МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет по учебной практике
Тема НИР: Разработка, развёртка и поддержка приложения, написанного на Java с использованием Spring.

Студентка гр. 5304 _____ Фомченко О.А.

Санкт-Петербург
2020

ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является изучение фреймворка Spring, получение опыта разработки, развертывания и поддержки web-приложения, написанного на Java с использованием Spring и получение четкого, развернутого ответа на вопросы: Из чего состоит Spring? Как он работает? Как написать на нем приложение?

Эти вопросы важны по той простой причине, что на данный момент Spring, без сомнения, является самым популярным инструментом для разработки web-приложений на основе языка Java, который является вторым по популярности языком программирования в мире.

Основная проблема в данной теме заключается в том, что Spring, это огромный фреймворк с десятками различных модулей. К тому же, он используется для разработки не только web-приложений и не только на Java. Но, в рамках текущей задачи, мы ограничимся использованием и описанием только основных модулей web-разработки на языке Java.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

В ходе работы были получены следующие результаты:

- Изучены и описаны основные модули Spring;
- Написано web-приложение, использующее данные модули;
- К приложению была подключена реляционная база данных;
- К приложению была подключена система миграции баз данных;
- Приложение было интегрировано с системой контроля версий;
- Приложение было интегрировано с системой анализа качества кода;
- Приложение было протестировано интеграционными и модульными тестами;

2. SPRING FRAMEWORK

2.1. Введение в Spring:

Spring - легкий, но мощный фреймворк для разработки приложений. До его появления, приложения разрабатывались с помощью стандартов JEE, с чем было связанно много проблем:

- Код становился все сложнее по мере развития приложения;
- Производительность системы страдала от тяжести приложений;
- Проблема поиска компонента.

Все оказалось решено с появлением Spring. Фреймворк заметили на рынке благодаря модульности его базового функционала. Его можно разделить на разные модули, каждый из которых выполняет свою функцию.

Другими словами, Spring может быть рассмотрен как коллекция меньших фреймворков. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании. Они делятся на структурные элементы типовых комплексных приложений.

2.2. Основные модули:

- Inversion of Control: конфигурирование компонентов приложений и управление жизненным циклом Java-объектов.
- Модуль аспектно-ориентированного программирования: работает с функциональностью, которая не может быть реализована возможностями объектно-ориентированного программирования.
- Модуль доступа к данным: работает с системами управления реляционными базами данных на Java-платформе, используя JDBC- и ORM-средства.
- Модуль управления транзакциями: координация различных API управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java.
- Модуль MVC: каркас, основанный на HTTP и сервлетах, предоставляющий множество возможностей для расширения и настройки.
- Модуль удалённого доступа: конфигурируемая передача Java-объектов через сеть в стиле RPC, поддерживающая RMI, CORBA, HTTP-based протоколы, включая web-сервисы.
- Модуль аутентификации и авторизации: конфигурируемый инструментарий процессов аутентификации и авторизации, поддерживающий много

популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект Spring Security.
- Модуль удалённого управления: конфигурируемое представление и управление Java-объектами для локальной или удалённой конфигурации с помощью JMX.
- Модуль работы с сообщениями: конфигурируемая регистрация объектов-слушателей сообщений для прозрачной обработки сообщений из очереди сообщений с помощью JMS, улучшенная отправка сообщений по стандарту JMS API.
- Тестирование: каркас, поддерживающий классы для написания модульных и интеграционных тестов.

2.2.1. Inversion of Control:

Центральная часть Spring Framework, которая предоставляет средства конфигурирования и управления объектами Java с помощью рефлексии. Контейнер отвечает за управление жизненным циклом объекта: создание объектов, вызов методов инициализации и конфигурирование объектов путём связывания их между собой.

Объекты, создаваемые контейнером, также называются управляемыми объектами (beans). Обычно, конфигурирование контейнера, осуществляется путём внедрения аннотаций, но также, есть возможность, загрузить XML-файлы, содержащие определение bean’ов и предоставляющие информацию, необходимую для создания bean’ов.

Объекты могут быть получены одним из двух способов:

- Поиск зависимости — шаблон проектирования, в котором вызывающий объект запрашивает у объекта-контейнера экземпляр объекта с определённым именем или определённого типа.
- Внедрение зависимости — шаблон проектирования, в котором контейнер передает экземпляры объектов по их имени другим объектам с помощью конструктора, свойства или фабричного метода.

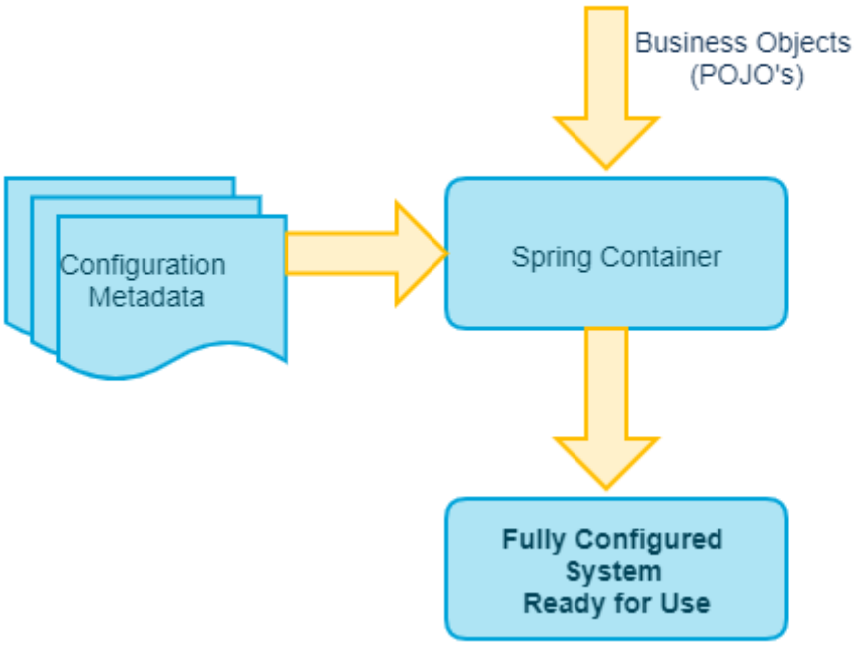

Рисунок 1. Spring Inversion Of Controle.

2.2.2. MVC:

Класс DispatcherServlet является основным контроллером фреймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.

Spring MVC является фреймворком, ориентированным на запросы. В нем определены стратегические интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса — быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают.

MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с Servlet API. Эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:

- HandlerMapping: выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.
- HandlerAdapter: вызов и выполнение выбранного метода обработки входящего запроса.
- Controller: включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные

ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.

- View: ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.
- ViewResolver: выбор, какое именно View должно быть показано клиенту.
- HandlerInterceptor: перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется DispatcherServlet-ом).
- LocaleResolver: получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.
- MultipartResolver: обеспечивает Upload — загрузку на сервер локальных файлов клиента.

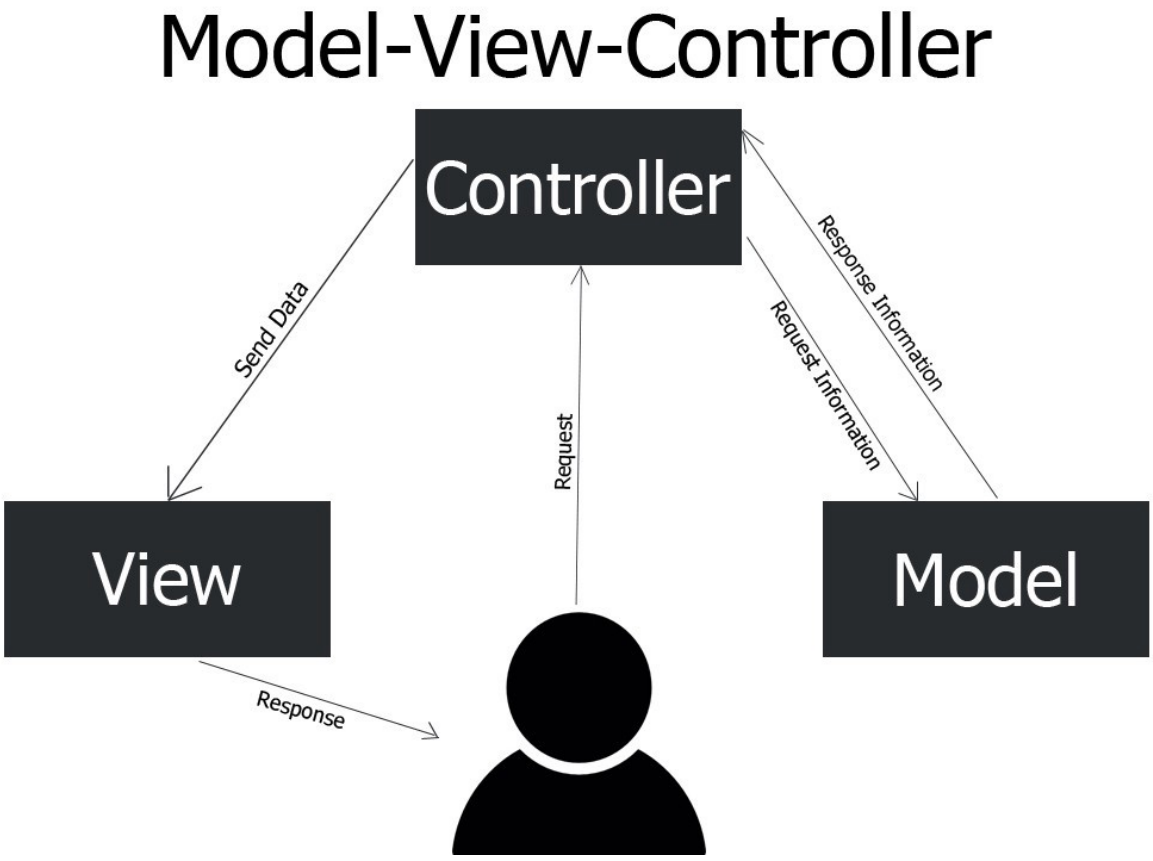

Рисунок 2. MVC

2.2.3. Доступ к данным:

Spring предоставляет свой слой доступа к базам данных посредством JDBC, который и использовался в разработанном приложении.

Кроме того, он поддерживает все популярные ORM: Hibernate, JPA, JDO, EclipseLink, iBatis, Apache OJB, Apache Cayenne и т. п.

Для всех этих фреймворков, Spring предоставляет такие особенности:

- Управление ресурсами — автоматическое получение и освобождение ресурсов базы данных.
- Обработка исключений — перевод исключений при доступе к данным в исключения Spring-а.
- Транзакционность — прозрачные транзакции в операциях с данными.
- Распаковка ресурсов — получение объектов базы данных из пула соединений.
- Абстракция для обработки BLOB и CLOB.

3. РЕЗУЛЬТАТЫ РАБОТЫ В ВЕСЕННЕМ СЕМЕСТРЕ

3.1. Общее описание результатов.

В результате работы в весеннем семестре были изучены основные модули Spring Framework и разработано web-приложение на их основе.

Результаты изучения модулей представлены выше.

3.2. Описание приложения.

Приложение написано на языке Java с использованием следующих технологий:

- Spring MVC;
- Inversion of Control;
- Spring AOP;
- Spring Security;
- Spring Boot;
- Spring Test;
- JDBC;
- Freemarker;
- Gradle;

- JSON Formater;
- Jupiter;
- FlyWay;
- PostgreSQL;
- SonarQube;

Само приложение представляет из себя инструмент для имплементации Agile flow при разработке приложений. В нем можно создавать тикеты разных типов, назначать исполнителей, переводить их из одного состояния в другое и т.д. Фактически, оно является упрощенным аналогом Jira.

В приложении есть системы регистрации, аутентификации и администрирования. Так же присутствует система ролей.

В качестве IDE использовалась Intellij IDEA.

3.3.Структура приложения.

Структура приложения показана на следующей UML-диаграмме:

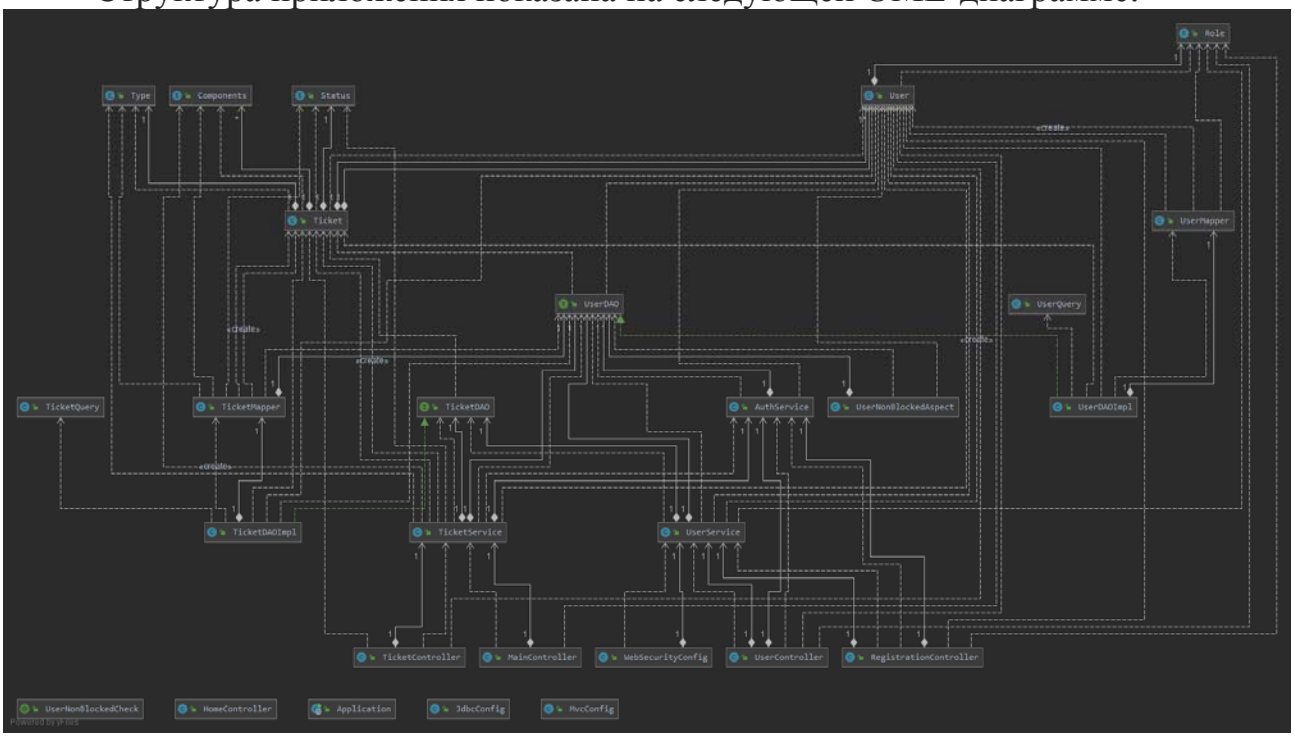

Рисунок 4. UML диаграмма.

3.4. Исходный код.

Так как приложение написано с использованием MVC, то оно соответствует определенной структуре, где view отвечает за front-end, model является передаваемыми сущностями, controller отвечает за получение request и передачу обратно response, за обработку request отвечает service, за взаимодействие с базами данных отвечает DAO.

Исходный код находится на github по следующей ссылке:
https://github.com/legiq/trackingBoard/tree/demo

3.5. Тестирование.

Приложение было протестировано при помощи интеграционных (с использованием тестовой базы данных) и модульных тестов с покрытием кода более 95%. Все тесты завершены успешно, приложение работает правильно.

4. ПЛАН РАБОТЫ НА ОСЕННИЙ СЕМЕСТР

План работы на весенний семестр расписан в таблице 1.

Таблица 1. План работы на весенний семестр

№ п/п	Наименование работ	Срок выполнения
1	Углубленное изучение предметной области и инструментов, детализация поставленных задач, обоснование актуальности разработки	03.09 – 29.09
2	Выбор метода достижения поставленных задач	01.10-15.10
3	Расширение функционала приложения и его подготовка к выпуску	16.10-15.11
4	Настройка сервера, настройка приложения для конкретного сервера, реализация выбранного решения	15.11-31.11
5	Подведение результатов работы и оформление НИР	01.12-31.12

5. ВЫВОД

В результате работы в осеннем семестре был изучен Spring Framework, и на основании полученных знаний был разработан каркас приложения, который в дальнейшем будет расширяться и выпускаться. Существующий функционал был протестирован на правильность выполнения логики и взаимодействие базой данных.

Так же приложение было проверено SonarQube’ом на качество кода. Для стабильного взаимодействия с базой данных была интегрирована система мираций, которая позволяет отслеживать внесенные в неё изменения, и в случае необходимости проводить откат.

В заключение могу сказать, что все поставленные цели были достигнуты и результат вполне удовлетворителен.

1.1.Список литературы

1. Spring documentation // . URL : https://spring.io/docs.
2. How to CI and CD Application Using GitHub Actions // Medium. URL: https://blog.bitsrc.io/https-medium-com-adhasmana-how-to-do-ci-and-cd-of-node-js-application-using-github-actions-860007bebae6 (дата обращения: 14.12.2019).
3. Continuous Integration для новичков // Хабр. URL: https://habr.com/ru/post/352282/ (дата обращения: 14.12.2019).
4. Applying CI/CD to Java Apps Using Spring Boot .URL: https://dzone.com/articles/applying-cicd-to-java-apps-using-spring-boot.
5. Микросервисная архитектура, Spring Cloud и Docker. URL: https://habr.com/ru/post/280786/.
6. Знакомимся со Spring Framework // . URL: https://www.kv.by/archive/index2009291108.htm.
7. Spring Boot // . URL: https://spring-projects.ru/projects/spring-boot/.
8. FlyWay //. URL: https://flywaydb.org/documentation/plugins/springboot
9. SonarQube //. URL: https://docs.sonarqube.org/latest/