АРХИТЕКТУРА СЕРВЕРНОЙ ЧАСТИ ОБЛАЧНОГО СЕРВИСА ПОДГОТОВКИ И ПРОВЕДЕНИЯ СОЦИОЛОГИЧЕСКИХ ОПРОСОВ

Тищенко Р.В., Ахриев Р.А., студенты кафедры АОИ

Научный руководитель: Сидоров А.А., канд. экон. наук, доцент кафедры АОИ

Разработка web-серверов на базе ASP.NET является одним из трудоёмких процессов, т.к. он включает в себя много задач, таких как разработка бизнес-логики, проектирование баз данных, создание и тестирование функционала. Первый этап создания любого web-сервера начинается с разработки бизнес-логики, на основе которой проектируются базы данных. При проектировании баз данных разработчики очень часто сталкиваются с проблемой выбора технологии. Классической технологией проектирования баз данных для платформы .NET является ADO.NET. Она работает с помощью объектов DataSet, которые предоставляют локальные копии любого количества взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Вместе с тем ADO.NET привязана к физической структуре данных, при взаимодействии с данными необходимо помнить схемы таблиц, отношений. Общение с БД осуществляет на языке SQL, что приводит к большому объёму кода, так как язык C# сильно отличается от SQL.

С данной проблемой сталкиваются все проекты, за исключением небольших. Проект web-сервер «Интервьюер» (рабочее название «CodeSurvey») системы облачного сервиса подготовки и проведения социологических опросов не стал исключением. Разрабатываемая система должна упростить всю работу путём избавления от опросов в печатном виде и их ручного анализа. В целом разрабатываемая система облачного сервиса подготовки и проведения социологических опросов представляет собой:

- сайт, на котором пользователь создаёт свои опросы;
- сервер, служащий хранилищем и обработчиком опросов;
- мобильное приложение для платформ Android и iOS, которое является клиентской частью для прохождения опросов, сформированных на сайте, и отправки данных на сервер;

Альтернативной технологией ADO.NET является ADO.NET Entity Framework. Данный фреймворк является более высоким уровнем абстракции, который позволяет не думать о базе данных и работать с данными независимо от типа хранилища на языке C#.

Первая версия Entity Framework-1.0 вышла еще в 2008 году и представляла очень ограниченную функциональность, базовую поддержку ORM (object-relational mapping – отображения данных на реальные объекты) и один единственный подход к взаимодействию с базами данных – Database First. С выходом версии 4.0 в 2010 году многое изменилось – с этого времени Entity Framework стал рекомендуемой технологией для доступа к данным. Дополнительные улучшения функционала последовали с выходом версии 5.0 в 2012 году. И наконец, в 2013 году был выпущен Entity Framework 6.0, обладающий возможностью асинхронного доступа к данным.

Центральной концепцией Entity Framework является понятие сущности (entity). Сущность представляет набор данных, ассоциированных с определенным объектом. В связи с этим данная технология предполагает работу не с таблицами, а с объектами и их наборами. Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст, вес. Свойства необязательно представляют простые данные типа int, но и могут представлять более комплексные структуры данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять её. Подобные свойства называют ключами. При этом сущности могут быть

связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи. Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД.

С помощью LINQ можно не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Другим ключевым понятием является Entity Data Model. Эта модель сопоставляет классы сущностей с реальными таблицами в БД. Entity Data Model состоит из трех уровней: концептуального, уровень хранилища и уровень сопоставления (маппинга). На концептуальном уровне происходит определение классов сущностей, используемых в приложении. Уровень хранилища определяет таблицы, столбцы, отношения между таблицами и типы данных, с которыми сопоставляется используемая база данных. Уровень сопоставления (маппинга) служит посредником между предыдущими двумя, определяя сопоставление между свойствами класса сущности и столбцами таблиц. Таким образом, можно через классы, определенные в приложении, взаимодействовать с таблицами из базы данных.

Entity Framework предполагает три возможных способа взаимодействия с базой данных: [1]

- Database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных.
- Model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере.
- Code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В результате анализа было выяснено, что Entity Framework упрощает работу с базами данных, избавляя от нужды использовать SQL и позволяя работать с данными в виде объектов (сущностей) используя язык C#.

Для реализации Entity Framework в проекте были созданы следующие модели данных (классы) для опросов:

- Surveys – модель данных для опроса;
- QuestionBase – модель данных для вопроса;
- AnswerOption – модель данных для ответа;
- FilledUpForm – модель данных для пройденного опроса;
- QuestionAnswerBase – модель данных для отвеченных вопросов;

Это обычные классы, которые содержат в себе некоторое количество автосвойств. Каждое свойство будет сопоставляться с отдельным столбцом в таблице из базы данных. Для взаимодействия с базами данных нужен контекст данных, это своего рода посредник между базами данных и классами, в нашем случае это класс ApplicationDbContext.

В данном классе реализован Entity Data Model (метод OnModelCreating), а именно уровень сопоставления (маппинга). В конструкторе этого класса вызывается конструктор базового класса, в который передается строка «DefaultConnection» – это имя будущей строки подключения к базе данных. Также в классе определены свойства Surveys, Questions, FilledUpForms, QuestionAnswers, которые будут хранить наборы объектов соответствующего типа. Через эти свойства будут осуществляться связь с таблицей объектов в базе данных. В проекте «CodeSurvey» используется паттерн репозиториев, поэтому метод сохранения данных в базу происходит в классе репозитория SurveyRepository.

В конструкторе данного класса создается объект класса ApplicationDbContext для работы с контекстом данных, с режимом «только чтение», т.к. данные контекста не должны меняться в ходе работы. Для добавления данных в базу данных используется метод AddIfIsNew, который принимает 2 аргумента. Первый это сама «сущность», как правило, это объект соответствующий одной из модели данных, второй это хранилище сущностей, которое должно соответствовать нашему контексту данных (пример: AddIfIsNew(survey, _dbcontext.Surveys)) [2].

Подводя итоги, можно сказать, что Entity Framework очень эффективен и компактен, в отличие от его предшественников. Для его реализации достаточно иметь модель данных, контекст данных и метод, который будет реализовывать весь функционал.

Список литературы

1. Руководство по ADO.NET Entity Framework 6 [Электронный ресурс]. – Режим доступа: http://metanit.com/sharp/entityframework/ (дата обращения: 30.11.2016)

2. Entity Framework Code First на практике [Электронный ресурс]. – Режим доступа: https://habrahabr.ru/post/236037/ (дата обращения: 30.11.2016)