

## **ОСОБЕННОСТИ ПОСТРОЕНИЯ ПРИКЛАДНЫХ ПРОГРАММНЫХ СИСТЕМ НА ОСНОВЕ ОНТОЛОГИЧЕСКИХ БАЗ ЗНАНИЙ**

Выделяются и анализируются этапы разработки приложения на основе онтологического хранилища знаний, проводится аналогия с процессом создания приложения, основанного на реляционной БД. Показывается принципиальная близость этих подходов и акцентируются значимые отличия.

### **Введение**

В настоящее время онтологические системы могут быть применены для решения различных задач в сфере представления знаний в Интернет и создания интеллектуальных систем. Подобные системы, основанные на знаниях (Knowledge Based Systems), используют методы искусственного интеллекта для помощи в принятии решений, обучения и проч. Достоинством онтологических систем является универсальное представление информации, позволяющее применять одни и те же знания в разных предметных областях и разных задачах. Система, основанная на онтологическом принципе, легко расширяется. Такие системы позволяют проводить интеллектуальный анализ и обработку информации, используя всю выразительную мощь таких языков как SWRL и SPARQL.

Недостатком онтологических систем является большая вычислительная сложность при работе на больших объемах данных. Учитывая последние тенденции при создании серверов баз знаний и анализ, проведенный в [1], видно, что для построения производственных систем, основанных на онтологиях, возможно использовать такие RDF-хранилища как Virtuoso, которые делают применение онтологических хранилищ близкими по производительности и сложности к использованию традиционных СУБД.

*Целью* исследования является разработка метода создания производственных программных продуктов на основе онтологических хранилищ знаний. Такие приложения потенциально имеют следующие преимущества:

- масштабируемость относительно количества источников данных и размера онтологии;
- расширяемость (минимизация затрат, связанных с адаптацией приложения при расширении схемы базы знаний);
- облегчение интеграции онтобазированных приложений в корпоративные информационные системы в связи с использованием универсальной стандартизированной модели представления знаний, не зависящей от конкретного приложения.

Для достижения поставленной цели были сформулированы следующие задачи:

- исследование возможностей и методик внедрения онтологических хранилищ знаний в высоконагруженные программные продукты, формирование универсального метода;
- апробация предложенного метода на тестовом приложении.

### **1. Анализ предметной области**

Шаблон MVC стал де-факто стандартом при проектировании приложений, основанных на данных [2]. В нем Model – абстракция, соответствующая графу данных предметной области, инкапсулирует бизнес-логику их обработки, а также в некоторых интерпретациях – логику сохранения/восстановления состояния модели между перезапусками системы (рис. 1).

Часто среды разработки предоставляют целые фреймворки (ORM) для автоматической генерации кода, реализующего логику сохранения/восстановления состояния модели на основе традиционных источников данных, например, таких как реляционные базы данных. Так (среди прочего), для платформы .NET разработан EntityFramework; для Java – Hibernate, JPA; для Python – Django.

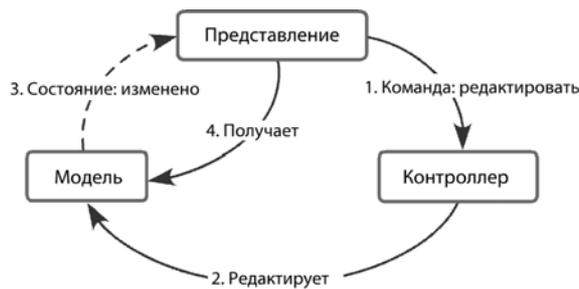


Рис. 1. Шаблон проектирования MVC

К сожалению, для инновационных в настоящее время источников данных (таких, как RDF-хранилища) выбор ORM-библиотек и их производительность значительно отстает. Например, известно решение на основе Tinkerpop, заточенное под Neo4J [3] или jenabean [4]. Но поскольку результат сравнения RDF-хранилищ [1] позволил выделить явного лидера в этой области Openlink Virtuoso, то практический интерес представляют способы обращения к графам, поддерживаемые именно этим сервером.

Доступ к RDF-хранилищу Virtuoso осуществляется с помощью языка запросов SPARQL. Разработчики Virtuoso предоставляют несколько вариантов подачи SPARQL запросов к серверу (рис.2): 1) через HTTP REST Endpoint; 2) через традиционные API, такие как ODBC, JDBC, .NET или OLE/DB (Virtuoso имеет возможность исполнять SPARQL запросы, являющиеся составной частью SQL); 3) через специализированные RDF Data-провайдеры Virtuoso посредством API распространенных RDF-фреймворков (Jena, Sesame, Redland, Linq2Rdf, dotnetRDF).

Как первое, так и второе – довольно низкоуровневые по сравнению с ORM подходы, подразумевающие большой объем рутинной работы для повышения уровня абстракции (в первом случае добавляется еще и низкоуровневая работа с протоколом HTTP). В третьем случае мы имеем дело с мощным API, по своим возможностям значительно перекрывающим классические задачи ORM. Хотя при оптимальной реализации первый подход может показать значительный выигрыш в производительности, по сравнению с последним, третий вариант по своей сути более близок к традиционным ORM-библиотекам. Этот вариант и был выбран базовым.

В качестве тестового приложения выбрана задача по созданию универсального модуля для проведения опросов. Языком программирования выступает Java, а в качестве среды Eclipse SDK. RDF-хранилище создано на Openlink Virtuoso. Доступ к онтологии производится через java-фреймворк Jena. Для доступа к RDF был применен наиболее высокоуровневый вариант через специализированный RDF Data-провайдер для Jena. Интерфейс приложения создается при помощи GWT (Google Web Toolkit).

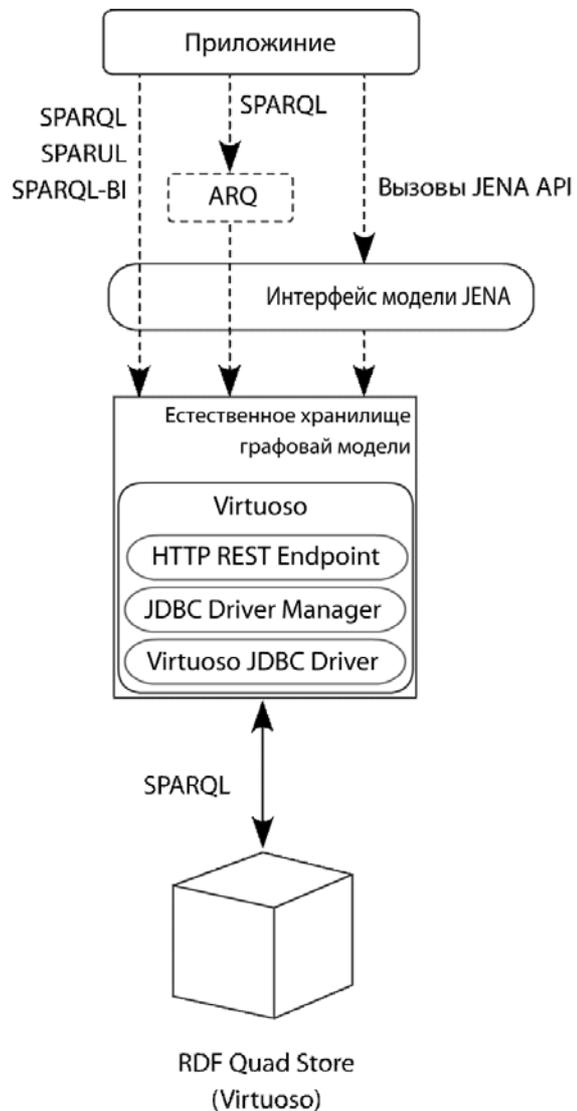


Рис. 2. Интерфейсы доступа к RDF-хранилищу Virtuoso

## 2. Метод создания программных продуктов на основе онтологических баз знаний

Все имеющиеся вопросы вместе с вариантами ответов предполагается хранить в виде OWL онтологии, сюда же будут заноситься ответы пользователей. Посредством SPARQL запросов вопросы с вариантами ответов будут выбираться из хранилища для отображения пользователю. Таким же образом результаты опроса будут возвращаться обратно в хранилище. Далее перечислены этапы, которые разработчик должен выполнить для реализации приложения, основанного на онтологиях.

**Этап 1. Создание схемы онтологии.** С помощью Protege была смоделирована предметная область. Выделен класс «Question», соответствующий одному вопросу. Класс «ResponseOption» представляет вариант ответа на конкретный вопрос; соответствие варианта ответа и вопроса задает слот «questionLink», ссылающийся на экземпляр класса «Question».

Класс «User» представляет в онтологии сущность «пользователь», конкретные его экземпляры (instances) – это пользователи, принимавшие участие в опросе. Каждый экземпляр класса «User» кроме данных о пользователе содержит и ответы на вопросы в виде ссылок на выбранный вариант ответа для каждого вопроса. Готовая онтология сохранена в формате OWL (рис.3).

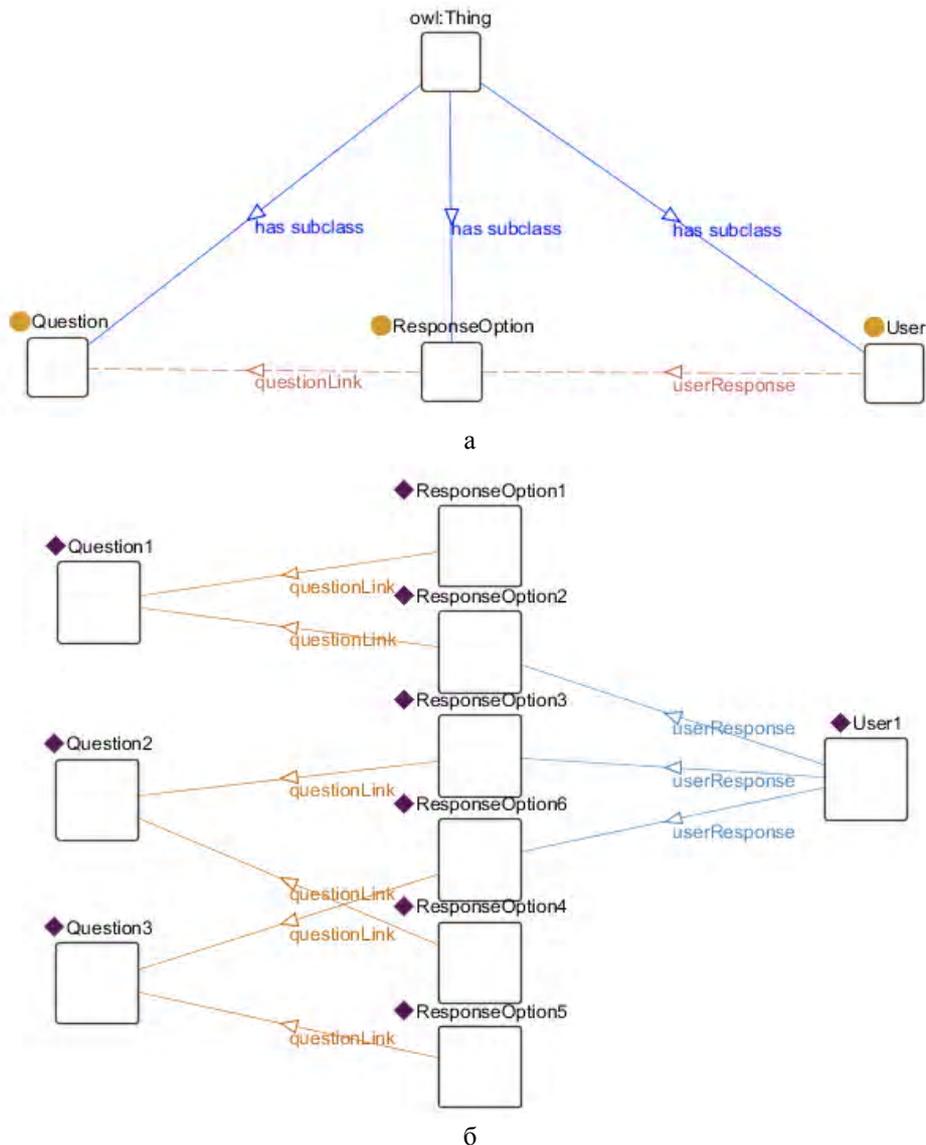


Рис. 3. Граф из редактора онтологий: а – только TBox; б – только ABox

**Этап 2. Интеграция онтологии в Virtuoso.** После загрузки и установки Virtuoso необходимо импортировать созданную онтологию в хранилище. Это осуществляется через консоль управления, которая при стандартных настройках после запуска Virtuoso Service Manager доступна по адресу <http://localhost:8890/>.

Для загрузки онтологии следует воспользоваться WebDAV Browser, где для любого пользователя возможно создать папку WebDAV-типа и в ней разместить данные в формате RDF или OWL, а затем загрузить их в хранилище триплетов. Необходимо помнить, что в Virtuoso каждая запись онтологии имеет дополнительный атрибут – идентификатор графа, которому принадлежит триплет. Поэтому на самом деле хранилище Virtuoso представляет собой quad store. Это позволяет содержать в Virtuoso множество отдельных онтологий-задач.

Далее нужно проверить правильность загрузки онтологии следующим образом: необходимо перейти по адресу <http://localhost:8890/sparql/>, в качестве URI графа по умолчанию указывается путь к онтологии. Для стандартного пользователя путь будет выглядеть так: [http://localhost:8890/DAV/home/dba/rdf\\_sink/](http://localhost:8890/DAV/home/dba/rdf_sink/), где dba – это имя пользователя. В поле ввода SPARQL запроса можно вписать, например: “SELECT DISTINCT ?p WHERE {?s ?p ?o} LIMIT 10”, в результате выполнения данного запроса должен отобразиться список предикатов онтологии с ограничением не более десяти.

**Этап 3. Разработка архитектуры приложения.** Внутренняя структура приложения, использующего онтологическое хранилище, ничем не ограничена и может быть построена согласно любой методологии. Единственное специфическое требование – это необходимость создания прослойки доступа к знаниям. Для взаимодействия с сервером, на котором расположен репозиторий онтологии, потребуются библиотеки фреймворка Jena, которые можно скачать на сайте OpenLink Software [5]. Общая модель архитектуры приложения приведена на рис. 4.



Рис. 4. Схема разработанного приложения в терминах MVC

Для упрощения работы с данными онтологии были созданы серверный класс «Question» и «ResponseOption» - элементы модели, отображающиеся на аналогичные классы в онтологии, в которых в виде объектной модели уровня приложения хранятся вопросы с вариантами

ми ответов и ссылки на эти данные в онтологии. Таким образом, упрощается процедура отображения вопросов и вариантов ответов на веб-форме, а также отправка результатов опроса на сервер и записи их в онтологию.

Как уже было сказано, для связи с репозиторием ORM-контроллер применяет фреймворк Jena. Он инкапсулирует использование библиотек Jena, производит установку связи с сервером и выполнение запросов для получения или отправки данных на сервер.

Ниже приведен пример запроса на получение списка вопросов из репозитория:

```
String queryStringQuestions =  
    "PREFIX table: <http://www.owl-ontologies.com/interview.owl#>" +  
    "select * from <http://localhost:8890/DAV/home/dba/rdf_sink/>" +  
    "where {?s table:question ?o}";
```

API отправки запроса на сервер и получение результатов аналогично стандартным классам для выполнения SQL-запросов:

```
queryExec = new QueryEngineHTTP(endpoint, queryStringQuestions);  
resultQuestions = queryExec.execSelect();
```

Здесь *endpoint* – аналог сущности Connection для работы с СУБД.

Аналогично производится запрос на добавление триплетов в онтологию:

```
String queryString =  
    "PREFIX table: <http://www.owl-ontologies.com/interview.owl#>" +  
    "INSERT INTO GRAPH <http://localhost:8890/DAV/home/dba/rdf_sink/> "+  
    "{table:User1 table:userResponse ?o." +  
    " table:User1 table:userResponse table:ResponseOption6.}";
```

Здесь под параметром ?o подразумевается объект, который будет добавлен в хранилище как свойство «вариант ответа» (table:userResponse) определенного пользователя (table:User1). Строка с параметром определяет шаблон, по которому сначала находится «место» в онтологии, куда необходимо добавить объект. В нашем случае пользователю добавляем вариант ответа.

**Этап 4. Создание интерфейса системы опросов при помощи GWT.** Технология создания пользовательского интерфейса, по сути, значения не имеет. Поскольку созданное тестовое приложение является веб-приложением, бизнес-логика приложения заключена в сервлет. Для создания интерфейса был выбран фреймворк Google Web Toolkit. На веб-форме располагаются поля для отображения вопроса и выпадающие списки с вариантами ответа. Элементы интерфейса формируются динамически в зависимости от количества вопросов в репозитории, количество вариантов ответа также может быть любым. На рис. 5 отображена веб-форма приложения для ввода ответов.

The image shows a web form titled "Qualification survey" with a light gray header. Below the header are seven numbered questions, each followed by a dropdown menu. The questions and their corresponding dropdown values are:

- 1) **Lecture material corresponds to the current state of knowledge in this subject area** (Good)
- 2) **Effective use of modern learning technologies (multimedia, telecommunications, etc.)** (Unsatisfactorily)
- 3) **Material is explained clearly, uses examples** (Satisfactorily)
- 4) **Involves students in a discussion** (No)
- 5) **Able to generate interest of the audience to the subject** (Unsatisfactorily)
- 6) **Change the tasks according to the degree of preparedness of students and their abilities** (Satisfactorily)
- 7) **Evaluates the knowledges objectively** (Do not know)

At the bottom of the form is a "Submit" button.

Рис. 5. Интерфейс системы опросов

**Выводы.** Практическая значимость предложенного в статье метода состоит в том, что он позволяет повысить уровень интеллектуализации высоконагруженных производственных приложений путем использования онтологических хранилищ знаний. Дальнейшие исследования должны быть направлены на оценку производительности таких приложений и поиск путей её повышения.

**Список литературы:** **1.** *Europeana* RDF Store Report [Текст]: The results of qualitative and quantitative study of existing RDF stores in the context of Europeana / Multimedia Information Systems Research Group, CS Faculty, University of Vienna; В. Haslhofer. Vienna, 2011. 30 p. **2.** Фаулер М. Шаблоны корпоративных приложений [Текст]/Фаулер М. М.: «Вильямс», 2009. С. 544. **3.** *RDF* data in Neo4J – the Tinkerpop story / Портал DZone для разработчиков. Режим доступа: [www / URL: http://java.dzone.com/news/rdf-data-neo4j-tinkerpop-story](http://java.dzone.com/news/rdf-data-neo4j-tinkerpop-story). 20.04.2012. Загл. с экрана. **4.** *Binding* Java Objects to RDF / Портал группы-участника W3C SemanticWeb. Режим доступа: [www / URL: http://semanticweb.com/binding-java-objects-to-rdf\\_b10682?red=su](http://semanticweb.com/binding-java-objects-to-rdf_b10682?red=su). 20.12.2011. Загл. с экрана. **5.** *OpenLink* Virtuoso Open-Source Edition: Downloads/ Официальный сайт Openlink Software. Режим доступа: [www / URL: http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSDownload#Jena%20Provider](http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSDownload#Jena%20Provider). 20.12.2011. Загл. с экрана.

*Поступила в редколлегию 16.11.2011*

**Шевченко Елена Леонидовна**, канд. техн. наук, доцент кафедры ПИ ХНУРЭ. Научные интересы: онтологический инжиниринг, формальные методы в разработке ПО. Адрес: Украина, 61000, Харьков, ул. Космическая, 11, кв. 37, тел. 0505748372.

**Шевченко Александр Юрьевич**, канд. техн. наук, доцент кафедры ИИ ХНУРЭ. Научные интересы: онтологический инжиниринг, распределенные интеллектуальные системы. Адрес: Украина, 61000, Харьков, ул. Космическая, 11, кв. 37, тел. 0506192638.

**Богдан Александр Николаевич**, студент, гр. КН-08-4, ХНУРЭ. Научные интересы: онтологический инжиниринг. Адрес: Украина, 61000, Харьков, ул. Блюхера, 42, кв. 96, тел. 0671201964.