Разработка системы публикации расходов бюджета Санкт-Петербурга

О.В. Пархимович

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики
olya.parkhimovich@gmail.com

Аннотация

В статье рассмотрены этапы разработки системы публикации открытых государственных данных ведомственной структуры расходов бюджета Санкт-Петербурга, целью которой является загрузка, отображение и визуализация массивов данных.

1. Введение

В последнее время тема открытых государственных данных становится все более актуальной, причем не только в зарубежных странах, но и в Российской Федерации. Например, согласно докладу Эльвиры Набиуллиной на заседании президиума Совета при Президенте по развитию информационного общества «Минэкономразвития России считает необходимым развивать тематику открытых государственных данных и планирует проводить с 2012 года ряд работ в рамках государственной программы «Информационное общество (2011-2020 годы)» [1].

Под открытыми государственными данными (ОГД) в мировой практике обычно понимают публичную государственную информацию, предоставляемую в цифровом виде посредством сети Интернет в форме, допускающей последующий анализ и ее повторное использование. Таким образом, благодаря наличию ОГД становится возможным создание сервисов и проектов, решающих отдельные проблемы граждан, появление новых возможностей для коммерческой деятельности, подкрепляющей экономику государства, повышение эффективности работы государственных органов и увеличение прозрачности управления государством.

Ценность информации зависит не только от ее содержания, но и от формата, в котором она представлена. Именно формат данных определяет значимость ресурса и область, в которой их можно повторно использовать. Например, обычный текст можно только читать, а таблицы – анализировать, то есть упорядочивать, сортировать, применять фильтры. Для того чтобы государственную информацию можно было сортировать, преобразовывать или искать, она должна публиковаться в машинно-обрабатываемом формате. Выбор формата важен, так как некоторые из них могут устареть, быть защищены патентными лицензионными или технологическими ограничениями.

С помощью компьютерных технологий можно осуществлять поиск по информации, ее сортировку и преобразование. Но основная проблема в том, что с помощью них нельзя обрабатывать любой файл. Если необходимо, чтобы информация автоматически обрабатывалась, надо преобразовывать данные в структурированные форматы, которые можно задать с помощью инструкций (алгоритмов).

Рассматривая форматы структурированных данных, необходимо помнить, что нет единого формата файлов, который был бы удобен для машинной обработки всех типов данных, поэтому при его выборе необходимо исходить из содержания информации и задач, для которых будет использоваться данный документ. Кроме того, необходимо предостеречь формат от устаревания, так как это может ограничить доступ к архивным файлам. С другой стороны, необходимо, чтобы технологии были доступны всем желающим, а не только высокотехнологическим слоям общества.

2. Описание системы

Целью данной работы является разработка системы публикации открытых государственных данных ведомственной структуры расходов бюджета Санкт-Петербурга. Ведомственная структура расходов бюджета (ВСРБ) является одним из Приложений закона о бюджете Санкт-Петербурга [2], содержащим распределение бюджетных ассигнований, предусмотренных законом о бюджете на соответствующий финансовый год главным распорядителем бюджетных средств по разделам, подразделам, целевым статьям и видам расходов бюджетной классификации РФ. В ВСРБ Санкт-Петербурга содержится важная информация о распределении доходов города. Значительная часть доходов формируется из налоговых отчислений граждан, поэтому они должны иметь возможность изучения и анализа рассматриваемого документа. Публикация бюджета Санкт-Петербурга (как и других открытых государственных данных) создает механизм подотчетности, ко-

Труды XV Всероссийской объединенной конференции «Интернет и современное общество» (IMS-2012), Санкт-Петербург, Россия, 2012.

торый позволит гражданам контролировать расходы органов государственной власти города. В настоящее время Приложения к Закону о бюджете Санкт-Петербурга, содержащие данные о бюджете, публикуются в формате PDF. PDF не содержит структуры информации и не может быть использован для анализа и машинной обработки (например, для автоматизированной визуализации данных).

Необходимость публикации в структурированном формате информации о бюджете городов подтверждается открытием в октябре 2011 г. портала Открытого бюджета города Москвы [3], на котором можно не только получить всю интересующую информацию в простой и доступной форме (как текстовой, так и визуальной), но и скачать массивы данных в структурированном формате и предложить свои идеи по изменению структуры бюджета.

Система предназначена для публикации, отображения и визуализации массивов открытых государственных данных бюджета Санкт-Петербурга. Основной целью ее создания является обеспечение доступа к информации о бюджете Санкт-Петербурга в удобном для пользователей формате, а также предоставление данных разработчикам в машиночитаемых форматах.

Структура разрабатываемой системы представлена на Рис. 1 и заключается в следующем. Массив данных в формате PDF скачивается с официального источника. После этого исследуется их структура, на основании которой проектируется иерархия классов онтологии. Полученная иерархия классов создается в программе Protégé и заполняется экземплярами классов. Полученный проект программы Protégé экспортируется в формат RDFs в виде двух RDFs-файлов: файла иерархии классов и файла экземпляров. После этого разрабатываются клиентская и серверная части веб-системы публикации данных ведомственной структуры расходов бюджета Санкт-Петербурга. В разработанную систему с помощью веб-интерфейса загружаются полученные RDFs-файлы, которые сохраняются в базе данных системы. После этого пользователь системы получает возможность скачать загруженные массивы данных, отобразить их на экране или построить графики по заданным параметрам.

Рис. 1. Структура разрабатываемой системы

Система должна обеспечивать реализацию следующих задач: предоставление разработчикам машиночитаемых данных для использования их в сервисах и программах; предоставление заинтересованным гражданам массивов данных с возможностью их отображения по выбранным пользователем параметрам; добавление массивов открытых государственных данных.

3. Исходные данные

Согласно Бюджетному кодексу РФ бюджет является «формой образования и расходования денежных средств, предназначенных для финансового обеспечения задач и функций государства и местного самоуправления» [2]. Он является главным политическим и финансовым инструментом реализации социально-экономической политики. Благодаря структуре бюджетных расходов может происходить рост благосостояния граждан, обеспечение социальных сдвигов, повышение конкурентоспособности национальной экономики. Публикация бюджета (как РФ, так и субъектов РФ) позволяет обществу влиять на расходы и отслеживать причины перераспределения денег. При изучении бюджетного процесса специалисты выделяют понятие «прозрачности бюджета», под которым обычно понимается открытость, доступность и полнота информации о бюджете. Она зависит как от выбранной политики государства, так и от способности и готовности населения и институтов гражданского общества осуществлять контроль за расходами и отстаивать интересы городского сообщества.

Бюджет Санкт-Петербурга публикуется с 1990-х гг. Его объем в 2012 г. более 500 стр., что значительно затрудняет понимание и анализ документа. Основная задача разрабатываемой системы публикации ОГД ВСРБ Санкт-Петербурга заключается в упрощении этого процесса, так как структурированный формат позволит сделать анализ данных автоматизированным, снизив требования к квалификации заинтересованного гражданина.

Бюджет Санкт-Петербурга на 2012 г. включает в себя доходы, ведомственную структуру расходов, источники финансирования дефицита бюджета, программы государственных внутренних заимствований Санкт-Петербурга и ряд других приложений. Разработка системы будет основана на данных ведомственной структуры расходов бюджета Санкт-Петербурга на 2012 г. (Приложение 3 к Закону Санкт-Петербурга «О бюджете Санкт-Петербурга на 2012 год и на плановый период 2013 и 2014 годов»). Выбор данного раздела позволит начать изучение с наиболее востребованных гражданами данных, но, при необходимости, будет возможно по аналогии расширить функционал системы на остальные разделы.

Термин «ведомственная структура расходов бюджета» (ВСРБ) в Бюджетном кодексе РФ определен как распределение бюджетных ассигнований, предусмотренных законом (решением) о бюджете на соответствующий финансовый год главным распорядителем бюджетных средств, по разделам, подразделам, целевым статьям и видам расходов бюджетной классификации РФ [4].

ВСРБ Санкт-Петербурга представляет собой таблицу, состоящую из сумм и статей расходов и классификаторов, обеспечивающих международную сопоставимость данных. Коды содержат основную информацию, необходимую для анализа бюджета. Поэтому для того чтобы перевести данные в структурированный формат, необходимо сгенерировать справочники классификаторов, благодаря которым станет возможным автоматизированный анализ информации.

Согласно Указаниям «О порядке применения бюджетной классификации Российской Федерации», утвержденным Приказом Министерства финансов РФ от 28.12.2010 № 190н, «классификация расходов бюджетов представляет собой группировку расходов бюджетов всех уровней и отражает направление бюджетных средств на выполнение единицами сектора государственного управления и местного самоуправления основных функций, решение социально- экономических задач» [4].

4. Разработка онтологии расходов бюджета

Онтологии необходимы для определения словаря терминов, совместно используемых специалистами. Они состоят из машинно-интерпретируемых формулировок основных понятий и отношений между ними, определяя общий словарь для ученых, которым необходимо использовать информацию о предметной области. Онтологии обычно используются другими приложениями, например, онтология ведомственной структуры расходов бюджета Санкт-Петербурга может применяться для визуализации содержащихся в ней данных по выбранным пользователем параметрам. Основной задачей онтологий является совместное использование людьми или программными агентами общего понимания структуры информации.

Перед проектированием онтологии необходимо определить, какую область она будет охватывать, и ответы на какие типы вопросов в ней должны содержаться. В данном случае областью онтологии является представление структуры ВСРБ Санкт-Петербурга и сумм расходов. Используя приложения, основанные на данной онтологии, пользователь сможет получать информацию по интересующим его статьям расходов, сравнивать расходы различных распорядителей бюджетных средств или сравнивать суммы по операциям сектора государственного управления. Разработку онтологии можно разбить на три этапа: определение классов и их иерархии, определение слотов и описание допустимых значений, создание экземпляров классов и заполнение значений слотов экземпляров.

В онтологии ВРСБ Санкт-Петербурга необходимо выделить классы используемых в бюджете классификаторов (распорядители бюджетных средств, операции сектора государственного управления, разделы и др.) и класс расходов, содержащий значения сумм расходов с их классификаторами.

После определения классов создается внутренняя структура понятий, которая описывается с помощью свойств (слотов) классов. Свойства могут быть внутренними, внешними, частью целого или отношениями с другими индивидуальными концептами. Все подклассы класса наследуют свойства этого класса, поэтому свойства должны быть привязаны к самому общему классу, у которого может быть данное свойство.

Последним шагом в разработке онтологии является создание экземпляров классов. Для определения отдельного экземпляра необходимо создать экземпляр нужного класса и ввести значения слотов. Классы онтологии описывают структуру расходов бюджета Санкт-Петербурга, а созданные экземпляры наполняют ее конкретными значениями.

На основании результатов исследования структуры данных ВСРБ Санкт-Петербурга была разработана онтология, структура которой представлена на Рис. 2. В данной онтологии содержатся 6 классов с классификаторами, применяемыми в бюджете Санкт-Петербурга (классы «Главный распорядитель бюджетных средств», «Вид расходов», «Операции сектора государственного управления», «Подраздел», «Раздел», «Целевая статья») и класс «Расходы», содержащий все строки с суммами расходов.

После этого онтология была реализована с помощью программного обеспечения Protégé. Protégé является свободным, открытым редактором онтологий и фреймворком для построения баз знаний. ПО Protégé поддерживает два основных способа моделирования онтологий: Protégé-Frames и Protégé-OWL. Онтологии, разработанные с помощью данной системы, могут быть экспортированы в форматы RDF (RDF Schema), OWL и XML-схема. С помощью поддержки модулей возможно расширение функциональности Protégé. В работе используется редактор Protégé-OWL, позволяющий пользователям строить онтологии для семантической паутины, в частности на OWL. С помощью такой онтологии формальная семантика OWL определяет, как получать логические следствия (факторы, которые не присутствуют непосредственно в онтологии, но могут быть выведены из существующих посредством семантики). Данные выводы могут быть основаны на одном документе или на множестве распределенных документов, которые объединяются с использованием механизмов OWL.

Рассмотрим создание классов на примере класса «Вид расходов». Новому классу присваивается стандартное имя (основанное на названии проекта), для изменения которого необходимо изменить значение поля “Name”. В системе Protégé приняты пра-

вила наименования, согласно которым первая буква в каждом слове в имени класса пишется в верхнем регистре, а остальные буквы в нижнем.

Рис. 2. Структура онтологии ВСРБ Санкт-Петербурга

Согласно данным правилам, название для класса «Вид расходов» будет записано как ClassOfExpenditures. В поле Description (описание) сохраняется русское название класса. В ПО Protégé классы могут быть как конкретными (Concrete), на основе которых система может создавать экземпляры, так и абстрактными, т.е. классами, у которых экземпляров нет. По умолчанию класс создается «конкретного» типа, изменить который можно в поле Role. Рассматриваемый класс может иметь экземпляры, поэтому тип класса изменять не нужно. Для добавления нового слота необходимо открыть окно создания слота и заполнить все необходимые поля: наименование слота, тип значения, мощность слота и другие. У рассматриваемого класса есть два слота: слот codeClassOfExpenditures (код Вида расходов) и слот nameClassOfExpenditures (наименование Вида расходов); в данном случае оба слота единичной мощности и типа данных Строка. Аналогично описанному выше способу создаются все классы из структуры онтологии, представленной на Рис. 2.

Для дальнейшего использования полученной онтологии без ПО Protege возможно ее экспортирование в формат RDFs. Фрагмент кода полученного файла представлен в Листинге 1. Тег <rdfs:Class rdf:about="&rdf_;ClassOfExpenditures"> декларирует некий класс, который описывается в некотором пространстве имен “rdf” как ClassOfExpenditures. Указание различных пространств имен дает возможность повторно использовать другие онтологии, уточнять и расширять их, объявляя свой подклассы. Данная возможность позволяет осуществлять интеграцию узконаправленных информационных ресурсов, объединенных широкой предметной областью.

Листинг 1. Структура классов в RDFs-файле

```
<rdfs:Class
rdf:about="&rdf_;ClassOfExpenditures"
        rdfs:comment="Вид расходов"
        rdfs:label="ClassOfExpenditures">
       <rdfs:subClassOf
rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property
rdf:about="&rdf_;codeClassOfExpenditures"

rdfs:label="codeClassOfExpenditures">
       <rdfs:domain
rdf:resource="&rdf_;ClassOfExpenditures"/>
       <rdfs:range
rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

Проанализируем код класса “ClassOfExpenditures” (Вид расходов): тег <rdf:about> обеспечивает название (ссылку) на онтологию, которой в данном случае является ссылка «http://protege.stanford.edu/rdf/ClassOfExpenditures»; тег <rdfs:comment> обеспечивает возможность аннотировать онтологию, то есть добавить комментарий о том, что данный класс обозначает «Вид расходов»; тег <rdfs:label> обеспечивает удобное для чтения пользователем название класса, тег <rdfs:subClassOf> связывает рассматриваемый класс с его надклассом, которым в данном случае является класс Resource. У рассматриваемого класса есть два свойства: codeClassOfExpenditures и nameClassOfExpenditures, которые в синтаксисе RDFs-файлов прописываются с тегом <rdf:property>. Тег <rdfs:domain> показывает, что данное свойство относится к свойствам класса ClassOfExpenditures, а тег <rdfs:range> обозначает диапазон свойств, ограничивающих экземпляры класса (в данном случае они должны быть типа Literal).

После создания иерархии классов необходимо создать экземпляры, то есть непосредственно данные базы знаний. Перед этим необходимо еще раз проверить структуру классов, так как при в случае необходимости изменения ее в дальнейшем, возможна потеря уже введенной информации.

Например, источником данных для класса CEOIncome (Главный распорядитель бюджетных средств) является документ «Перечень и коды главных администраторов доходов бюджета Санкт-Петербурга, которые являются органами государственной власти Санкт-Петербурга, и закрепляемые за ними виды доходов бюджета Санкт-Петербурга» Значениями поля codeCEOIncome являются значения столбца «Код главного администратора бюджетной классификации РФ», значениями поля nameCEOIncome – значения столбца «Наименование».

Аналогично происходит создание экземпляров всех классов, кроме экземпляров класса Costs, происходит с помощью интерфейса программы Protege. После этого полученный проект экспортируется в формат RDFs, получая на выходе два RDFs-файла: файл с иерархией классов и файл с экземплярами классов. Его структура аналогична структуре файла с иерархией классов, за исключением того, что вместо тегов <rdfs:Class> или <rdf:Proprty> указывается тег <rdf:ClassName>, где ClassName – имя класса, экземпляр которого создается в данном теге.

Схема автоматизированного создания экземпляров класса Costs представлена на Рис. 3. Из первоначального файла ведомственной структуры сводной бюджетной росписи бюджета Санкт-Петербурга на 2012 год (формат PDF) данные копируются в файл формата TXT. После этого с помощью регулярных выражений удаляются все текстовые данные и нумерация строк из столбца «Номер» ведомственной структуры сводной бюджетной росписи.

Рис. 3. Схема автоматизированного создания экземпляров класса Costs

В результате данных действий получен файл, в котором содержатся данные для свойств экземпляра класса Costs, разделенные символом пробела. Данный файл является структурированным, поэтому для его преобразования написан парсер на языке программирования Java.

Принцип работы парсера следующий: на входе парсер получает два файла: RDFs-файл с созданными экземплярами класса (source.txt) и полученный txt-файл с данными расходов бюджета Санкт-Петербурга (data.txt). В парсере открывается два потока для чтения данных из файлов и поток для записи результатов работы программы в файл (result.txt). После считывания всех данных в массив парсер построчно сохраняет значение кодов из файла data.txt в переменные, одноименные будущим свойствам экземпляров класса Costs и записывает их в выходной файл с соблюдением синтаксиса RDFs-файла. Полученный в результате работы программы файл с экземплярами класса Costs копируется в файл с экземплярами классов онтологии. Таким образом, в результате получается онтология расходов бюджета Санкт-Петербурга на 2012 г., состоящая из двух RDFs-файлов, доступная для дальнейшей машинной обработки.

5. Разработка архитектуры системы

Для реализации системы была выбрана платформа Java 2 Enterprise Edition, позволяющие разрабатывать сложные серверные приложения, обеспечивающие высокую стабильность в работе и обеспечивающие целостность данных во время работы системы. Приложение разрабатывалось на основе облачного сервиса Google App Engine. Эта платформа предоставляет хостинг для Java-приложений в масштабируемой инфраструктуре Google. Платформа предоставляет виртуальную машину Java 6, интерфейс сервлетов Java, а также поддержку стандартных интерфейсов для масштабируемого хранилища данных и служб App Engine, таких как JDO, JPA, JavaMail и JCache. Стандартные средства поддержки упрощают разработку приложений, делают среду привычной и позволяют перемещать приложения непосредственно между средой разработки и собственной средой сервлетов.

Разрабатываемая система представляет собой web-приложение, которое имеет двухуровневую клиент-серверную структуру. В качестве клиентской части приложения выступает Интернет-браузер. Главными задачами клиентской части приложения, ориентированного на представление данных, являются: обеспечение интерфейса пользователя и накопление данных, передаваемых на клиент. Серверные компоненты выполняют основную бизнес-логику приложения.

Типовой концепцией приложения является архитектурный шаблон MVC (модель – представление – контроллер), схема которого изображена на Рис. 4. Шаблон проектирования MVC предполагает разделение приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо. Представление отвечает за отображение информации, поступающей из

системы или в систему. Им может быть шаблонизатор, целью которого является только в выводе информации в виде HTML на основе каких-либо готовых данных. Контроллер является модулем управления вводом и выводом данных. Данный модуль должен следить за переданными в систему данными (через форму, стоку запроса, cookie или любым другим способом) и на основе введенных данных решить: передавать ли их в модель или вывести сообщение об ошибке и запросить повторный ввод. Кроме того, контроллер определяет тип данных, полученных от модели и передавать информацию в модуль представления. Модель – модуль, отвечающий за непосредственный расчет чего-либо на основе полученных от пользователя данных. Результат, полученный этим модулем, должен быть передан в контроллер, и не должен содержать ничего,

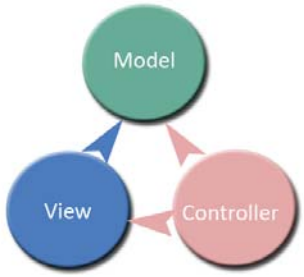

Рис. 4. Архитектура «MVC»

относящегося к непосредственному выводу (то есть должен быть представлен во внутреннем формате приложения).

Для реализации шаблона проектирования MVC выбран простой фреймворк Struts. Его выбор обусловлен достаточной стабильностью в работе, гибкостью и простотой реализации. Struts предоставляет стандартный контроллер и различные средства для создания страниц представления. Разработчик веб-приложения отвечает за написание кода модели и создание конфигурационного файла, связывающего воедино модель, представление и контроллер. Запросы от клиента передаются в виде Actions (действий), определенных в конфигурационном файле. При получении контроллером такого запроса, он передает его соответствующему классу. Последний взаимодействует с кодом Модели и возвращает контроллеру ActionForward строку, определяющую страницу для отправления клиенту. Диаграмма соответствующих классов представлена на Рис. 5.

В данном проекте модели (models) реализуют доступ к базе данных при помощи ORM (Object-Relation Mapping). ORM – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

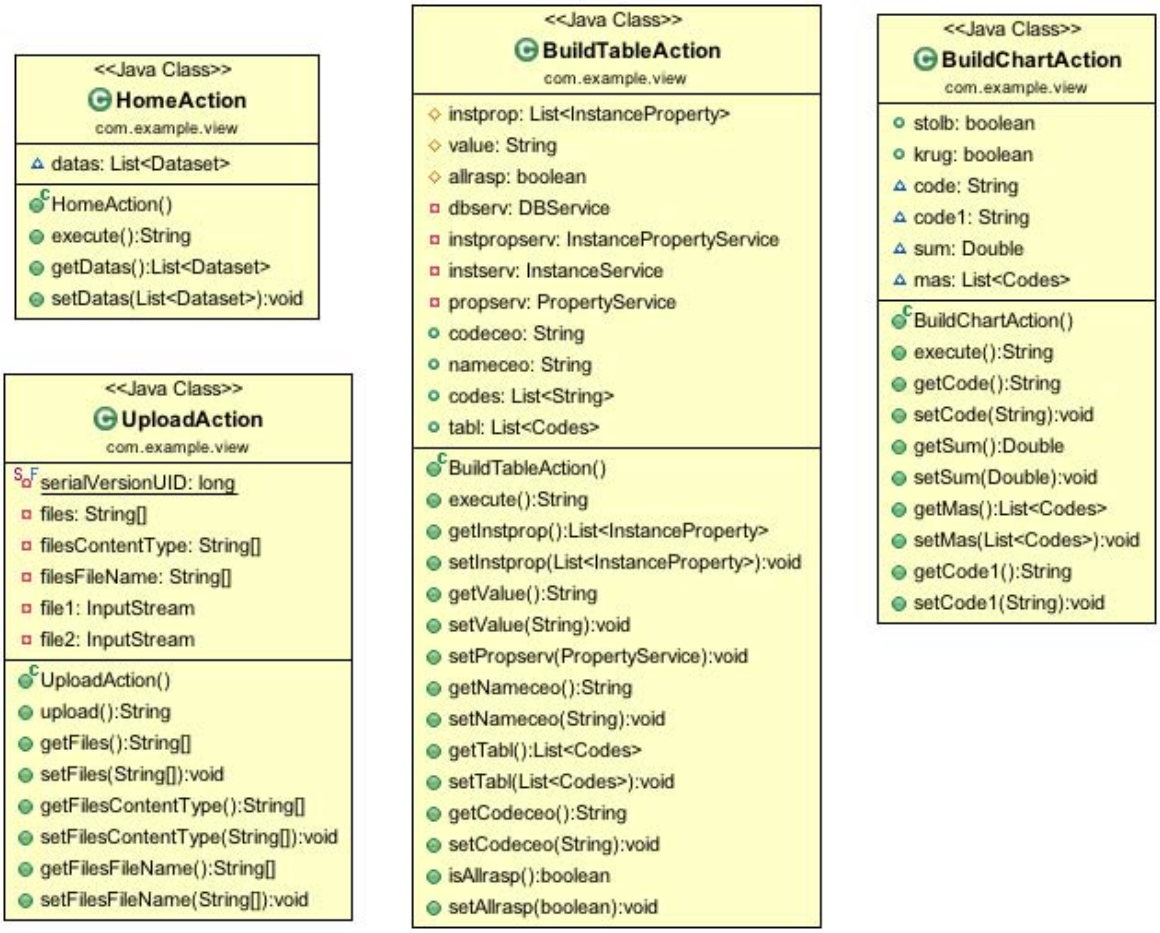

Рис. 5. Диаграмма классов Action

Она избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым повышая скорость разработки. Модели описываются как классы с аннотациями (JPA), которые представляют собой сущности и находятся в пакете com.example.model. Классы могут содержать только публичные аксессоры и аннотированные mapped поля. Для каждой «таблицы» в базе данных были созданы модели (классы), которые выполняют роль «медиатора». Для сохранения данных в базу необходимо создать экземпляр модели и заполнить его поля значениями. Этот объект передается в базу данных с помощью JDO и сохраняется в виде записи в таблице. Когда необходимо получить объект — JDO также возвращает готовый экземпляр класс модели, который можно использовать как java -объект без дополнительных преобразований.

В разрабатываемом приложении использовалась многоуровневая архитектура, которая облегчает поддержку кода и делает возможным масштабирование приложения. Архитектура имеет 3 уровня:

1. уровень представления (Actions)
2. уровень бизнес-логики (Service)
3. уровень базы данных (AppEngine)

Архитектура изображена на рис. 6.

Пользователь (client-side) отправляет запрос к серверу (1). Например, пользователь запросил данные по определенному департаменту. Все запросы обрабатываются фреймворком Struts 2. Сначала запрос проходит через Interceptor (перехватчик запросов), в котором определяется какой из классов типа ActionSupport (или просто Action) должен обработать этот запрос. В Action происходит проверка данных, а также извлечение всех необходимых данных из запроса и вызывается необходимый Service (1.1). На этом шаге управление и параметры передаются следующему уровню — бизнес-логики. Этот уровень отвечает за обработку данных, в данном случае может происходить проверка — имеет ли текущий пользователь права на получение этих данных, а также формируется запрос к базе данных на основе известных типов данных, идентификаторов (key id).

Рис. 6. Трехуровневая архитектура приложения

На следующем уровне 1.1.1 (AppEngine, уровень базы данных), происходит непосредственно обращение к облачному хранилищу BigTable и выполняется запрос. Этот уровень является для нашего приложения «черным ящиком», т.е. наша программа никак не контролирует то, что там происходит и по сути ничего об этом знать не должна. Этот уровень после выполнения запроса, возвращает на уровень бизнес-логики коллекцию (набор) объектов. Уровень бизнес-логики формирует необходимый ответ для уровня представления, а уровень представления в свою очередь передает эти данные в JSP-страницу, где и формируется ответ в виде HTML-содержимого, которое передается браузеру пользователя для отображения. Таким образом каждый уровень решает только свои задачи:

1. уровень представления: извлечение данных из запроса, проверка данных.
2. уровень бизнес-логики: обработка данных.
3. уровень базы данных: сохранение данных для постоянного хранения и получение данных.

Эта архитектура накладывает определенные ограничения: например, нельзя обращаться непосредственно к базе данных на уровне представления в Action-классе. Это делает работу с базой данных более безопасной, так как предотвращает попадание в нее некорректных непроверенных данных. А извлечение данных из запроса и их проверка делается только на уровне представления — это позволяет отделить логику приложения и упростить чтение и понимание кода. Это очень удобно, так как разработчик всегда знает, где решается какая задача, и сервисы могут быть использованы многократно в других Action-классах.

Сущностями, отображающими полученную моделями информацию, являются jsp- страницы, которые содержат информацию двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, XML и JSP элементы, которые конструируют динамическое содержимое. JSP является одной из высокопроизводительных технологий, так как весь код страницы в результате транслируется в java-код.

Кроме вышеперечисленных сущностей и моделей были разработаны наборы собственных утилит, которые могут использоваться на любом уровне архитектуры и помогают решать сторонние задачи, не связанные непосредственно с бизнес-логикой приложения.

Заключение

В рамках данной работы реализовано создание структурированного массива данных ведомственной структуры расходов бюджета Санкт-Петербурга и разработан прототип системы публикации, выполняющей следующие функции:

Загрузка RDF-файлов с массивом данных. После выбора кнопки «Загрузить данные» пользователь попадает на страницу с загрузкой данных, на которой он должен заполнить мета- информацию о загружаемом массиве: название массива данных; название и тип ресурса, с которого его можно скачать; формат данных; краткое описание данных; категория данных; дата внесения последних изменений; версия файла; источник данных и ответственное лицо за них; лектронный адрес ответственного лица; нормативные документы, связанные с данным массивом данных; теги. Затем пользователь загружает два RDF-файла: файл с структурой онтологии и файл с экземплярами. После нажатия кнопки «Сохранить» система загружает файлы, сохраняет данные из них в базу данных системы и сохраняет мета-информацию о файлах.

Просмотр массива данных. Функция позволяет просмотреть информацию о массиве данных и скачать связанные с ним файлы.

Выбор данных для построения таблицы и графиков. Пользователь имеет возможность выбрать диапазон данных, по которым необходимо построить таблицу или график. В случае с массивом данных ведомственной структуры расходов бюджета Санкт-Петербурга пользователь может выбрать интересующие его коды.

Построение таблиц и графиков. После выбора данных система строит графики (столбчатую и круговую диаграммы) и отображает таблицу с суммами расходов.

Литература

[1] Доклад «Об обеспечении доступа населения к информации о деятельности государственных органов и органов местного самоуправления» [Электронная версия] // URL: http://ivan.begtin.name/wp-content/uploads/2011/09/doklad.pdf (дата обращения: 31.05.2012).

[2] Закон Санкт-Петербурга от 26.10.2011 No 658-120 «О бюджете Санкт-Петербурга на 2012 год и на плановый период 2013 и 2014 годов» [Электронная версия] // URL: http://www.fincom.spb.ru/comfin/budjet/laws/doc.htm?id=506@cf_npa_bud (дата обращения: 31.05.2012).

[3] Портал Открытого бюджета г. Москвы [Электронный ресурс] // URL: http://budget.mos.ru/ (дата обращения: 31.05.2012).

[4] Указания «О порядке применения бюджетной классификации Российской Федерации» (утверждены Приказом Министерства финансов РФ от 28.12.2010 No 190н) [Электронная версия] // URL: http://www1.minfin.ru/ru/budget/classandaccounting/classification/ (дата обращения: 31.05.2012).

Development of Saint-Petersburg's budget costs publication system

O.V. Parkhimovich

The article describes development stages of open government data of Saint-Petersburg's budget costs departmental structure; the aim of such system is loading, display and visualisation of data array.