УДК 004.42

РЕАЛИЗАЦИЯ СИСТЕМЫ ЗАЩИТЫ WEB-ПРИЛОЖЕНИЯ С ПОМОЩЬЮ АСПЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

С.А. Зайкова, О.Ю. Рыжко

Разработана система защиты web-приложения с использованием аспектно-ориентированного программирования. Изучены целевые web-приложения с примером реализации аспектов безопасности, с помощью языка AspectJ в реализации Spring AOP Использован annotation-driven подход и спроектирована конфигурация web-приложения с применением стека технологий, отвечающих современным требованиям для построения безопасных информационных ресурсов.

Введение

Одним из основных требований, предъявляемых к современным информационным ресурсам и сервисам, является их широкая доступность. Наиболее активное развитие получили web-приложение, в которых роль клиента играет браузер, а роль сервера – web-сервер. В связи с этим встает вопрос об организации разграничения доступа и защите информации. Современные web-приложения имеют многослойную структуру и сложность кода, порой, возрастающего в геометрической прогрессии. Отдельные модули приложений разрабатываются разными командами, которые могут знать о другом модуле только его интерфейс.

Актуальность исследования вытекает из необходимости выработки методов выделения функционала обеспечения безопасности web-приложения в отдельный независимый модуль, с возможностью повторного использования кода в совместимых проектах. Это позволило бы изменять политики безопасности web-приложения централизованно и без изменения бизнес-логики проекта. Проблематика заключается в тесной интеграции инструментов аспектно-ориентированного программирования с кодом целевого web-приложения, сложности внедрения аспектно-ориентированного программирования в проект, а также большом влиянии аспектов на производительность web-приложения, что, в свою очередь, зависит от типа внедрения аспектов и архитектуры приложения.

Аспектно-ориентированное программирование (АОП) в контексте реализации аспектов безопасности web-приложений может успешно применяться для разработки «сквозного» функционала модулей. Такой функционал достаточно сложно, а иногда невозможно, реализовать в рамках классической концепции объектно-ориентированного программирования (ООП). Типичные классы проблем, решение которых требует «сквозной» функциональности, относятся к следующим направлениям. Безопасность – аутентификация пользователей и программ; авторизация; криптографические операции над данными с целью обеспечения их конфиденциальности. Надежность – проверка выполнения пред- и постусловий в модулях и

инвариантов в классах, обработка ошибок. Безопасность многопоточного выполнения кода – синхронизация по ресурсам или по событиям, выделение критических участков кода, взаимное исключение доступа к ним. Протоколирование и профилирование работы программы – трассировка начала и окончания выполнения каждой функции, вывод их аргументов и результатов, сбор и вывод статистической информации об исполнении различных фрагментов программы.

Можно отметить, что не существует устоявшихся методик выделения, разработки и реализации аспектов таких приложений. Несмотря на то, что сегодня это скорее академические исследования, множество компаний (BEA, IBM, Microsoft), а так же open source объединения (JBoss, ObjectWeb, Eclipse) работают в этом направлении с видимыми результатами. Основателем АОП является Г. Кичалес (G. Kiczales), Университет Британской Колумбии, Канада, а наиболее популярным инструментом АОП для платформы Java – разработанная под его руководством система AspectJ, первая версия которой была создана в 1995 г. (Xerox Palo Alto Research Corporation, США) [1].

В основе аспектно-ориентированного программирования лежит понятие crosscutting concerns, которое не имеет устоявшегося эквивалента в русском языке. Наиболее близким по смыслу считается и чаще всего используется словосочетание «сквозная функциональность» Под сквозной функциональностью понимается функциональность, реализовать которую в отдельном компоненте языка программирования традиционными средствами процедурного или объектно-ориентированного программирования или очень сложно, или невозможно, поскольку эта функциональность необходима в большей части модулей системы. Кроме того, эта функциональность не относится напрямую к предметной области. Примером такой функциональности является протоколирование работы системы (logging).

Использование аспектно-ориентированного программирования помогает следовать принципу разделения ответственности (separation of concerns), что положительно сказывается на многих характеристиках разрабатываемого web-приложения, некоторыми из которых являются: более качественная реализация, улучшенные возможности тестирования и модульность системы.

Целью работы является разработка тестового web-приложения, с использованием полного стека технологий, с внедрением аспектов и выделением сквозного функционала обеспечения безопасности.

Разработка защиты web-приложения

При выборе инструментов разработки тестового web-приложения учитывались современные требования, предъявляемые с учетом корпоративного уровня, а также существующие практики в разработке подобных приложений. web-приложение разработано средствами языка программирования Java EE в интегрированной среде разработки Eclipse. Архитектура приложения является многослойной и построена в соответствии с классическим шаблоном проектирования MVC на базе открытого фреймворка

Spring MVC. Для разметки и компоновки JSP-страниц использовался шаблонизатор Apache Tiles и Metro UI CSS. Доступ к слою данных осуществляется с использованием фреймворка Hibernate. web-приложение работает в контейнере сервлетов Apache Tomcat.

При рассмотрении различных подходов к применению принципов АОП, сразу стоит упомянуть, что существует два типа внедрения аспектов: статически или динамически. Инструменты АОП со статическим внедрением осуществляют преобразования целевой программы либо на уровне исходного кода, либо на уровне промежуточного кода. Например, байт-кода Java, универсального промежуточного кода CIL платформы .NET или какого-либо собственного внутреннего представления, используемого только данным инструментом АОП, или на уровне платформенно-зависимого объектного кода (native code). Используется также подход, основанный на внедрении аспекта на этапе just-in-time компиляции, т.е. в процессе динамической компиляции метода из промежуточного в платформенно-зависимый объектный код при первом его вызове. В каждой из этих моделей внедрения, в том или ином смысле, выполняется вставка кода аспекта в целевую программу. Преимущество – более высокая эффективность результирующего кода, недостаток – увеличение его объема, что не всегда приемлемо для мобильных устройств с ограниченными ресурсами.

При динамическом внедрении инструмент АОП ведет себя подобно отладчику, в котором заданы контрольные точки. Для активизации аспекта, перед исполнением каждого оператора целевой программы, инструмент АОП проверяет выполнимость каждого из условий, заданных в аспекте, и при истинности какого-либо из условий выполняет соответствующее ему действие аспекта. Подобный подход удобен, прежде всего, для отладки программы, но неприемлем при ее эксплуатации, с точки зрения эффективности [2].

Определение аспекта может быть выполнено на специальном метаязыке либо в терминах расширений базового языка программирования конструкциями АОП. При использовании метаязыка возникает проблема его отображения в базовый язык реализации, которая в некоторых инструментах решается путем конвертирования конструкций метаязыка АОП в определения специализированных атрибутов (custom attributes) в случае C# или аннотаций (annotations) в случае Java. При втором подходе возникает зависимость инструмента АОП от базового языка и всех его изменений. Код на таком расширенном языке требует для компиляции специализированный компилятор, и при любом изменении базового языка необходимо вносить соответствующие изменения в компилятор.

Первым и основным шагом в реализации защиты web-приложения является введения протокола AAA. Таким образом вводится ограничение на доступ к отдельным разделам web-приложения и ведется журналирование (аудит) действий пользователей для анализа их поведения, обнаружения неавторизированных действий. Аутентификация пользователя реализована в отдельном классе аспекта, который представляет собой оболочку над слоем контроллеров. При каждом вызове функций пакета web-аспект проверяет

наличие валидной учетной записи пользователя и соответствующих атрибутов сессии. При отсутствии информации об активном пользователе, аспект перенаправляет на страницу авторизации и обрабатывает введенные данные. Таким образом, аспект аутентификации переопределяет поведение контроллера и изменяет возвращаемые значения в зависимости от того аутентифицирован пользователь или нет. После успешной проверки данных учетной записи пользователя, аспект аутентификации передает управление аспекту авторизации.

Авторизация пользователя происходит после проверки данных его учетной записи. Аспект извлекает информацию из сущности пользователя и определяет доступен ли ему вызов определенных функций пакета web. В случае отсутствия необходимых привилегий, аспект авторизации изменяет модель и представление и перенаправляет на страницу с ошибкой.

Аудит действий пользователя обычно реализуется с помощью изменения методов, для которых необходимо логирование действий пользователя и стабильности работы метода. В случае с аспектно-ориентированным программированием, аспект аудита является надстройкой над бизнес-логикой. При обращении пользователя к определенным функциям web-приложение, аспект считывает информацию о пользователе, сигнатуры методов и результаты их работы. Результаты записываются в файл журнала в строго определенном виде, который содержит дату, тип события и основную информацию о событии. Формат файла журнала позволяет провести парсинг и извлечь записи журнала в другом приложении, которое будет играть роль панели администрирования.

Все разработанные аспекты находятся в одном пакете и могут быть вынесены в отдельную библиотеку классов. Связь аспектов и исходного кода приложения осуществляется с помощью конфигурационного файла фреймворка Spring, это является особенностью реализации Spring AOP.

Рассмотрим пример. Web-приложение представляет собой набор сервлетов, написанных на языке Java и работающих под управлением контейнера сервлетов Apache Tomcat. Приложение является упрощенной версией новостного сайта, где можно просматривать/добавлять новости, а также оставлять комментарии. Существуют такие сущности, как: пользователь, роль, новость, комментарий (рис. 1). Модель данных web-приложения соответствует разработанным таблицам базы данных (БД) под управлением СУБД MySQL. В модели выстроены связи один-ко-многим и наложены ограничения на каскадные операции для обеспечения целостности данных.

Web-приложение разработано на языке Java в интегрированной среде разработки Eclipse IDE. Создана структура пакетов и папки для конфигурационных файлов. Тип проекта – Dynamic Web Project. Для облегчения управления зависимостями в проекте выбран продукт Apache Maven. Все необходимые библиотеки указывались в конфигурационном файле pom.xml и далее загружались из репозитория maven. Можно отметить, что данный продукт будет полезен для работы с пользовательскими репозиториями и пакетами.

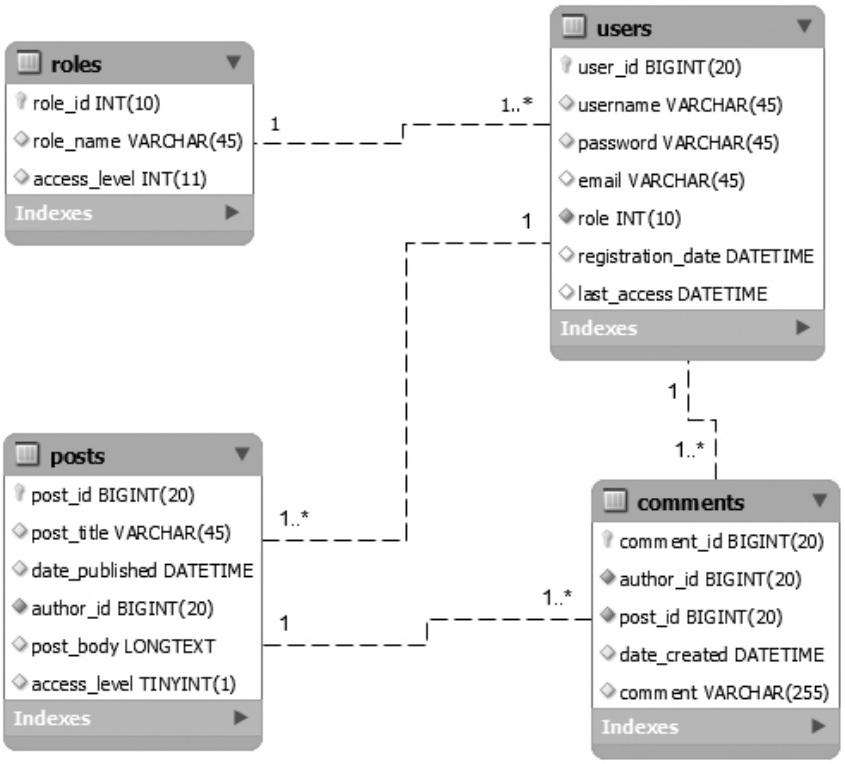

Рис.1. Модель данных Web-приложения

Пакеты domain.model и domain.service содержат классы объектов (сущностей) приложения, а также интерфейсы сервисов, которые определяют функционал. Пакеты repository.* содержат классы сущностей для БД, интерфейсы репозиториев и их реализации. Пакет service содержит классы сервисов, являющиеся реализациями интерфейсов из доменной (предметной) области. Пакеты web.* содержат контроллеры приложения, отвечающие на запросы пользователей, и вспомогательные классы. В контроллерах содержится информация о url-mapping-е, а также о возвращаемых представлениях. Пакет aop содержит разработанные аспекты.

В приложении реализована mvc-архитектура, для разработки выбран Spring MVC фреймворк с использованием стиля annotation-driven. Таким образом, компоненты определяются с помощью аннотаций непосредственно в теле класса, что позволяет значительно упростить развертывание spring. Фреймворк, исходя из этих аннотаций, самостоятельно определяет контроллер, сервис и т.д. Конфигурационный файл Spring, также как и Hibernate ORM и реквизиты подключения к БД были помещены в папку web-inf, которая является защищенной от доступа извне среды выполнения.

После обработки запроса пользователя контроллер возвращает какое-либо представление пользователю (представление и модель). Представление является jsp-страницей, с данными из модели. Для уменьшения объема кода страниц, а также для их упрощенной разработки и поддержания целостности стиля используется jsp-шаблонизатор Apache Tiles 3.0. Он позволяет разбить

страницу на отдельные блоки, которые потом можно использовать повторно. Таким образом, можно сделать одинаковое заглавие для всех страниц. Для того чтобы страницы приложения были более привлекательными, использовался css-framework Metro UI CSS. Основой шаблонизатора является конфигурационный xml файл, где указаны основные компоненты каждого представления, а также файл разметки с указанием общего формата страницы.

Рассмотрим форму входа, с которой работает аспект, фиксирующий все действия пользователя. Обработкой входа в приложение занимается контроллер LoginController. Метод login() не принимает параметров и обрабатывает GET-запросы по адресу *~/login*. Он возвращает только страницу пользователю. В возвращаемом параметре будет указано имя представления, это особенность функционирования шаблонизатора Tiles. Метод loginPost() принимает обязательные параметры: логин и пароль, которые проверяет и возвращает представление с моделью пользователя, либо сообщение об ошибке в противном случае. При успешной авторизации, открывается страница с данными пользователя. Метод обрабатывает POST-запросы по адресу *~/login*. Перед тем, как пользовательские данные будут отправлены, приложение вносит запись о том, что предпринята попытка входа, с указанием даты и времени. После того, как данные будут проверены, выводится сообщение о результате попытки входа с указанием имени пользователя. Если попытка была неудачной, в сообщении будет указано введенное имя пользователя. Параметр с паролем, в целях безопасности, не выводится для неудачной попытки.

Перехват управления у контроллера посредством аспекта из пакета aop происходит следующим образом. Аспект представляет собой класс, помеченный аннотацией *@Aspect* При разработке и использовании annotation-driven стиля советами называют методы данного класса. Рассмотрим листинг (рис.2):

```
@Aspect
public class LoggingAspect {

	@Around("execution(* loginPost(..))")
	public Object logAround(ProceedingJoinPoint jp) throws Throwable {

		System.out.println("Попытка входа в систему в " + new Date());
		ModelAndView value = (ModelAndView)jp.proceed();
		User user = (User) value.getModel().get("user");
		if(user != null){
		System.out.println("Выполнен вход пользователем " +
user.getUsername() + " в " + new Date());}
		else{System.out.println("Неудачная попытка входа с именем " +
jp.getArgs()[0].toString() + " в " + new Date());}
		return value;
	}

	}
```

Рис.2. Перехват управления у контроллера

Совет (метод) logAround() помечен аннотацией *@Around*, что означает возможность перехвата управления у контроллера до выполнения его метода (окружить метод). В параметрах аннотации указан срез PointCut, который указывает область применимости данного совета. В нашем случае указано, что совет актуален при выполнении метода loginPost(). Метод выводит сообщение до выполнения метода контроллера, и затем получает модель пользователя. Если она не пуста, выводит сообщение об удачном входе, иначе о неудачном. Таким образом, реализован микро-модуль журналирования без внедрения кода в основной рабочий метод для обеспечения безопасности функционирования ресурса.

Заключение

Как показало исследование, использование annotation-driven подхода при разработке, а также конфигурация web-приложения через xml файлы позволяет значительно сократить объем кода приложения, но может вызвать дополнительные временные затраты на отладку и проверку совместимости зависимых библиотек. Разработанная система безопасности является проектно-независимой и может быть применена в другом web-приложении, при условии совместимости интерфейсов в контрактном программировании. Следует заметить, что вследствие особенностей Spring AOP внедрение аспектов происходит во время выполнения кода, что сказывается на скорости работы приложения, в отличие от внедрения во время загрузки классов или компиляции. Таким образом, метод применим в малых и средних Web-приложениях, где снижение производительности будет приемлемым.

Список литературы

1. Safonov, V.O. Using aspect-oriented programming for trustworthy software development – Wiley Interscience: John Wiley & Sons, 2008. – 338 с.
2. Шаньгин, В. Ф. Информационная безопасность компьютерных систем и сетей: учеб. пособие / В.Ф. Шаньгин. – М: Форум: Инфра-М, 2008. – 416 с.

Зайкова Светлана Алексеевна, доцент кафедры системного программирования и компьютерной безопасности Гродненского государственного университета им. Янки Купалы, кандидат физико-математических наук, доцент, sunny@mf.grsu.by

Рыжко Олег Юрьевич, студент 4 курса специальности Компьютерная безопасность, факультет математики и информатики Гродненского государственного университета им. Янки Купалы, ryzhko_oj_11@mf.grsu.by