A.V. Bubnov, P.P. Keyno

Features of Data Receiving Over HTTP Protocol in the Compilable Web-application and Their Unification in Own DSL

Keywords: HTTP, DSL, BlockSet, BML.

The article describes methods handling incoming requests data. The main idea of this approach resides in determining what enities Web-service should work with. After that we define attributes, which can be accesed during data exchange procedure considering request division into idempotent and modifying.

А.В. Бубнов, П.П. Кейно

ОСОБЕННОСТИ ПРИЁМА ДАННЫХ ПО ПРОТОКОЛУ HTTP В КОМПИЛИРУЕМОМ WEB-ПРИЛОЖЕНИИ И ИХ УНИФИКАЦИЯ В ПРЕДМЕТНО-ОРИЕНТИРОВАННОМ ЯЗЫКЕ

Всю деятельность пользователя в среде Web можно разделить на две категории: получение информации и её создание (рис. 1).

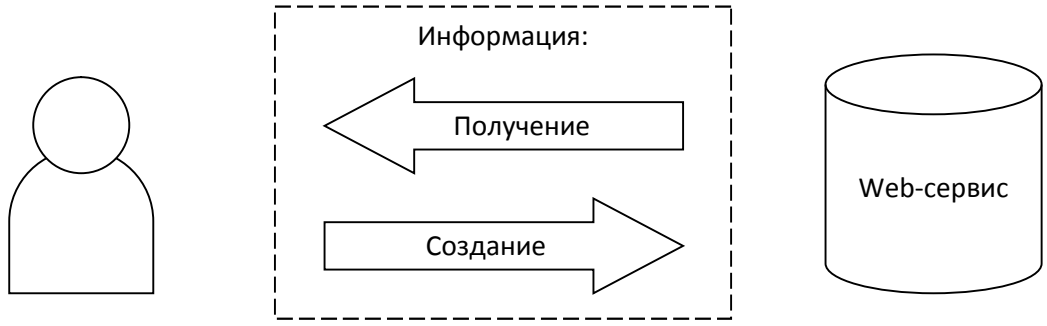

Рис. 1. Упрощённая схема взаимодействия с пользователем и Web-сервисом

Другими словами, действия конечного пользователя сводятся либо к простому просмотру нужной информации, не влияя на работу сервиса (например, вводя один и тот же запрос в поисковый сервис, мы получаем те же результаты, пока поисковые базы не будут обновлены новыми данными). Это делает возможным кеширование результатов запроса. Или же пользователь сам наполняет сервис содержимым (блоги, форумы, вики). Таким образом, запросы к Web-сервису можно также разделить на две категории: идемпотентные, которые не влияют на работу сервиса, и модифицирующие, которые позволяют участвовать в работе генерируемых данных, то есть, по сути, управлять контентом Web-сервиса.

При работе с Web-сервисами для обмена данными используется HTTP-протокол, а именно: HTTP-методы [1]. Наиболее часто используемыми методами передачи данных являются методы GET и POST. Несмотря на то, что технология позволяет передавать параметры запроса любым методом, согласно стандарту HTTP [2] идемпотентным является метод GET, в то время как создание, изменение и удаление информации должны осуществляться другими методами – например, POST.

Но решение этой задачи ложится на плечи разработчика Web-сервиса. Причем основной сложностью при обработке запросов является отнюдь не семантически вер-

ное использование методов. Так как они позволяют передать серверу некоторую пищу для размышления, то таким образом можно вмешаться в работу сервиса. Неверное использование запросов может позволить злоумышленнику получить доступ к закрытым данным сервиса или изменить то, что изменять напрямую нельзя. Простейший пример: если содержимое запроса напрямую передавать в СУБД, то это открывает путь к SQL-инъекциям, что, в свою очередь, может позволить злоумышленнику получить доступ ко всей базе или прописать себя в качестве администратора.

Есть несколько способов защиты от этого:

– экранирование спецсимволов;

– проверка входных значений.

Если с первым способом всё понятно, то второй осложняется тем, что проверки надо выполнять каждый раз, когда запускается процесс приема данных. Таким образом, можно упустить некоторые параметры и тем самым оставить уязвимую область в работе сервиса.

В рамках разработки проекта BlockSet было необходимо решить задачу обмена данных, учитывая особенность логики построения динамического содержимого. Методология BlockSet подразумевает разделение логики работы динамической Web-страницы (локации) на наборы и блоки. Набор представляет собой отдельную сущность, в то время как блоки отражают ее отдельные аспекты. Блоки в свою очередь содержат информацию о типе хранящейся в них информации, а также различных ограничениях, накладываемых на нее.

Все данные о структуре наборов и блоков задаются при помощи декларативного языка BML и хранятся в специальных документах, построенных на основе XML. Вся эта информация затем обрабатывается специальной программой-интерпретатором, которая на основе полученной структуры модели данных, содержимого БД и параметров запроса генерирует итоговую HTML-страницу, отправляемую в браузер клиента в качестве ответа. Важно отметить тот факт, что модель данных загружается при старте работы интерпретатора, таким образом, в процессе обработки входящих запросов интерпретатор «осведомлен» обо всех возможностях загружаемой локации. Локация – часть BML документа, содержащая наборы с блоками и идентифицируемая определенным URL-адресом. В модели допустимо задать атрибуты по умолчанию. К таким атрибутам могут относиться и те, которые указывают, каким блокам доступны лишь идемпотентные запросы, а какие могут принимать участие в модифицирующих.

Для обеспечения обмена данными блокам добавлены следующие атрибуты:

– onread – для передачи идемпотентных данных;

– oncreate, onupdate, ondelete – для передачи данных модифицирующих запросов создания, обновления и удаления контента соответственно.

Эти атрибуты могут принимать одно из двух значений, которое определяет, откуда блок будет принимать данные:

– input – данные, которые должны прийти от клиента;

– default – данные берутся из атрибута “default”.

Если из атрибута “default” невозможно получить данные, то данные этого блока будут проигнорированы.

Для обработки модифицирующих данных вводится служебный блок “act”:

<block name=“act” default=“create” onread=“input”/>

Он определяет, каким образом должен обрабатываться набор. Так же, как и для других блоков, можно жестко задать значение по умолчанию через атрибут “default”. Для указанного блока это метод обработки. В противном случае допустимо брать значение из запроса, указав значение атрибута “onread”, равное “input”. Это позволяет со-

здавать гибкие локации, которые могут выполнять обработку схожих данных разными способами. Например, можно реализовать редактирование или создание новых статей при помощи одной локации.

Прием данных в проекте BlockSet реализован следующим образом: данные модифицирующих запросов могут быть приняты только методом POST, в то время как идемпотентные данные клиент может передавать как методом POST, так и методом GET. В случае если одинаковые данные передаются обоими методами, конфликт разрешается с помощью системы приоритетов (рис. 2): если модифицирующие данные были обнаружены в запросе, то искать идемпотентные данные мы будем уже среди GET-данных. Если модифицирующих данных не было, то сначала будет произведен поиск информации идемпотентного запроса среди POST данных, и только в случае, если там ничего не было обнаружено, будет произведен поиск среди GET-данных.

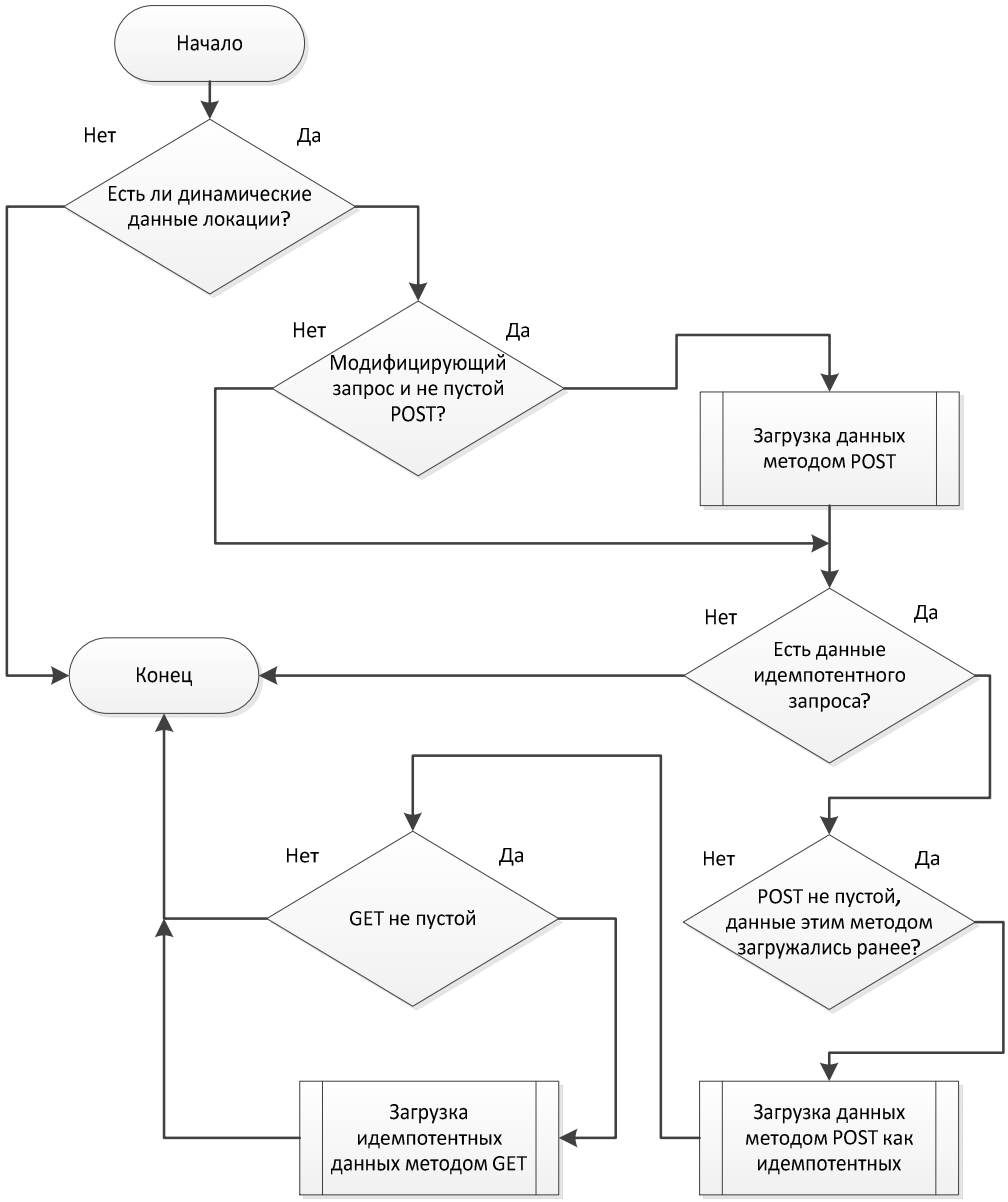

Рис. 2. Схема обработки данных различными методами

Таким образом, решается главная проблема: при помощи блоков осуществляется автоматическая проверка получаемых значений. А заранее построенная модель данных позволяет принимать значения только для определенных блоков, что исключает возможность инъекции в другие сущности и их атрибуты. Данный подход способствует масштабируемости проекта в целом, так как при создании новых локаций не нужно будет каждый раз заново определять правила обработки запросов, достаточно лишь указать, каким блокам информацию от каких запросов принимать.

СПИСОК ЛИТЕРАТУРЫ:

1. Mulloy B. Web API Design: Crafting Interfaces that Developers Love. 2012. 9 p.
2. Method Definitions URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.
3. O'reilly T. What is Web 2.0: Design patterns and business models for the next generation of software //Communications &strategies. 2007. №. 1. P. 17.
4. Hebeler J. et al. Semantic web programming. – John Wiley & Sons, 2011. 67 p.

REFERENCES:

1. Mulloy B. Web API Design: Crafting Interfaces that Developers Love. 2012 9 p.
2. Method Definitions URL: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.
3. O'reilly T. What is Web 2.0: Design patterns and business models for the next generation of software //Communications &strategies. 2007. №. 1. P. 17.
4. Hebeler J. et al. Semantic web programming. – John Wiley & Sons, 2011. 67 p.