DOI: 10.24412/2413-2527-2023-234-58-66

Использование фреймворков семейства Spring Projects для разработки веб-приложений на платформе Java

В. В. Сакович, к.т.н. Г. И. Кожомбердиева
Петербургский государственный университет путей сообщения Императора Александра I
Санкт-Петербург, Россия
lampirg1@gmail.com, kgi-liizht@yandex.ru

к.т.н. Д. П. Бураков
независимый исследователь
Санкт-Петербург, Россия
burakovdmitry8@gmail.com

Аннотация. **Обсуждаются особенности разработки веб-приложений на платформе Java с использованием фреймворков семейства Spring Projects. Рассматриваются основные концепции Spring Framework: IoC-контейнер, Spring Scopes, Spring MVC и Spring AOP. Освещаются возможности, предоставляемые Spring Boot: запуск приложения, профили, конфигурационные файлы. Кроме того, уделяется внимание организации взаимодействия с базами данных с помощью Spring Data, а также использованию средств авторизации, предоставляемых Spring Security. В качестве демонстрационного примера веб-приложения, разработанного с использованием Spring, представлено клиент-серверное приложение для записи студентов на консультации и управления консультациями.**

Ключевые слова: **веб-приложение, платформа Java, аннотации Java, фреймворк, Spring Projects, Spring Framework, архитектура MVC.**

Введение

Широкое распространение веб-приложения получили в начале 2000-х годов в условиях бурного развития сети Интернет и гипертекстовой системы WWW [1]. Вследствие этого в области программной инженерии было выделено даже отдельное направление — веб-программирование, то есть разработка использующих веб-технологии информационных систем, представляющих собой полноценные веб-приложения, или частей этих систем, использующих веб-страницы для представления пользовательского интерфейса.

В настоящей статье рассматриваются особенности использования семейства фреймворков Spring Projects при разработке веб-приложений на платформе Java.

В качестве примера представлено демонстрационное приложение, функциональность которого ориентирована на решение практически значимой в вузовской среде задачи записи студентов на консультации и управления консультациями, а реализация — на показательное применение возможностей и преимуществ семейства фреймворков Spring Projects.

Статья содержит результаты выпускной квалификационной бакалаврской работы В. В. Саковича. Описание фреймворков семейства Spring Projects сопровождается отсылками к конкретным примерам их использования при разработке демонстрационного приложения. Приложение разработано в интегрированной среде разработки IntelliJ IDEA (с использованием в качестве дополнительных инструментов веб-приложения Spring Initializr и средства для сборки проекта Maven) и может рассматриваться как действующий прототип реальной системы.

Веб-приложения и средства их разработки

По своей структуре веб-приложения представляют собой приложения, построенные в соответствии с архитектурой «клиент-сервер». Структура типичного приложения представлена на рисунке 1 [2].

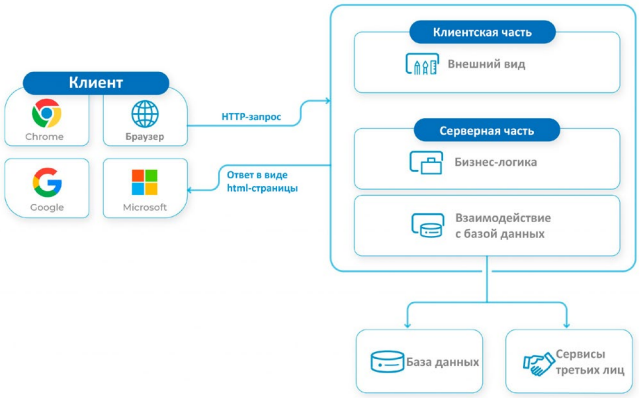

Рис. 1. Структура веб-приложения

Отличительной чертой веб-приложений является то, что клиентская часть выполняется в стандартном веб-браузере, причем клиентская и серверная части могут работать на разных операционных системах, то есть веб-приложения являются межплатформенными службами.

Клиентская часть веб-приложения реализует пользовательский интерфейс, формирует запросы к серверной части приложения и обрабатывает ответы от нее. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет ее клиенту по сети с использованием протокола HTTP. При этом серверная часть веб-приложения может выступать в качестве клиента других служб, например сервера базы данных или другого веб-приложения, расположенного на другом сервере.

Поскольку веб-приложения состоят из серверной и клиентской части, в веб-программировании выделяется программирование клиентской части, выполняемой непосредственно в браузере, и программирование серверной части, работающей на стороне веб-сервера. Для реализации как клиентской, так и серверной части могут быть использованы различные технологии по усмотрению разработчика или организации, внедряющей веб-приложение. Так, для реализации клиентской части, помимо языка разметки

HTML, активно используются таблицы стилей CSS и скриптовый язык JavaScript. На серверной стороне веб-приложений в основном применяются средства динамической генерации веб-страниц, например широко известный скриптовый язык PHP. Однако для разработки корпоративных веб-приложений, отличающихся, как правило, сложной структурой и разнообразием сервисов, предоставляемых конечным пользователям, применяются решения, основанные на использовании таких универсальных языков программирования, как Java.

Будучи высокоуровневым кроссплатформенным языком, язык Java используется во многих областях разработки программного обеспечения. При этом в стандартной редакции платформы Java (Java SE) не предусмотрены средства для удобной реализации серверных приложений. Для решения этой проблемы разработчиками Java была создана редакция платформы для разработки корпоративных систем Java EE (Java Platform, Enterprise Edition), включившая в себя такие технологии разработки серверной части веб-приложений, как JSP (Java Sever Pages) и сервлеты. Однако, ввиду того, что Java EE (с 2018 года — Jakarta EE) развивалась довольно медленно, а ее функциональность была относительно низкоуровневой, существовал запрос на альтернативные средства разработки серверных приложений, которые, использовав созданные в Java EE спецификации, позволяли бы создавать негромоздкие, легко масштабируемые, высокоуровневые веб-приложения.

Такой альтернативой стало семейство фреймворков Spring, существенно упростившее создание приложений и вследствие этого завоевавшее популярность среди разработчиков [3]. Данное семейство фреймворков является одним из самых востребованных средств для разработки веб-приложений [4], которое используется в таких компаниях, как Amazon, Google, Microsoft, Netflix и многих других [5].

Использование фреймворков как программных платформ, определяющих структуру разрабатываемой прикладной программной системы, облегчает разработку и объединение различных компонентов большого программного проекта. Как правило, облегчение разработки сложных систем достигается за счет использования каркасного подхода, при котором любая конфигурация программы строится на основе постоянной части, то есть каркаса, не зависящего от конфигурации и несущего в себе «гнезда», к которым подключаются сменные модули или точки расширения каркаса, определяющие специфическую функциональность разрабатываемой системы.

Семейство фреймворков Spring Projects

Изначально под термином Spring подразумевался только Spring Framework — фреймворк с открытым исходным кодом для Java-платформы, появившийся в середине 2000-х годов как альтернатива технологии Enterprise JavaBeans, которая поддерживает разработку серверных компонентов с бизнес-логикой и является частью Java EE. Однако по мере развития этого фреймворка на его основе сформировались новые фреймворки, которые образовали расширенное семейство Spring Projects [6]. Некоторые фреймворки данного семейства представлены на рисунке 2 [7].

Отметим фреймворки семейства, представляющие наибольший интерес в рамках настоящей статьи:

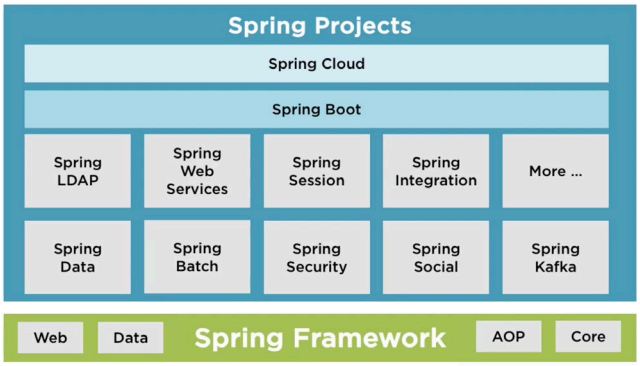

Рис. 2. Семейство фреймворков Spring Projects

• **Spring Framework** — основной фреймворк, использующийся во всех Spring-приложениях, предназначенный для управления ходом функционирования программы;

• **Spring Boot** — фреймворк, позволяющий быстро создавать и настраивать полноценные и готовые к запуску приложения, основанные на Spring;

• **Spring Data** — фреймворк для работы с базами данных, работающий с JDBC, JPA, а также NoSQL СУБД (например, MongoDB);

• **Spring Security** — фреймворк, предоставляющий средства для авторизации и аутентификации, а также базовую защиту от злоумышленных операций.

Для быстрой генерации каркасного проекта приложения, использующего фреймворки семейства Spring Projects, предназначен специальный инструмент — веб-приложение **Spring Initializr** [8]. Оно позволяет пользователю выбрать инструментальное средство для сборки проекта (например, Maven), версию Spring Boot, версию платформы Java, а также используемые средства семейства Spring Projects (и некоторых других инструментов). На основе выбранных пунктов меню и введенной информации Spring Initializr генерирует готовый zip-архив каркасного проекта приложения.

Возможности фреймворка Spring Framework

Рассмотрим полезные возможности Spring Framework, поддерживаемые ядром и внутренними пакетами этого основного фреймворка семейства. Описание средств, предоставляемых Spring Framework, сопровождается отсылками к примерам их использования В. В. Саковичем при разработке демонстрационного приложения.

Внедрение зависимостей IoC-контейнером

В решениях семейства Spring Projects активно применяется так называемая *инверсия управления* (Inversion of Control, IoC), которая также иногда называется *внедрением зависимостей* (Dependency Injection, DI). Это шаблон проектирования, при использовании которого зависимости экземпляров класса (то есть объекты, с которыми они взаимодействуют) определяются на основе аргументов конструктора (или метода, создающего экземпляр класса), а также полей объекта, значение которых определяется уже после создания экземпляра класса.

Внедрением зависимостей после создания экземпляра в Spring Framework занимается так называемый IoC-контейнер. IoC-контейнер представляет собой объект типа интерфейса *BeanFactory*. Как правило, при этом данный

объект создается на основе класса, реализующего производный интерфейс *ApplicationContext*. Например, в веб-приложениях используется автоматически создаваемый фреймворком Spring Boot экземпляр класса *AnnotationConfigServletWebServerApplicationContext*. При внедрении зависимостей Spring использует имеющийся в Java механизм рефлексии, в частности, активно применяются аннотации, опрашиваемые при выполнении приложения с помощью рефлексии [9].

Объекты, которые настраивает и внедряет IoC-контейнер, называются *Spring bean*-компонентами. Объявить Spring bean можно либо пометив его класс аннотацией *@Component*, либо использовав метод, помеченный аннотацией *@Bean* в классе, помеченным аннотацией *@Configuration*, либо используя xml-файлы [10]. Например, экземпляру класса *NotificationAspect* (который в демонстрационном приложении используется при формировании уведомления для отправки по электронной почте) для выполнения функций требуется объект типа интерфейса *EmailService*. Для этого классы, реализующие данный интерфейс, помечены аннотацией *@Component*. Во время выполнения программы IoC-контейнер создаст экземпляр класса, реализующего интерфейс *EmailService* и передаст его в качестве аргумента конструктору при создании экземпляра класса *NotificationAspect*.

Области применения Spring Scopes

Каждый объект Spring bean имеет свою *область применения* (scope), причем Spring Framework поддерживает шесть основных областей применения:

- **singleton** — на протяжении всей работы приложения существует только один Spring bean данного класса (выбирается по умолчанию);
- **prototype** — для каждого использующего его объекта внедряется свой Spring bean;
- **request** — для каждого HTTP-запроса создается свой Spring bean;
- **session** — для каждой сессии создается свой Spring bean;
- **application** — для каждого объекта типа интерфейса *ServletContext* создается свой Spring bean;
- **websocket** — для каждого объекта типа интерфейса *WebSocket* создается свой Spring bean.

Для приписывания области применения объекту Spring bean, необходимо применить аннотацию *@Scope* к классу, помеченному аннотацией *@Component* или методу, помеченному аннотацией *@Bean*. В качестве значения члена аннотации передается название области применения.

Для применения большинства областей scope фреймворку Spring необходимо использование прокси (объекта-посредника). Тип используемого прокси (на основе интерфейсов или на основе классов) также указывается в значении члена аннотации [11].

В качестве примера использования настройки области применения в демонстрационном приложении можно привести класс *TeacherController*, который содержит поле типа *ConsultationPatternListWrapper*, предназначенного для хранения данных об одном пользователе-преподавателе на протяжении нескольких запросов. Для этого используется область применения *session* и прокси на основе классов.

Архитектура Spring MVC

Spring Framework содержит два внутренних фреймворка для создания веб-приложений: **Spring MVC** и **Spring WebFlux** (предназначенный для реактивного веб-программирования).

Spring MVC основан на сервлете *DispatcherServlet*, который обрабатывает запросы, делегируя функции соответствующим компонентам Spring bean.

Архитектура MVC (Model-View-Controller), применяемая при проектировании программного обеспечения, представлена на рисунке 3 [12]. Она основана на взаимодействии трех типов компонентов [13]:

- **Model** (модель) содержит изменяемые данные, которые обновляет контроллер и которые передаются представлению;
- **View** (представление, вид) отвечает за отображение данных модели пользователю;
- **Controller** (контроллер) управляет моделью и отображением на основе входных данных.

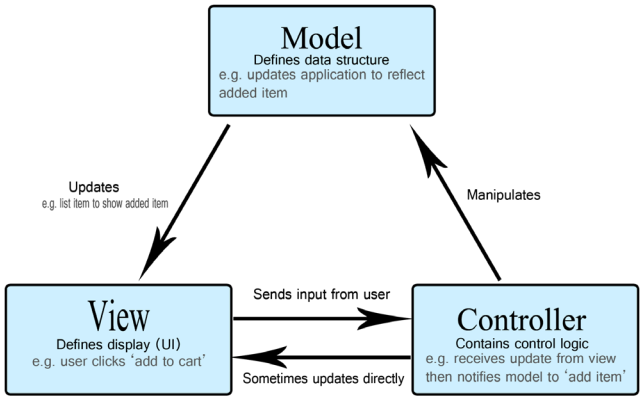

Рис. 3. Архитектура MVC

Экземпляр класса *DispatcherServlet* обрабатывает запрос в соответствии с этой архитектурой, как показано на рисунке 4 [14].

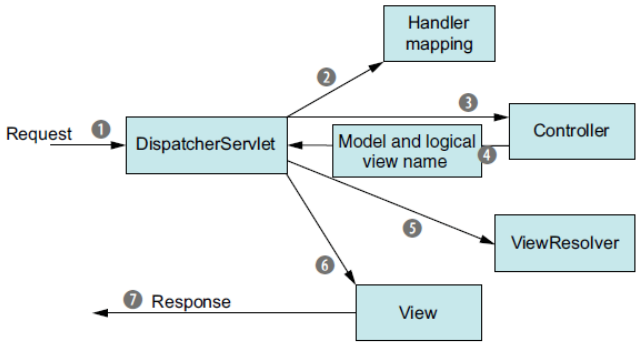

Рис. 4. Обработка HTTP-запроса экземпляром класса *DispatcherServlet*

Обработка запроса происходит следующим образом:

1. Экземпляр класса *DispatcherServlet* принимает HTTP-запрос.

2. На основе объекта типа интерфейса *HandlerMapping* определяется контроллер, которому будет передана обработка запроса.

3. Контроллер обрабатывает запрос.

4. В результате обработки запроса контроллером экземпляр *DispatcherServlet* получает модель и логическое имя представления.

5. Объект типа интерфейса *ViewResolver* выбирает представление по его имени.

6. Представление, данные которого основаны на модели, формирует HTTP-ответ.

7. Пользователю отправляется HTTP-ответ.

Представление может быть создано при помощи таких средств, как генератор веб-страниц **Thymeleaf** или технология **JSP** (Java (или Jakarta) Server Pages) [14].

Контроллер представляет собой Java-класс, помеченный аннотацией *@Controller*. Он объявляет методы, к которым применяется аннотация *@RequestMapping* (или одна из аннотаций, которая помечена данной аннотацией, например, *@GetMapping* или *@PostMapping*). На основе значений членов данной аннотации объект типа *HandlerMapping* сможет сопоставить HTTP-запрос соответствующему методу контроллера [15].

Примером контроллера в демонстрационном приложении является класс *TeacherController*. Этот класс объявлен как Spring bean-компонент, что позволяет внедрить в объект этого класса экземпляры других классов для выполнения различных функций. При этом в классе объявлены такие методы, как *getTeacherProfile*, *deleteConsultation* и т. д., к которым применяются аннотации *@GetMapping* или *@PostMapping*. Данные методы обрабатывают запрос, используя объект типа интерфейса *Model*, если необходимо передать пользователю данные.

Концепция Spring AOP

Для того чтобы неявным образом вызвать функцию после, перед или вместо выполнения определенного метода, Spring Framework использует *аспекты* и средства **Spring AOP** (Aspect Oriented Programming).

Внутренний фреймворк Spring AOP основан на **AspectJ** и использует идею прокси (объекта-посредника). Прокси подменяет собой изначальный объект и при вызове метода сначала выполняет свою логику, а затем делегирует полномочия изначальному методу. Принцип его работы приведен на рисунке 5 [16].

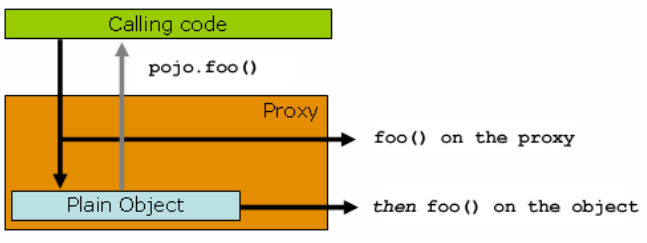

Рис. 5. Принцип работы прокси-объекта

Для того чтобы объявить класс в качестве аспекта, необходимо применить к нему аннотацию *@Aspect*. Для того чтобы метод аспекта был выполнен при вызове метода другого объекта, необходимо применить к нему одну из следующих аннотаций: *@Before* (перед), *@AfterReturning* (после при успешном завершении), *@AfterThrowing* (после при завершении в результате возникновения исключительной ситуации), *@After* (после) или *@Around* (перед и после). Метод или методы, при вызове которых должны отработать аспекты, определяются на основе значения члена одной из перечисленных аннотаций.

Такие методы в качестве аргумента могут принимать объект типа интерфейса *JoinPoint*, который содержит различную информацию об аспекте и методе, к которому применяется метод аспекта (например, аргументы метода). Если к методу аспекта применяется аннотация *@Around*, то он должен принимать в качестве аргумента объект типа интерфейса *ProceedingJoinPoint*, который содержит метод *proceed* для вызова оригинального метода [17].

Примером использования аспекта в демонстрационном приложении является уже упоминавшийся класс *NotificationAspect*, в котором объявлен метод *notifyStudentsAboutDeletion*. Метод помечен аннотацией *@After*, а в качестве значения члена аннотации устанавливается название метода удаления консультации. Таким образом, данный метод будет вызван после удаления консультации независимо от места вызова метода удаления.

Другие фреймворки семейства Spring Projects

Описание средств, которыми располагают фреймворки Spring Boot, Spring Data и Spring Security, сопровождается отсылками к примерам их использования В. В. Саковичем при разработке демонстрационного приложения.

Spring Boot

Фреймворк Spring Boot предоставляет возможность ***запуска*** веб-приложения при помощи вызова метода *run* класса *SpringApplication* из метода *main*. Класс, содержащий метод *main*, должен быть помечен аннотацией *@SpringBootApplicationю*.

При вызове метода *run* Spring Boot автоматически настроит множество параметров, а для веб-приложения дополнительно автоматически создаст сервер, используя встроенный контейнер сервлетов (по умолчанию — Tomcat).

Другой важной возможностью, поддерживаемой Spring Boot, являются ***профили.*** При разработке приложения может потребоваться использование разных Spring bean для одной и той же задачи. Например, в демонстрационном приложении имеется имитационный сервис оповещения по электронной почте, который оставляет сообщения в системе, но не отправляет сами письма. Однако на этапе эксплуатации должен использоваться настоящий сервис оповещений.

Реализовать такую возможность позволяет аннотация *@Profile*, которая принимает в качестве значения члена аннотации профиль или логически объединенное (операторами И, ИЛИ, НЕ) множество профилей. Аннотация применяется при настройке Spring bean (либо к классу, помеченному аннотацией *@Component*, либо к методу, помеченному аннотацией *@Bean*) [3]. Таким образом, имитационный сервис включается только при активном профиле *dev*, а настоящий сервис — при активном профиле *prod*.

Spring Boot позволяет настраивать приложение посредством ***конфигурационных файлов*** (расширения *properties* или *yml*). Отметим некоторые параметры, предоставляемые Spring Boot [18]:

- **server.port** — адрес порта, с которого сервер принимает запросы (по умолчанию — 8080);

• **spring.profiles.active** — активные профили приложения;

• **spring.security.user.password** — пароль пользователя, создаваемый по умолчанию;

• **server.servlet.session.timeout** — время действия сессии;

• **spring.mail.host** — хост SMTP-сервера.

Используя разделитель «---», можно настроить разные значения конфигурационных параметров для разных профилей. Для этого необходимо добавить конфигурационный параметр **spring.config.activate.on-profile** [3].

Spring Data

Фреймворк Spring Data содержит средства для работы с базами данных. Фреймворк предоставляет приложению интерфейс *Repository*, который является интерфейсом-маркером, с помощью которого выявляются потомки этого интерфейса. Наследующие интерфейсу *Repository* интерфейсы *CrudRepository* и *ListCrudRepository* объявляют базовые методы по работе с базой данных (CRUD — Create, Read, Update, Delete).

Для практического применения средств для работы с базами данных при разработке приложения необходимо создать интерфейс, наследующий *Repository*. Объявить метод интерфейса-репозитория можно двумя способами: указанием имени метода в соответствии с соглашением об именах или при помощи аннотации *@Query*, указав в качестве значения члена аннотации запрос к базе данных.

При первом подходе Spring Data предлагает множество вариантов создания комплексного запроса. Среди них поиск по какому-либо полю, поиск с условием, логические объединения (И, ИЛИ), сортировка значений, разбиение запроса на страницы и т. д. При этом Spring Data позволяет такому методу возвращать как единственный объект типа, соответствующего хранимым в репозитории данным, так и коллекцию таких объектов.

При запуске приложения Spring Data проанализирует всех потомков интерфейса *Repository* и, если у какого-либо интерфейса не будет реализации (и он не помечен аннотацией *@NoRepositoryBean*), обеспечит реализацию интерфейса с помощью механизма рефлексии в соответствии с соглашением об именах методов или на основе значения члена аннотации *@Query*. Однако, при необходимости, разработчик может создать реализацию интерфейса самостоятельно (в виде компонента Spring bean) [19].

В качестве примера автоматической реализации интерфейса в демонстрационном приложении можно привести интерфейс *TeacherRepository*, для которого на этапе выполнения программы создается прокси-объект типа *jdk.proxy4.$Proxy139*.

Интерфейс *TeacherRepository* объявляет метод для поиска консультации, который помечен аннотацией *@Query*. На этапе выполнения прокси на основе значения члена аннотации *@Query* реализует метод, выполняющий поиск консультации в базе данных. Помимо этого, прокси реализует метод *findByEmail* суперинтерфейса *PersonRepository* на основе имени метода.

Spring Security

Средства обеспечения безопасности и авторизации пользователей предоставляются фреймворком Spring Security и основаны на фильтре сервлетов, который делегирует полномочия по настройке безопасности набору фильтров *SecurityFilterChain*, как показано на рисунке 6 [20].

Набор фильтров безопасности может быть создан методом, помеченным аннотацией *@Bean* в классе, помеченным аннотацией *@Configuration*. Такой метод должен принимать экземпляр класса *HttpSecurity* в качестве аргумента [3].

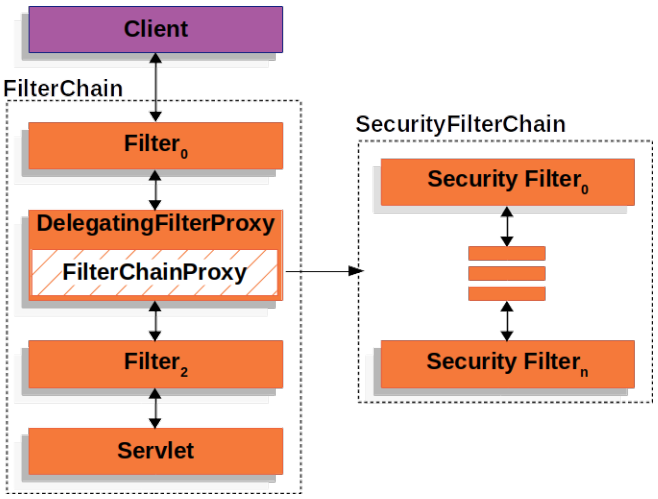

Рис. 6. Принцип работы средств безопасности Spring Security

Отметим некоторые методы класса *HttpSecurity* [21]:

• **securityMatcher** — устанавливает типы URL-адресов, при которых будет применятся набор фильтров;

• **authenticationProvider** — устанавливает источник аутентификации;

• **authorizeHttpRequests** — настраивает правила доступа пользователей к веб-страницам;

• **formLogin** — настраивает страницу входа в аккаунт;

• **logout** — настраивает страницу выхода из аккаунта.

В качестве примера настройки авторизации в демонстрационном приложении, структура и функциональность которого представлены в следующем разделе статьи, можно рассматривать класс *TeacherSecurityConfig*, который содержит три метода, помеченных аннотацией *@Bean*:

• **teacherDetailService** возвращает объект типа интерфейса *UserDetailsService*, который находит объект типа интерфейса *UserDetails*, предоставляющий следующую информацию (для поиска используются средства для работы с базой данных) [21]: имя пользователя; пароль; права доступа; различная информация о состоянии аккаунта.

• **teacherAuthenticationProvider** возвращает объект типа интерфейса *AuthenticationProvider*, который аутентифицирует пользователя. Для поиска пользователя используется объект типа интерфейса *UserDetailsService*.

• **teacherFilterChain** настраивает фильтры безопасности, в качестве источника аутентификации используя объект, возвращаемый предыдущим методом. Метод возвращает объект типа интерфейса *SecurityFilterChain*.

Веб-приложение для записи студентов на консультации и управления консультациями

Источником примеров использования средств фреймворков семейства Spring Projects, описываемых в предыдущих разделах, является демонстрационное приложение, позволяющее преподавателю организовать запись студентов

на свои консультации через Интернет. Преподаватели могут вносить сведения о предстоящих консультациях (в том числе — запланировать на будущее повторяющиеся, например еженедельные, консультации) и отменять их, просматривать списки студентов, записавшихся на консультации, а студенты — просматривать расписание консультаций, записываться на них и отменять свою запись. В необходимых случаях приложение оповещает об изменениях как преподавателей, так и студентов по электронной почте.

Структура приложения соответствует общей структуре веб-приложений, представленной на рисунке 1.

Клиентская часть приложения представлена html-страницами, доступными для просмотра в веб-браузере.

Серверная часть реализована на платформе Java с использованием описанных ранее возможностей фреймворков семейства Spring Projects. Серверная часть приложения реализует следующие функции:

• **Авторизация.** Функции приложения недоступны для неавторизированных пользователей, в процессе авторизации определяется две роли пользователей — студенты и преподаватели, которым доступны разные функции приложения.

• **Взаимодействие с базой данных.** Информация о студентах, преподавателях, консультациях и записях о них хранится в реляционной БД (используется СУБД H2, предоставляемая Spring по умолчанию). Реализованы средства для создания, чтения, обновления и удаления данных.

• **Запись студентов на консультации** позволяет авторизованному студенту записаться на консультацию или отменить свою запись.

• **Управление консультаций преподавателями** позволяет преподавателям создавать, просматривать и удалять консультации или планы консультаций.

• **Оповещение по электронной почте.** При записи студента на консультацию оповещается преподаватель, при удалении преподавателем консультации оповещаются все записанные на нее студенты.

На рисунках 7 и 8 представлены диаграммы прецедентов для разрабатываемого веб-приложения, полученные на этапе его логического проектирования при анализе постановки задачи.

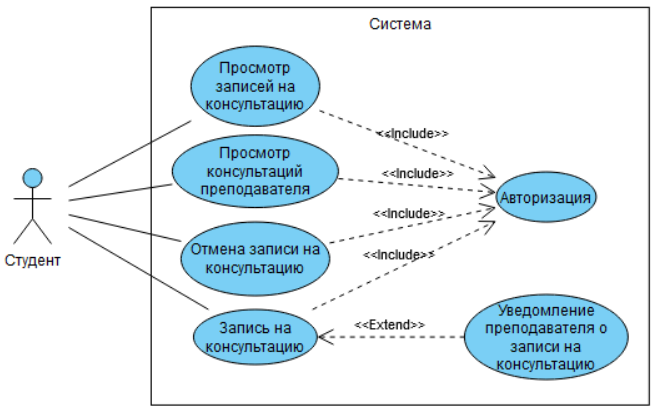

Рис. 5. Диаграмма прецедентов студента

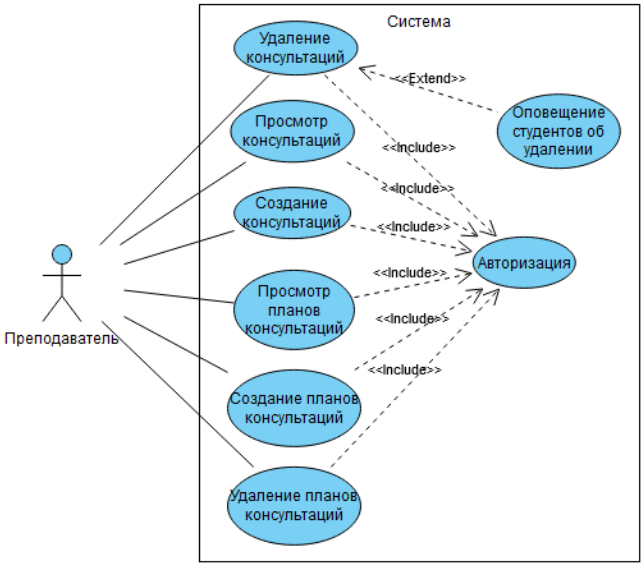

Рис. 6. Диаграмма прецедентов преподавателя

Исходя из выявленных прецедентов, для клиентской части разработано одиннадцать веб-страниц приложения, реализующих варианты взаимодействия с пользователями типа «студент» и «преподаватель». В качестве средства для реализации клиентской части выбран инструмент генерации веб-страниц на основании шаблона Thymeleaf.

На рисунке 9 представлена диаграмма компонентов разработанного веб-приложения.

Приложение включает в себя такие компоненты, как:

• общая клиентская часть, предназначенная для всех пользователей;

• интерфейс студента, предназначенный для работы студентов с веб-приложением;

• интерфейс преподавателя, предназначенный для работы преподавателей с веб-приложением.

• средства по взаимодействию с базой данных, предназначенные для манипуляции данными в БД;

• средства авторизации, предназначенные для авторизации пользователя и определения его типа;

• компонент записи на консультации, с помощью которого студенты могут пользоваться функциями, обеспечивающими запись на консультации;

• компонент управления консультациями, с помощью которого преподаватели могут пользоваться функциями, связанными с созданием, планированием и отменой консультаций;

• компонент оповещения по электронной почте студентов и преподавателей об изменениях в графике проведения консультаций.

Заключение

В настоящее время семейство фреймворков Spring Projects активно применяется для разработки корпоративных систем и веб-сервисов на платформе Java.

В статье обсуждаются особенности разработки веб-приложений с использованием средств фреймворков Spring Framework, Spring Boot, Spring Data и Spring Security. Кратко описывается инструмент Spring Initializr, применяемый для создания проектов приложений. Рассматриваются основные концепции Spring Framework: IoC-контейнер, Spring

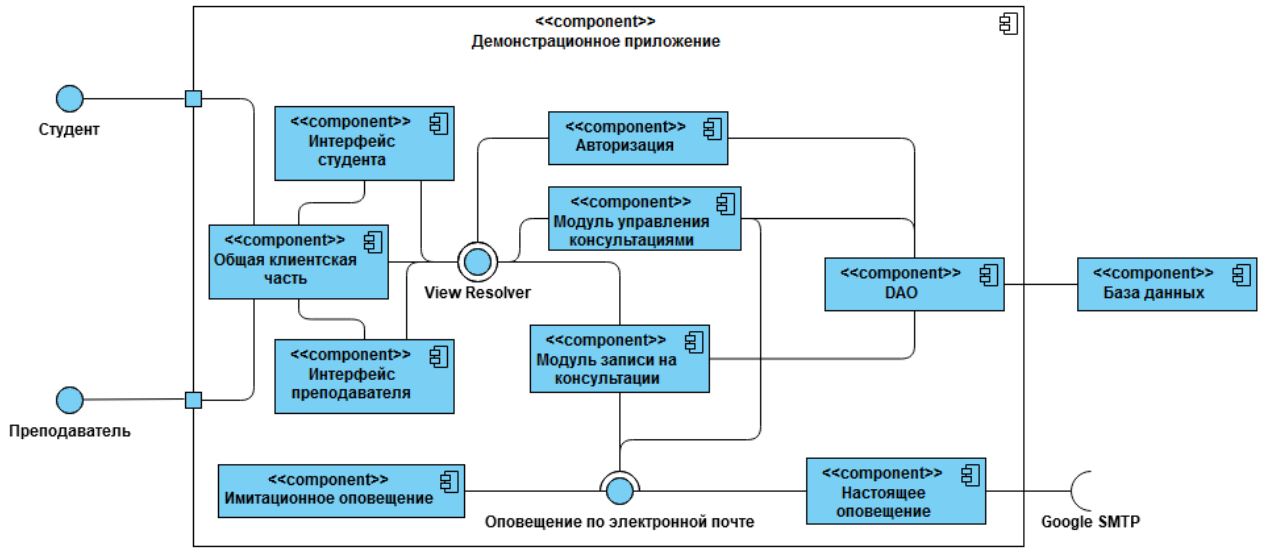

Рис. 7. Диаграмма компонентов веб-приложения

Scopes, Spring MVC и Spring AOP. Освещаются возможности, предоставляемые Spring Boot: запуск приложения, профили, конфигурационные файлы. Уделяется внимание организации взаимодействия с базами данных с помощью Spring Data и использованию средств авторизации, предоставляемых Spring Security.

Описание возможностей фреймворков семейства Spring Projects сопровождается отсылками к примерам их использования при разработке демонстрационного приложения веб-приложения, функциональность которого ориентирована на решение практически значимой в вузовской среде задачи записи студентов на консультации и управления консультациями. Приложение разработано и отлажено при подготовке выпускной квалификационной бакалаврской работы В. В. Саковича и может рассматриваться как действующий прототип реальной системы.

По мнению авторов, преимущества использования Spring Projects заключаются в следующем.

Во-первых, обилие специализированных фреймворков, входящих в семейство, позволяет упростить решение задач, с которыми сталкиваются разработчики при разработке сложных систем. Так, IoC-контейнер позволяет сосредоточиться на разработке классов, отвечающих за функциональность приложения, а их взаимодействие в ходе выполнения обеспечивает Spring Framework. Во-вторых, использование Spring Boot позволяет упростить запуск приложения путем автоматизации его настройки. Кроме того, фреймворки Spring Projects позволяют разработчику интегрировать приложение с разнообразным набором сервисов, включая базы данных и контейнеры сервлетов.

Таким образом, Spring Projects хорошо подходит для разработки сложных и многофункциональных систем, в том числе в области транспорта, например для решения задач комплексного управления логистикой и предоставления услуг клиентам такими компаниями в области транспорта, как ОАО «РЖД».

Использованные в статье материалы прошли апробацию на LXXXIII Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых «Транспорт: проблемы, идеи, перспективы», состоявшейся в ПГУПС в рамках фестиваля «Неделя науки — 2023». Материалы доклада В. В. Саковича, подготовленного под руководством его соавторов, рекомендованы к публикации кафедрой «Информационные и вычислительные системы».

ЛИТЕРАТУРА

1. Хомоненко, А. Д. Разработка Web-приложений для работы с базами данных: Учебное пособие / А. Д. Хомоненко, В. В. Рогальчук, А. В. Тырва; под ред. А. Д. Хомоненко. — Санкт-Петербург: ПГУПС, 2012. — 87 с.

2. Web Application Architecture: The Latest Guide 2022. — 2022. — 10 March // ClickIT. DevOps & Software Development URL: http://www.clickittech.com/devops/web-application-architecture (дата обращения 18.06.2023).

3. Уоллс, К. Spring в действии. Шестое издание = Spring in Action. Sixth Edition / пер. с англ. А. Н. Киселева. — Москва: ДМК Пресс, 2022. — 544 с.

4. Vermeer B. Spring Dominates the Java Ecosystem with 60% Using It for Their Main Applications — 2020. — 5 February // Snyk. Developer Security Platform. URL: http://snyk.io/blog/spring-dominates-the-java-ecosystem-with-60-using-it-for-their-main-applications (дата обращения 18.06.2023).

5. Why Spring? // Spring. URL: http://spring.io/why-spring (дата обращения 18.06.2023).

6. Projects // Spring. URL: http://spring.io/projects (дата обращения 18.06.2023).

7. Understanding Spring Framework and Spring Ecosystem // Pranay Bathini's Blog. — 2021. — 02 July. URL: http://www.pranaybathini.com/2021/07/understanding-spring-framework-and-ecosystem.html (дата обращения 18.06.2023).

8. Spring Initializr. URL: http://start.spring.io (дата обращения 18.06.2023).

9. Шилдт, Г. Java 8. Полное руководство. Девятое издание = Java: The Complete Reference. Ninth Edition / пер. с англ. и редакция И. В. Берштейна. — Москва: Издательский дом «Вильямс», 2015. — 1376 с.

10. Spring Framework. Core Technologies // Spring. URL: http://docs.spring.io/spring-framework/reference/core.html (дата обращения 18.06.2023).

11. Spring Framework. Bean Scopes // Spring. URL: http://docs.spring.io/spring-framework/reference/core/beans/factory-scopes.html (дата обращения 18.06.2023).

12. MVC — MDN Web Docs Glossary: Definitions of Web-related terms // MDN Web Docs. URL: http://developer.mozilla.org/en-US/docs/Glossary/MVC (дата обращения 18.06.2023).

13. Späth, P. Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications. — New York: Apress, 2020. — 460 p.

14. Walls, C. Spring in Action. Fourth Edition. — New York: Manning, 2014. — 624 c.

15. Spring Framework. Spring Web MVC // Spring. URL: http://docs.spring.io/spring-framework/reference/web/webmvc.html (дата обращения 18.06.2023).

16. Spring Framework. Proxying Mechanisms // Spring. URL: http://docs.spring.io/spring-framework/reference/core/aop/proxying.html (дата обращения 18.06.2023).

17. Spring Framework. Aspect Oriented Programming with Spring // Spring. URL: http://docs.spring.io/spring-framework/reference/core/aop.html (дата обращения 18.06.2023).

18. Common Application Properties // Spring. URL: http://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html (дата обращения 18.06.2023).

19. Spring Data Commons — Reference Documentation. Version 3.1.1 / O. Gierke, T. Darimont, C. Strobl, [et al.] // Spring. — Обновлено 16.06.2023. URL: http://docs.spring.io/spring-data/commons/docs/current/reference/html (дата обращения 18.06.2023).

20. Spring Security. Architecture // Spring. URL: http://docs.spring.io/spring-security/reference/servlet/architecture.html (дата обращения 18.06.2023).

21. Spring Security Docs 6.1.0 API // Spring. URL: http://docs.spring.io/spring-security/site/docs/current/api/index.html (дата обращения 18.06.2023).

DOI: 10.24412/2413-2527-2023-234-58-66

Using Spring Projects Family Frameworks for Developing Web Applications on Java Platform

V. V. Sakovich, PhD G. I. Kozhomberdieva
Emperor Alexander I St. Petersburg State Transport University
Saint Petersburg, Russia
lampirg1@gmail.com, kgi-liizht@yandex.ru

PhD D. P. Burakov
independent researcher
Saint Petersburg, Russia
burakovdmitry8@gmail.com

Abstract. **The article discusses the features of developing web applications on the Java platform using frameworks from the Spring Projects family. The basic concepts of the Spring Framework are considered: IoC container, Spring Scopes, Spring MVC, and Spring AOP. The possibilities provided by the Spring Boot are covered: application launch, profiles, and configuration files. In addition, attention is paid to the organization of interaction with databases using the Spring Data as well as using of authorization tools provided by the Spring Security. As a demo of a web application developed using the Spring features, the article provides a client server application for signing up students for a consultation and consultation management.**

Keywords: **web application, Java platform, Java annotation, framework, Spring Projects, Spring Framework, MVC architecture.**

REFERENCES

1. Khomonenko A. D., Rogalchuk V. V., Tyrva A. V. Development of Web Applications for Working with Databases: Study guide [Razrabotka Web-prilozheniy dlya raboty s bazami dannykh: Uchebnoe posobie]. Saint Petersburg, PSTU, 2012, 87 p.

2. Web Application Architecture: The Latest Guide 2022, *ClickIT. DevOps & Software Development*. Published online at March 10, 2022. Available at: http://www.clickittech.com/devops/web-application-architecture (accessed 18 Jun 2023).

3. Walls C. Spring in Action. Sixth Edition [Spring v deystvii. Shestoe izdanie]. Moscow, DMK Press Publishing House, 2022, 544 p.

4. Vermeer B. Spring Dominates the Java Ecosystem with 60% Using It for Their Main Applications, *Snyk. Developer Security Platform*. Published online at February 05, 2020. Available at: http://snyk.io/blog/spring-dominates-the-java-ecosystem-with-60-using-it-for-their-main-applications (accessed 18 Jun 2023).

5. Why Spring, *Spring*. Available at: http://spring.io/why-spring (accessed 18 Jun 2023).

6. Projects, *Spring*. Available at: http://spring.io/projects (accessed 18 Jun 2023).

7. Understanding Spring Framework and Spring Ecosystem, *Pranay Bathini's Blog*. Published online at July 02, 2021. Available at: http://www.pranaybathini.com/2021/07/understanding-spring-framework-and-ecosystem.html (accessed 18 Jun 2023).

8. Spring Initializr. Available at: http://start.spring.io/ (accessed 18 Jun 2023).

9. Schildt H. Java: The Complete Reference. Ninth Edition [Java 8. Polnoe rukovodstvo. Devyatoe izdanie]. Moscow, Williams Publishing House, 2015, 1376 c.

10. Spring Framework. Core Technologies, *Spring*. Available at: http://docs.spring.io/spring-framework/reference/core.html (accessed 18 Jun 2023).

11. Spring Framework. Bean Scopes, *Spring*. Available at: http://docs.spring.io/spring-framework/reference/core/beans/factory-scopes.html (accessed 18 Jun 2023).

12. MVC — MDN Web Docs Glossary: Definitions of Web-related terms, *MDN Web Docs*. Available at: http://developer.mozilla.org/en-US/docs/Glossary/MVC (accessed 18 Jun 2023).

13. Späth P. Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications. New York, Apress, 2020, 460 p.

14. Walls C. Spring in Action. Fourth Edition. New York, Manning, 2014, 624 p.

15. Spring Framework. Spring Web MVC, *Spring*. Available at: http://docs.spring.io/spring-framework/reference/web/webmvc.html (accessed 18 Jun 2023).

16. Spring Framework. Proxying Mechanisms, *Spring*. Available at: http://docs.spring.io/spring-framework/reference/core/aop/proxying.html (accessed 18 Jun 2023).

17. Spring Framework. Aspect Oriented Programming with Spring, *Spring*. Available at: http://docs.spring.io/spring-framework/reference/core/aop.html (accessed 18 Jun 2023).

18. Common Application Properties, *Spring*. Available at: http://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html (accessed 18 Jun 2023).

19. Gierke O., Darimont T., Strobl C., et al. Spring Data Commons — Reference Documentation. Version 3.1.1, *Spring*. Last updated at June 16, 2023. Available at: http://docs.spring.io/spring-data/commons/docs/current/reference/html (accessed 18 Jun 2023).

20. Spring Security. Architecture, *Spring*. Available at: http://docs.spring.io/spring-security/reference/servlet/architecture.html (accessed 18 Jun 2023).

21. Spring Security Docs 6.1.0 API, *Spring*. Available at: http://docs.spring.io/spring-security/site/docs/current/api/index.html (accessed 18 Jun 2023).