**РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ ГЕНЕРАЦИИ СЕМАНТИЧЕСКИХ ДАННЫХ**

Апсаликов М.Ю.

*Саратовский государственный технический университет  
имени Гагарина Ю.А., Россия,* dee\_jay@bk.ru

*Данная статья посвящена описанию основной проблемы развития Семантического Веб и варианта его частичного решения. В статье даны краткое описание понятия Семантический Веб, библиотек Entity Framework, ASP.NET WebAPI и подхода Entity Framework Code First, на основе которых планируется решение. Приведен анализ аналогичных программных средств. Обосновывается необходимость создания библиотеки, связывающей технологии Семантического Веб, Entity Framework Code First и ASP.NET WebAPI для их использования в коммерческих проектах. Кратко описан план реализации библиотеки.*

**SEMANTIC DATA GENERATION FRAMEWORK DEVELOPMENT**

Apsalikov M.

*Yuri Gagarin State Technical University of Saratov, Russia,* dee\_jay@bk.ru

*The article describes the main development problem of the Semantic Web and a possible partial solution of it. The article gives a brief description of the Semantic Web concepts as well as Entity Framework, ASP.NET WebAPI libraries and Entity Framework Code First those are going to be used for the solution. There is necessity of creating the framework binding Semantic Web, Entity Framework Code First and ASP.NET WebAPI technologies for using in production. There is a framework development plan as well.*

Семантический Веб (Semantic Web) – это стек технологий, существующих над Всемирной паутиной (World Wide Web), который позволяет размещать информацию в сети Интернет в виде, пригодном для машинной обработки[1]. В частности, это реализовано с помощью технологий RDF и OWL. Где RDF – это модель представления данных, а также специальный язык, который позволяет описать любую информацию в виде триплетов: субъект – предикат – объект[2]. OWL, в свою очередь, позволяет описывать классы и их взаимоотношения. Главное достоинство Семантического Веба – это возможность семантического поиска информации, которая основана на логическом выводе. Семантическая и Всемирная паутина работают на протоколах HTTP/HTTPS и используют в качестве идентификаторов ресурсов URI (Uniform Resource Identifier).

Вся информация в Семантическом Веб предоставлена в виде триплетов. В то же время, во Всемирной паутине информация предоставляется технологиями HTML, CSS и JavaScript. Это означает, что если компании нужен программный продукт, поддерживающий Семантические технологии, то при его написании, необходимо будет написать два разных слоя представления: один для человека, а другой – машиночитаемый. Следовательно, это значительно увеличивает время и стоимость разработки. Как следствие, компании отказываются от использования Семантических технологий в своих проектах.

Если посмотреть на современные стандарты веб-разработки, то можно увидеть, что достаточно большую долю рынка крупных веб-приложений занимает язык C# и его стандартной библиотекой ASP.NET[3]. Самыми современными средствами на момент написания статьи являются ASP.NET MVC 5 и ASP.NET WebAPI 2. Для большинства разрабатываемых приложений сегодня выбирают эти библиотеки.

Для работы с базами данных в программных продуктах, написанных на C#, в том числе и коммерческих, часто используется ORM Entity Framework. Это библиотека, которая позволяет работать с базами данных используя три подхода: Database First, Model First и Code First. Database First позволяет по готовой базе сгенерировать модель, Model First – по готовой модели сгенерировать базу. Эти два подхода считаются устаревшими, так как имеют недостаток: при изменении схемы базы данных необходимо модель пересоздавать. Таким образом, актуален сейчас только один подход Code First. При данном подходе модель собирается из классов C#. Это позволяет отследить изменения модели и применять их (по возможности) автоматически, либо с помощью миграций.

Вследствие всего вышеуказанного, в коммерческих проектах очень часто можно встретить связку ASP.NET + Entity Framework. Более того, разработка логики на этих технологиях занимает в среднем меньше времени по сравнению с аналогичными технологиями на рынке.

Вернемся к Семантическому Веб. Как было сказано ранее, основной его проблемой является повторное написание кода. В качестве решения этой проблемы можно рассматривать вариант, когда код хотя бы одного из слоёв представления сгенерируется автоматически. С пользовательским интерфейсом в коммерческих приложения это невозможно, потому что бизнес всегда будет диктовать новые требования и постоянно модифицировать HTML, CSS и JavaScript код. А вот с семантическим предоставлением информации – это возможно. Так как и модель базы данных, и семантическое хранилище являются машиночитаемыми. Иными словами, идея заключается в том чтобы создать библиотеку под платформу .NET, которая в процессе работы готового коммерческого приложения генерирует семантическое хранилище из базы данных. Далее, параллельно с обычным сервисом WebAPI можно разместить семантический сервис, который будет отдавать клиентам данные в формате RDF. Конвертация данных будет произведена с помощью библиотеки dotNetRdf, которая предоставляет удобный интерфейс для работы с семантическими данными, а именно создание, редактирование и конвертация между диалектами.

В данном решении есть несколько моментов, требующих решения. Во-первых, в отношениях баз данных есть только две сущности, но нет аналога предиката. Эта проблема может быть решена с помощью аттрибутов в языке C#. Таким образом, рабочий процесс конвертации сущности в триплет будет выглядеть следующим образом: субъект равен сущности, к которой было произведено обращение; предикат равен информации из аттрибута, который необходимо заполнить разработчику; объект равен сущности, с которой субъект связан. Возможны несколько вариантов, если разработчик не укажет предикат. Самый простой вариант – это предотвращение компиляции приложения. Также возможно сгенерировать уникальный предикат по умолчанию. В разрабатываемой библиотеке планируется реализовать оба варианта и выбирать из них на основе конфигурационного файла.

Во-вторых, проблемы архитектуры приложения. Здесь планируется внедрить в библиотеку шаблон проектирования Репозиторий[4], используя его с обобщенными типами. Этот подход был выбран в связи с тем, что у него нет проблем с расширяемостью. Это значит, что у каждой сущности при импорте библиотеки уже будет набор операций создания, удаления, получения и модификации. Кроме того, данный паттерн планируется расширить для создания контроллеров для сервисов WebAPI и Семантического сервиса. Однако, разработчику приложения необходимо знать, что логика сервисов не должна иметь несоответствий.

В-третьих, так как RDF имеет несколько различных сериализаций или диалектов, то необходимо задуматься об их реализации в библиотеке. В данном случае проблема решается с помощью библиотеки dotNetRdf, о которой было сказано ранее. Однако, в целях расширяемости приложения, выбор диалекта необходимо вынести в конфигурационный файл.

Если задуматься о рабочем процессе использования библиотеки, то можно обнаружить два варианта. Если приложение разрабатывается с нуля, то разработчикам достаточно добавить аттрибуты предикатов на все отношения в модели базы. Если существует уже готовое приложение, необходимо перенести логику в репозитории и заменить сервисы. В любом случае классы сущностей не будут затронуты. Очевидно, такой рефакторинг кода займет некоторое время и ресурсы. Однако, в большинстве случаев, на рефакторинг будет затрачено значительно меньше ресурсов, чем на ручную разработку и поддержку второго слоя представления.

В течение одного года планируется реализовать вышеописанную библиотеку. Разработка будет производиться с использованием IDE Visual Studio 2013 Ultimate, SQL Server Management Studio для работы с базой данных и Git в качестве системы контроля версий. Также планируется реализовать прототип коммерческого приложения из 5-10 сущностей для демонстрации работы библиотеки.

Огромное преимущество данного подхода состоит в том, что семантические данные всегда актуальны и их не требуется обновлять вручную или синхронизировать. Это достигается за счет того, что при запросе к семантическому сервису данные поступают напрямую из базы данных. В качестве недостатка можно указать, что Entity Framework работает только на одной платформе .NET и данную библиотеку невозможно использовать при разразработке на других платформах, таких как Java, PHP, Python, node.js и других.

Список использованных источников:

1. Berners-Lee, T. The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities [Электронный ресурс] / T. Berners-Lee, J. Hendler, O. Lassila. Режим доступа: http://jeckle.de/files/tblSW.pdf (дата обращения 11.07.2015).
2. Klyne, G. Resource Description Framework (RDF): Concepts and Abstract Syntax [Электронный ресурс] /G. Klyne, J. Carroll. − Режим доступа: http://www.w3.org/TR/rdf-concepts/ (11.07.2015).
3. World Wide Web Technology Surveys [Электронный ресурс] / W3Techs. – Режим доступа: http://w3techs.com/technologies/overview/programming\_language/all/ (11.07.2015)
4. Microsoft Development Network: The Repository Pattern [Электронный ресурс] / MSDN. Режим доступа - https://msdn.microsoft.com/en-us/library/ff649690.aspx (11.07.2015).

Сведения об авторе

Апсаликов Михаил Юрьевич

Студент 1 курса магистратуры направления «Информатика и вычислительная техника» кафедры «Информационные системы и технологии»

Junior Software Engineer. EPAM Systems.

Email: dee\_jay@bk.ru