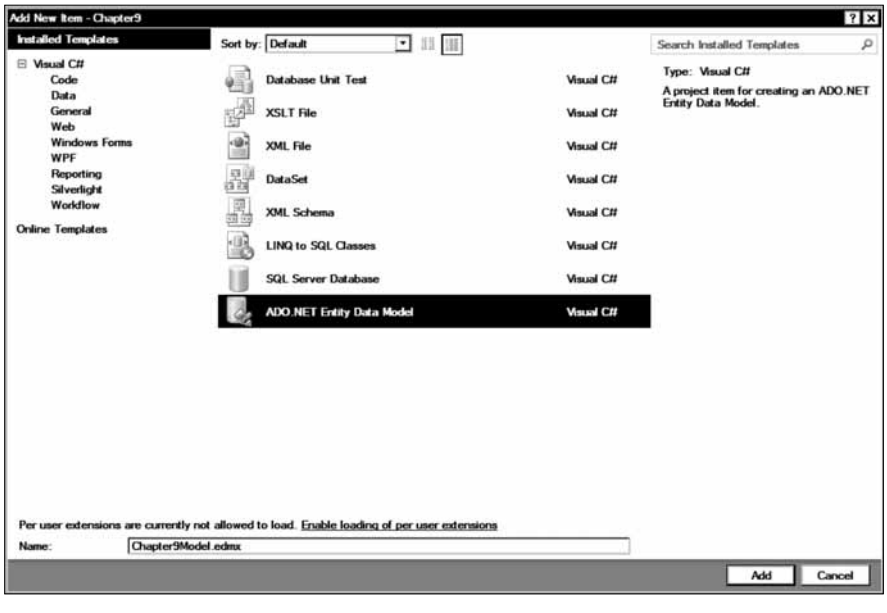

Рис. 9.1. Добавление сущностной модели данных ADO.NET

3. Щелкните на кнопке Add (Добавить).
4. Visual Studio предложит расположить эти файлы в каталоге `App_Code`. Согласитесь.
5. Теперь Visual Studio спросит, как необходимо генерировать модель. Выберите вариант Generate from database (Генерировать из базы данных) и затем щелкните на кнопке Next (Далее), как показано на рис. 9.2.
6. Если пока еще нет соединения с базой данных примеров, создайте его, щелкнув на кнопке New Connection (Новое соединение) и введите параметры соединения (рис. 9.3).
7. Visual Studio исследует структуру базы данных и отобразит экран, подобный показанному на рис. 9.4, где можно будет выбрать элементы для генерации классов для EF. Разверните узел Tables (Таблицы), чтобы отобразить все доступные таблицы.

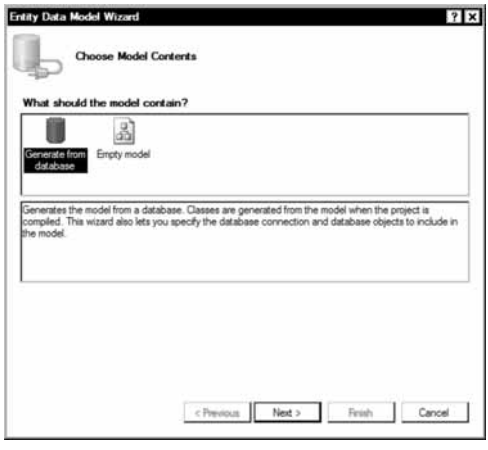

Рис. 9.2. Генерация модели из базы данных

Рис. 9.3. Создание нового соединения с базой данных

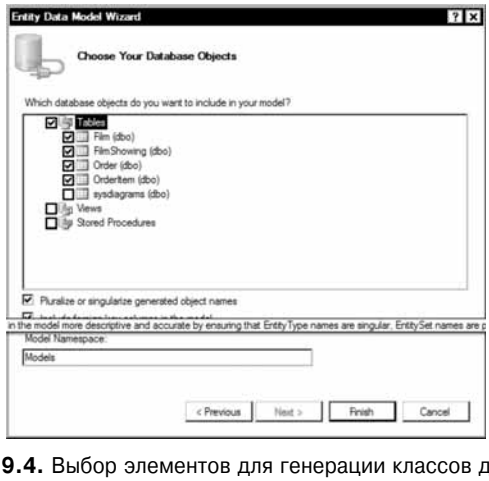

Рис. 9.4. Выбор элементов для генерации классов для EF

8. Отметьте флажки возле всех таблиц кроме `sysdiagrams`.
9. Удостоверьтесь, что флажок Pluralize or singularize generate object names (Установить множественную или единственную форму для имен генерируемых объектов) отмечен.
10. Удостоверьтесь, что в поле Model Namespace (Пространство имен модели) установлено значение `Models`.
11. Щелкните на кнопке Finish (Готово).
12. Щелкните на кнопке OK.

VS2010 сгенерирует классы EF для базы данных и отобразит поверхность визуального конструктора (рис. 9.5).

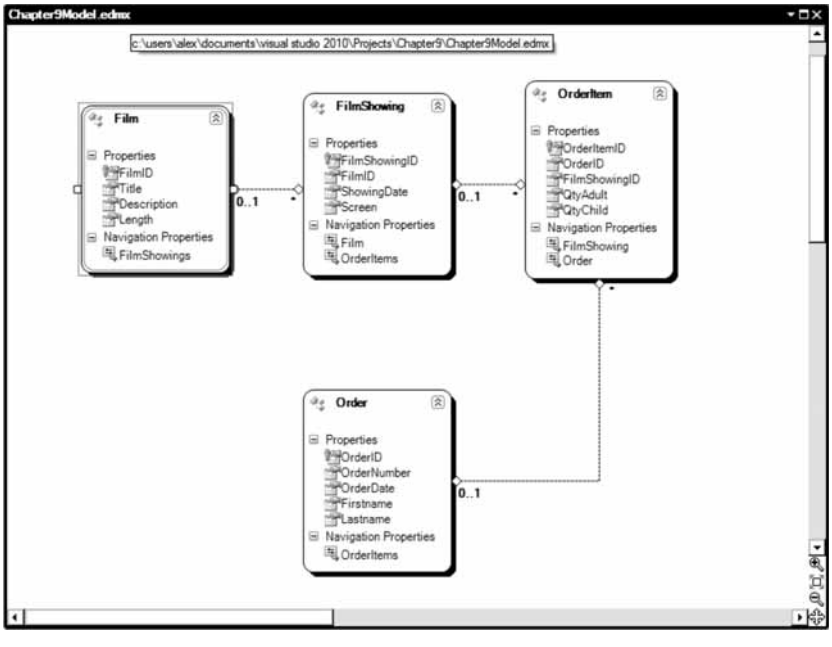

Рис. 9.5. Сущностная модель

Создание службы данных

Все, что теперь осталось — это представить классы EF, добавив новую службу данных и сконфигурировав правила для доступа к ней. Ниже перечислены необходимые шаги.

1. Добавьте в проект новую службу данных ADO.NET и назовите ее `MovieService.svc`.
2. Щелкните на кнопке OK.
3. Откройте `~/MovieService.cs`.
4. Теперь потребуется сообщить WDS, какой тип класса будет представлять служба. Модифицируйте код следующим образом:

```
public class MovieService : DataService<Models.BookEntities>
{
  // Этот метод вызывается только один раз для инициализации политик службы.
```

```
    public static void InitializeService(DataServiceConfiguration config)
    {
        config.SetEntitySetAccessRule("*", EntitySetRights.AllRead);
        config.DataServiceBehavior.MaxProtocolVersion =
                DataServiceProtocolVersion.V2;
    }
}
```

Внимание! В приведенном примере `DataService` принимает тип по имени `BookEntities`, что зависит от имени базы данных. VS по умолчанию снабжает сущности префиксом по имени базы данных (при желании это можно исправить в `Chapter9Model.Designer.cs`).

Настройка содержимого в Internet Explorer

По умолчанию в ответ на запрос WDS возвращает XML-код. Во время тестирования удобно просматривать возвращенный XML-код в браузере, подобном Internet Explorer. Однако по умолчанию Internet Explorer воспринимает результаты, возвращенные службой данных, как информацию канала RSS в форме AtomPub.

Это не слишком полезно, потому для просмотра XML-кода следует изменить настройки Internet Explorer.

1. Откройте Internet Explorer и выберите в меню пункт Сервис⇨Свойства обозревателя. В открывшемся диалоговом окне перейдите на вкладку Содержание.
2. Щелкните на кнопке Параметры в разделе Каналы и веб-фрагменты и в открывшемся диалоговом окне снимите отметку с флажка Включить показ ленты чтения канала (рис. 9.6).

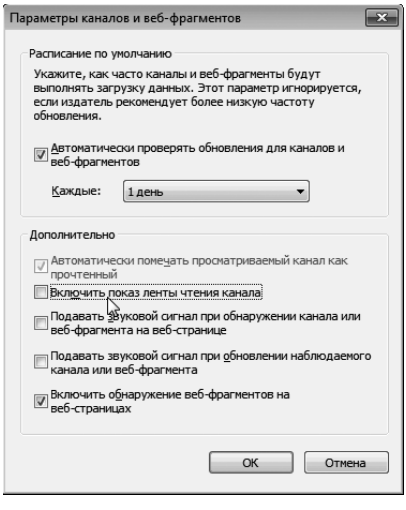

Рис. 9.6. Изменение настроек представления содержимого в Internet Explorer 8

Введение в WDS

К этому моменту служба данных готова.

1. Щелкните правой кнопкой мыши на `MovieService.svc` и выберите в контекстном меню пункт Set As Start Page (Установить как стартовую страницу).
2. Нажмите <F5> для запуска приложения. Если все в порядке, то на экране отобразится XML-представление классов EF (рис. 9.7).

Запрос служб WCF Data Services

Для передачи параметров запроса службы WCF Data Services используют URL. Например, чтобы извлечь название фильмов, в конец существующего URL необходимо добавить `/Films` (например, URL может выглядеть так: `http://localhost/Chapter9/MovieService.svc/Films`). В результате должен быть получен список всех фильмов в формате AtomPub, как показано на рис. 9.8.

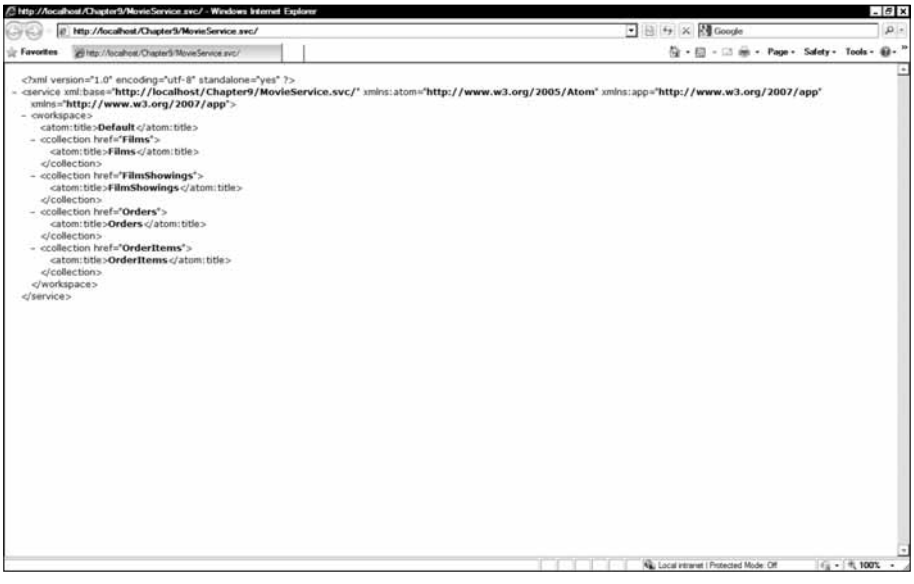

Рис. 9.7. Вывод, полученный в результате обращения к тестовой службе

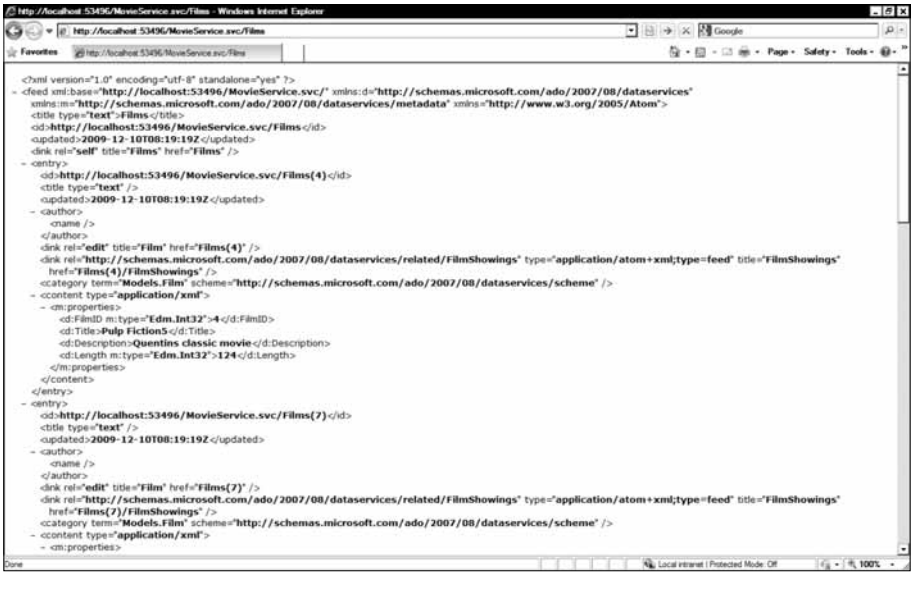

Рис. 9.8. Возврат списка всех фильмов из базы данных

Получать все данные сразу — не всегда удобно, поэтому WDS поддерживает набор операций запросов, которые работают с URL.

Некоторые часто используемые методы перечислены в табл. 9.1, а полный их список доступен по адресу:

```
http://msdn.microsoft.com/en-us/library/cc907912.aspx
```

Таблица 9.1. Список распространенных операций запросов

Действие	Операция
Получить фильм с идентификатором, равным 3	`Films(3)`
Выбрать фильм с `FilmID`, равным 3	`Films?$filter=FilmID%20eq%203`
Получить поле `FilmName` (Название фильма) из первого `FilmShowing` (Сеанс)	`FilmShowings(1)/Film/Title`
Получить сеансы первого фильма	`Films(1)/FilmShowings`
Отобразить заказы по дате заказа	`Orders?$orderby=OrderDate`
Упорядочить список фильмов по убыванию	`Orders?$orderby=OrderDate%20desc`
Выбрать два верхних заказа	`Orders?$top=2`
Пропустить первый заказ и выбрать следующие два	`Orders?$skip=1&top=2`

При работе со службами WCF Data Services необходимо помнить о следующих моментах.

- Объекты чувствительны к регистру символов. `Film` — это не то же самое, что `film`.
- Запрос должен быть закодирован для URL, поэтому пробелы следует заменять `%20`.
- Для указания опций применяется символ строки запроса `?` и `$`.

Существует ли способ ограничения количества возвращаемых записей?

Существует, если используются службы WCF Data Services версии 1.5. Изменения в VS2010/.NET 4.0 описаны далее в главе.

Безопасность служб WCF Data Services

WDS позволяет организовать тонко настраиваемый доступ к индивидуальным сущностям. Например, откройте `MovieService.cs` и найдите следующую строку:

```
config.SetEntitySetAccessRule("*", EntitySetRights.AllRead);
```

Этот код позволяет организовать доступ только для чтения ко всем сущностям (определяется символом `*`). Чтобы открыть полный доступ ко всему (обычно это очень плохая идея, но для целей тестирования подходит), необходимо указать такую строку:

```
config.SetEntitySetAccessRule("*", EntitySetRights.All);
```

Внимание! Если сделать так, то любой, имеющий доступ к службе данных, получит полный доступ к данным.

Привилегии могут применяться к отдельным объектам, которые указываются своими именами. Следующая строка кода откроет полный доступ к сущности `Film`:

```
config.SetEntitySetAccessRule("Films", EntitySetRights.All);
```

Очень скоро нам понадобится полный доступ к сущности `Film`, поэтому добавьте приведенную ниже строку в `MovieService.cs`.

Перехватчики запросов

Иногда требуется перехватить пользовательский запрос, чтобы применить к нему дополнительную логику (например, чтобы фильтровать данные в зависимости от текущего пользователя). WDS позволяет делать это с помощью перехватчиков запросов.

В следующем примере демонстрируется применение этой техники к любым запросам фильмов, разрешая передавать только те, у которых значение `FilmID` равно `1`. Добавьте в `MoveService.svc` такой код:

```
using System.Linq.Expressions;
[QueryInterceptor("Films")]
public Expression<Func<Film, bool>> FilterOnlyShowFilmID1()
{
   return f => f.FilmID==1;
}
```

Возврат результатов в других форматах

Службы WDS могут возвращать данные в следующих форматах:

- AtomPub
- JSON
- XML

AtomPub — формат возврата по умолчанию. Начиная с WDS 1.5, можно управлять отображением элементов на элементы AtomPub.

Давайте посмотрим, как возвращать результаты в формате JSON, используя jQuery, а затем — как обратиться к службе WDS из консольного приложения.

Использование JSON в JavaScript

JSON — это формат, обычно используемый веб-приложениями, поскольку он существенно менее многословен, чем XML. Формат JSON также воспринимается многими библиотеками JavaScript. Если необходимо, чтобы службы WCF Data Services форматировали результаты в JSON, при выполнении запроса установите заголовок `Accept` в `application/json`.

В следующем примере показано, как использовать jQuery для извлечения названия первого фильма в наборе (за дополнительной информацией о jQuery обращайтесь в главу 12):

```
<script>
   function getMovieTitle() {
       $.ajax({
          type: "GET",
          dataType: "json",
         url: "MovieService.svc/Films(1)",
         success: function (result) {
            alert(result.d.Title);
         },
         error: function (error) {
            alert('error ');
         }
      });
   }
</script>
<input type="button" Value="Get Movie Title"
       onclick="javascript:getMovieTitle();" />
```

Какие данные JSON будут возвращены предыдущим вызовом? Используя прокси Fiddler (www.fiddlertool.com), предназначенный для отладки веб-приложений, можно посмотреть, что возвращается. Ниже приведен низкоуровневый код JSON:

```
{ "d" : {
"__metadata": {
"uri": "http://localhost/Chapter9/MovieService.svc/Films(1)", "type":
"Models.Film"
}, "FilmID": 1, "Title": "Kung Fu Panda",
"Description": "Classic martial arts tale",
"Length": 120, "FilmShowings": {
"__deferred": {
"uri": " http://localhost/Chapter9/MovieService.svc/Films(1)/FilmShowings"
}
}
} }
```

Обратите внимание, что результаты упакованы в объект по имени d. Это предотвращает атаки с использованием CSRF (Cross Site Request Forgery — межсайтовая подделка запросов). (Более подробную информацию можно получить в главе 11.)

Использование JSON в C#

В следующем коде показано, как извлекать результаты, сформатированные в виде JSON, с использованием класса HttpWebRequest в C#:

```
System.Net.HttpWebRequest Request =
    (System.Net.HttpWebRequest)System.Net.HttpWebRequest.Create(
      "http://localhost/Chapter9/MovieService.svc/Films(1)"
    );
Request.Method = "GET";
Request.Accept = "application/json";
System.Net.HttpWebResponse Response = (System.Net.HttpWebResponse)Request.
GetResponse();
using (System.IO.StreamReader sr = new System.IO.StreamReader(Response.
GetResponseStream()))
{
    Console.WriteLine(sr.ReadToEnd());
}
Console.ReadKey();
```

Прокси-классы WDS

Хотя во многих случаях работа будет осуществляться с низкоуровневым кодом XML или JSON, возвращенным из службы WDS, все же проще иметь дело со сгенерированными прокси-классами. Такие классы могут существенно облегчить выполнение простых операций CRUD и операций запросов. Разумеется, внутренне они используют HTTP.

Извлечение элементов с помощью прокси-классов

Давацте создадим приложение, которое будет выполнять итерацию по объектам Order, используя LINQ и класс DataServiceContext. Необходимые шаги перечислены ниже.

1. Добавьте к решению новое консольное приложение по имени Chapter9.ADOProxy.
2. Щелкните правой кнопкой мыши на папке References.

3. Выберите в контекстном меню пункт Add Service Reference (Добавить ссылку на службу) для добавления URL, на который настроена служба WDS (например, `http://localhost/Chapter9/MovieService.svc`).
4. Выберите узел `BookEntities` в поле Services (Службы).
5. Введите пространство имен `MovieService` (рис. 9.9).

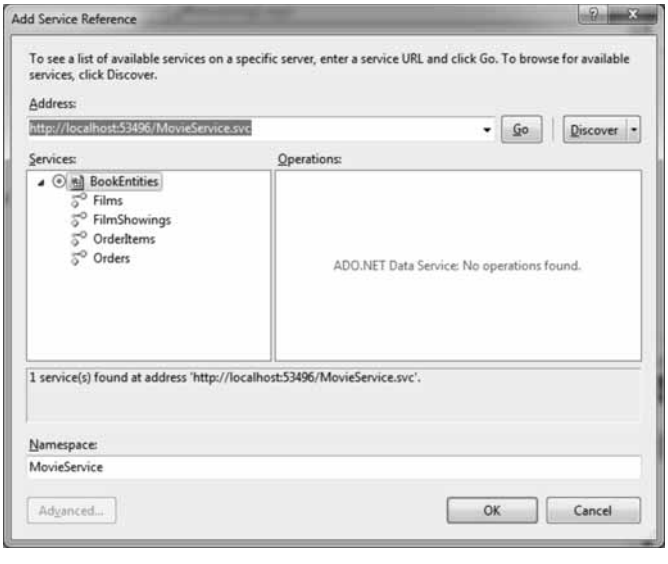

Рис. 9.9. Добавление ссылки на службу WDS

6. Щелкните на кнопке OK. Visual Studio сгенерирует классы, представляющие сущности в консольном приложении.
7. Откройте `Program.cs` и добавьте следующий оператор `using`:

```
using System.Data.Services.Client;
```

8. Введите приведенный ниже код для прохождения по списку заказов `Order` с выводом имени в окно консоли:

```
static void Main(string[] args)
{
  DataServiceContext ctx = new DataServiceContext(
                new Uri("http://localhost/Chapter9/MovieService.svc"));
  var Orders = ctx.Execute<MovieService.Models.Order>(
                new Uri("Orders", UriKind.Relative));
  foreach (MovieService.Models.Order Order in Orders)
  {
     Console.WriteLine(Order.Firstname);
  }
  Console.ReadKey();
}
```

9. Нажмите <F5> для запуска приложения. Вы должны увидеть список имен из таблицы `Orders`.

Добавление нового элемента с помощью прокси-классов

Создать новый элемент, используя WCF Data Services, очень легко. Для этого достаточно воспользоваться методами `AddObject()` и `SaveChanges()` класса `DataServiceContext`:

```
static void Main(string[] args)
{
  DataServiceContext ctx =
    new DataServiceContext(new Uri("http://localhost/Chapter9/MovieService.svc"));
  MovieService.Models.Film NewFilm = new MovieService.Models.Film();
  NewFilm.Title = "Pulp Fiction";
  NewFilm.Length = 124;
  NewFilm.Description = "Quentins classic movie";
  ctx.AddObject("Films", NewFilm);
  ctx.SaveChanges();
}
```

Обновление элемента

Чтобы обновить существующий элемент, загрузите объект и воспользуйтесь методами `UpdateObject()` и `SaveChanges()` класса `DataServiceContext`:

```
static void Main(string[] args)
{
  DataServiceContext ctx =
    new DataServiceContext(new Uri("http://localhost/Chapter9/MovieService.svc"));
  MovieService.Models.Film FilmToUpdate =
    ctx.Execute<MovieService.Models.Film>(
                          new Uri("Films(1)", UriKind.Relative)).First();
  FilmToUpdate.Title = "Updated title";
  ctx.UpdateObject(FilmToUpdate);
  ctx.SaveChanges();
}
```

Удаление элемента

Чтобы удалить элемент, последовательно вызовите методы `DeleteObject()` и `SaveChanges()` класса `DataServiceContext` (если определены табличные ограничения или правила, потребуется обеспечить их выполнение, иначе этот код даст сбой):

```
static void Main(string[] args)
{
  DataServiceContext ctx =
    new DataServiceContext(new Uri("http://localhost/Chapter9/MovieService.svc"));
  MovieService.Models.Film FilmToDelete =
    ctx.Execute<MovieService.Models.Film>(
                          new Uri("Films(1)", UriKind.Relative)).First();
  ctx.DeleteObject(FilmToDelete);
  ctx.SaveChanges();
}
```

WDS 1.5

WDS версии 1.5 входит в состав VS2010 и добавляет ряд средств, таких как возможность ограничения количества возвращаемых элементов данных и генерация веб-каналов (отображений элементов AtomPub).

На заметку! Команда разработчиков даже утверждает, что они собираются реализовать автономный режим WDS.

Подсчет строк и управляемое сервером разбиение на страницы

Ранее одной из основных проблем при работе с WDS было отсутствие возможности узнать количество возвращаемых результатов. Это не позволяло организовать разбиение данных на страницы и эффективную работу со службами WDS. Версия WDS 1.5 предлагает возможность запросить количество записей в результате через операции `$count` и `$inlinecount`.

$count

Операция `$count` вернет одиночное текстовое значение, которое указывает количество строк, возвращаемых определенным запросом. Ниже показано, как вернуть количество заказов:

```
http://localhost/Chapter9/MovieService.svc/Orders/$count
```

Обратите внимание, что `$count` применяется к набору из 100 элементов. Кроме того, используются операции `$skip`, `$top` или `$take`, после чего возвращенное число представляет собой количество *после* применения операций `$skip` или `$take`.

$inlinecount=allpages

Операция `$inlinecount=allpages` возвращает счетчик элементов вместе с текущим результатом запроса. Это вернет количество элементов с игнорированием любых операций `$skip` или `$take` (в отличие от `$count`). Значение количества содержится в новом элементе `m:count` (`m` здесь означает "metadata" (метаданные)). Ниже демонстрируется использование этого средства в таблице `Orders`:

```
http://localhost/Chapter9/MovieService.svc/Orders?$inlinecount=allpages
```

В коде XML будет присутствовать новый элемент, показывающий встроенное значение счетчика:

```
<m:count>4</m:count>
```

Ограничение количества возвращаемых результатов

Ограничение количества результатов, возвращаемых вызовом WDS, осуществляется с помощью нового метода `Config.SetResourcePageSize()`. Следующий код ограничивает запрос к таблице `Orders` только двумя результатами:

```
config.SetEntitySetPageSize("Orders", 2);
```

При наличии большего числа результатов, чем указанный размер страницы, то WDS также сделает доступной следующую ссылку, через которую можно извлечь последующие страницы:

```
<link rel="next"
  href="http://localhost/Chapter9/MovieService.svc/Orders?$skiptoken=2" />
```

В настоящее время не существует поддержки этой функциональности на стороне клиента, хотя команда разработчиков WDS утверждает, что они намерены добавить ее в будущем выпуске.

Проекции

Проекции позволяют сообщить службе WDS о необходимости возврата только определенных типов элементов и создаются добавлением операции `$select` к запросу. Например:

```
http://localhost/Chapter9/MovieService.svc/Orders?$select=Firstname,Lastname
```

Обратите внимание, что для использования этого средства должна быть активизирована версия 2 служб WDS:

```
config.DataServiceBehavior.MaxProtocolVersion = DataServiceProtocolVersion.V2;
```

При использовании проекций можно также применять сложные типы и свойства навигации:

```
http://localhost/Chapter9/MovieService.svc/
  Orders?$select=Firstname,Lastname,OrderItems
```

Предыдущий запрос возвращает ссылку на сложный тип или свойство навигации. Для возврата деталей от связанного элемента служит операция `$expand`. Следующий запрос возвращает `Firstname` и `Lastname`, а также детали `OrderItems`:

```
http://localhost/Chapter9/MovieService.svc/
  Orders?$select=Firstname,Lastname,OrderItems&$expand=OrderItems
```

Дружественные каналы

Теперь можно отображать свойства модели EF на индивидуальные элементы AtomPub. В рассматриваемом примере с фильмами можно было бы отобразить имя режиссера на AtomPub-элемент `author`. Можно также создавать специальные классы и управлять форматированием вывода. За дополнительной информацией об этих средствах обращайтесь на `http://blogs.msdn.com/astoriateam/archive/2009/09/01/customizable-feed-support-in-ctp2.aspx`.

Прочие усовершенствования

Команда разработчиков WDS также внесла и ряд других усовершенствований.

- WDS 1.5 поддерживает двунаправленную привязку данных (`DataServiceCollection`), что замечательно для сценариев WPF/Silverlight. За дополнительной информацией обращайтесь в главу 15 и к следующей статье в блоге: `http://blogs.msdn.com/astoriateam/archive/2009/09/17/introduction-to-data-binding-in-silverlight-3-with-ctp2.aspx`
- Улучшенная поддержка потоковой передачи крупных массивов двоичных данных.

Какая связь между службами WDS и WCF RIA Services?

Скотт Гатри (Scott Guthrie) говорил об этом на RedDevNews.com (еще до изменения названия служб):

> ***Следует ли отдавать предпочтение службам .NET RIA Services перед ADO.NET Data Services при доступе к данным Silverlight Data Access?***
>
> *Нет. То, что реализовано сегодня для RIA Services, построено на базе ADO.NET Data Services. Поэтому к ADO.NET Data Services можно относиться, как к низкоуровневому API-интерфейсу RAW/REST, а к RIA Services — как к верхнему уровню.*

Мы определенно полагаем, что бывают сценарии, когда понадобится иметь дело с чистой моделью службы REST. Службы .NET RIA Services предоставляет такие вещи, как проверка достоверности, межуровневое взаимодействие и высокоуровневые службы. Мы усердно потрудились, чтобы должным образом упорядочить их, так что службы RIA Services не являются конкурирующей технологией, и на самом деле они построены на основе ADO.NET Data Services.

`http://reddevnews.com/articles/2009/07/13/interview-with-scott-guthrie-on-silverlight-3.aspx`

Резюме

Службы WCF Data Services существенно облегчают представление данных в сценариях интеграции с третьими сторонами. WDS могут также использоваться как интеграционный уровень для проектов веб-приложений и проектов Silverlight. Последний выпуск WCF Data Services содержит столь необходимые средства разбиения на страницы и проекции, так что выглядит весьма удобной технологией. Заинтересованным разработчикам также стоит обратить внимание на построенные поверх WDS службы WCF RIA Services, которые добавляют множество дополнительных средств.

Материалы для дополнительного чтения

- `http://blogs.msdn.com/astoriateam/`
- `http://msdn.microsoft.com/en-us/library/cc907912.aspx`