



# Разработчик на Spring Framework

Курс о разработке веб-приложений на Spring, о фреймворках и вспомогательных технологиях Spring.

Длительность курса: 181 академический час

## 1 Введение в Spring Framework

### Цели занятия:

ориентироваться в проектах Spring для дальнейшего изучения;  
применять принцип IoC при написании классов и тестов;  
создавать контекст Spring;  
определять Spring Beans в контексте;  
организовывать правильный DI.

### Краткое содержание:

Spring Framework и его Spring Projects;  
IoC в общем виде;  
IoC-контейнер в Spring (контекст);  
XML-контекст со Spring Beans и DI между ними.

### Домашние задания

- 1 Приложение по проведению тестирования студентов - только вывод вопросов и вариантов ответа (если имеются)

Цель: Создать приложение с помощью Spring IoC, чтобы познакомиться с основной функциональностью IoC, на которой строится весь Spring.

Результат: простое приложение, сконфигурированное XML-контекстом.

#### Описание задания:

В ресурсах хранятся вопросы и различные ответы к ним в виде CSV файла (5 вопросов). Вопросы могут быть с выбором из нескольких вариантов или со свободным ответом - на Ваше желание и усмотрение.

Приложение должна просто вывести вопросы теста из CSV-файла с возможными вариантами ответа (если имеются).

#### Требования:

0. В приложении должна присутствовать объектная модель (отдаём предпочтение

объектам и классам, а не строчкам и массивам/ спискам строчек).

1. Все классы в приложении должны решать строго определённую задачу (см. п. 18-19 "Правила оформления кода.pdf", прикреплённые к материалам занятия).
2. Контекст описывается XML-файлом.
3. Все зависимости должны быть настроены в ЮС контейнере.
4. Имя ресурса с вопросами (CSV-файла) необходимо захардкодить строчкой в XML-файле с контекстом.
5. CSV с вопросами читается именно как ресурс, а не как файл.
6. Scanner, PrintStream и другие стандартные типы в контекст класть не нужно!
7. Весь ввод-вывод осуществляется на английском языке.
8. Крайне желательно написать юнит-тест какого-нибудь сервиса (оцениваться будет только попытка написать тест).
9. Помним - "без фанатизма".

Опционально (задание со "звёздочкой"):

1\*. Приложение должно корректно запускаться с помощью "java -jar"

Задание сдаётся в виде ссылки на pull-request в чат с преподавателем в личном кабинете ОТУС, не в Slack!

Инструкция по работе с pull-request-ами находится в документе "Настройка окружения.pdf" в материалах занятия. Вопросы можно задавать в чате, но для оперативности рекомендуем Slack.

Код, написанный в данном ДЗ будет использоваться дальше в домашних заданиях #2 (Занятие #2), #3 (Занятие #4), #4 (Занятие #5)

---

## 2 **Конфигурирование Spring-приложений**

### **Цели занятия:**

конфигурировать Spring-приложения в современном Java-based стиле (как сейчас и все пишут); ориентироваться в многослойной и луковой (Onion) архитектурах;

пользоваться Spring Expression Language (SpEL); задавать параметры приложения с помощью .properties файлов.

### **Краткое содержание:**

Java-based конфигурацию контекста Spring;  
Annotation-based конфигурацию контекста;  
аннотация стереотипов, @Autowired и многие другие;  
многослойная и луковая (Onion) архитектура;  
SpEL;  
параметры конфигурации из .properties-файлов

### **Домашние задания**

- 1 Приложение по проведению тестирования студентов (с самим тестированием)

Цель: Цель: конфигурировать Spring-приложения современным способом, как это и делается в современном мире

Результат: готовое современное приложение на чистом Spring

Новый функционал:

Программа должна спросить у пользователя фамилию и имя, спросить 5 вопросов из CSV-файла и вывести результат тестирования.

Выполняется на основе предыдущего домашнего задания + , собственно, сам функционал тестирования.

Требования:

1. Переписать конфигурацию в виде Java + Annotation-based конфигурации.
2. Добавить функционал тестирования студента.
3. Добавьте файл настроек для приложения тестирования студентов.
4. В конфигурационный файл можно поместить путь до CSV-файла, количество правильных ответов для зачёта - на Ваше усмотрение.
5. Если Вы пишете интеграционные тесты, то не забудьте добавить аналогичный файл и для тестов.

6. Scanner, PrintStream и другие стандартные типы в контекст класть не нужно!
7. Ввод-вывод на английском языке.
8. Помним, "без фанатизма" :)

Задание сдаётся в виде ссылки на pull-request в чат с преподавателем.

Вопросы можно задавать в чате, но для оперативности рекомендуем Slack.

Код, написанный в данном ДЗ будет использоваться дальше в домашних заданиях №3 (Занятие №4), №4 (Занятие №5)  
Данное задание засчитывает ДЗ №1 (Занятие №1).

Если Вы хотите засчитать, то обязательно пришлите ссылку в чат соответствующего предыдущего занятия.

---

### 3 ЛикБез - Основы архитектуры ПО

---

### 4 "Чёрная магия" Spring Boot

#### **Цели занятия:**

ориентироваться в возможностях Spring Boot для различных функциональностей и технологий;  
максимально быстро создавать production-grade standalone Spring-приложения с помощью Spring Boot Starters;  
писать автоконфигурации и использовать существующие;  
писать property в YAML-формате.

#### **Краткое содержание:**

Spring Boot и зачем он;  
проект и научимся собирать артефакт;  
pom-файл, стартеры, структура проекта;  
пример применения – Rest-сервис;  
пример теста;  
разные интересные возможности.

#### **Домашние задания**

## 1 Перенести приложение для тестирования студентов на Spring Boot

Цель: Цель: использовать возможности Spring Boot, чтобы разрабатывать современные приложения, так, как их сейчас и разрабатывают.

Результат: Production-ready приложение на Spring Boot

Это домашнее задание выполняется на основе предыдущего.

1. Создать проект, используя Spring Boot Initializr (<https://start.spring.io>)
2. Перенести приложение проведения опросов из прошлого домашнего задания.
3. Перенести все свойства в application.yml
4. Локализовать выводимые сообщения и вопросы (в CSV-файле). MessageSource должен быть из автоконфигурации Spring Boot.
5. Сделать собственный баннер для приложения.
6. Перенести тесты и использовать spring-boot-test-starter для тестирования

\*Опционально:

- использовать ANSI-цвета для баннера.
- если Ваш язык отличается от русского и английского - локализовать в нём.

Коммитить wrapper или нет в репозиторий - решать Вам.

Задание сдаётся в виде ссылки на pull-request в чат с преподавателем.

Вопросы можно задавать в чате, но для оперативности рекомендуем Slack.

Написанное приложение будет использоваться в ДЗ №4 (к занятию №5).

Данное задание засчитывает ДЗ №1 (к занятию №1) и ДЗ №2 (к занятию №2).

Если Вы хотите засчитать, то обязательно пришлите ссылку в чат соответствующего предыдущего занятия.

---

## 5 AOP, Spring AOP

### Цели занятия:

использовать аспектно-ориентированное программирование (там где нужно),  
видеть в коде ключевую функциональность Spring - Spring AOP,  
реализовывать в приложениях crosscutting-функциональность с помощью Spring AOP

### Краткое содержание:

crosscutting-функциональность;  
AOP и как устроены Proxy;  
Advices и где их применять;  
Join point и Pointcut, Pointcuts в стиле AspectJ;  
свои аспекты;  
best practices, связанные с AOP

---

## 6 Продвинутая конфигурация (часть 1) - Scopes, Lifecycle

### Цели занятия:

разбираться в областях видимости (Scopes) бинов;  
использовать возможность жизненного цикла бинов.

### Краткое содержание:

Scopes (области видимости);  
Lifecycle (жизненный цикл) бинов;  
Также будут рассмотрены дополнительные возможности для конфигурирования приложений;

---

## 7 Продвинутая конфигурация (часть 2) - Conditionals, Events, тестирование

### Цели занятия:

использовать Best Practices для конфигурирования Spring Boot-приложений;  
использовать события (Events) для передачи управления;  
эффективно тестировать Spring Boot-приложения.  
писать приложения с использованием Spring Shell.

### Краткое содержание:

профили;  
@Conditional-аннотации;  
@SpringBootTest;  
приложение на Spring Shell.

## Домашние задания

- 1 Перевести приложение для проведения опросов на Spring Shell

Цель: Цель: После выполнения ДЗ вы сможете использовать Spring Shell, чтобы писать интерфейс приложения без Web.

Результат: Приложение на Spring Shell

Домашнее задание выполняется на основе предыдущего.

Необходимо:

1. Подключить Spring Shell, используя spring-starter.
2. Написать набор команд, позволяющий проводить опрос.
3. Написать Unit-тесты с помощью spring-boot-starter-test, учесть, что Spring Shell в тестах нужно отключить.

Набор команд зависит только от Вашего желания. Вы можете сделать одну команду, запускающую Ваш Main, а можете построить полноценный интерфейс на Spring Shell.

Локализовывать команды Spring Shell НЕ НУЖНО (хотя можно, но это долго и непросто).

Задание сдаётся в виде ссылки на pull-request в чат с преподавателем.

Вопросы можно задавать в чате, но для оперативности рекомендуем Slack.

Данное задание НЕ засчитывает предыдущие!

Это домашнее задание больше нигде не будет использоваться. Но интерфейс Spring Shell мы будем использовать в дальнейшем.

---



**8 Разбор домашних заданий, QnA**

**Цели занятия:**

писать код с учётом Best Practices;  
не допускать частых ошибок;  
получить ответы на вопросы.

**Краткое содержание:**

типичные ошибки при выполнении ДЗ;  
наставники ответят на ваши вопросы.

### 1 DAO на Spring JDBC

#### Цели занятия:

эффективно использовать JDBC вместе со Spring JDBC для разработки приложений с мощностью чистого SQL; правильно применять паттерн DAO для подключения к БД; пользоваться embedded БД для написания тестов и при разработке простых приложений.

#### Краткое содержание:

паттерн DAO;  
необходимость кода для чистого JDBC;  
простое DAO на Spring JDBC с минимумом boilerplate кода.

#### Домашние задания

- 1 Создать приложение хранящее информацию о книгах в библиотеке

Цель: Цель: использовать возможности Spring JDBC и spring-boot-starter-jdbc для подключения к реляционным базам данных

Результат: приложение с хранением данных в реляционной БД, которое в дальнейшем будем развивать

Это домашнее задание выполняется НЕ на основе предыдущего.

1. Использовать Spring JDBC и реляционную базу (H2 или настоящую реляционную БД). Настоятельно рекомендуем использовать NamedParametersJdbcTemplate
2. Предусмотреть таблицы авторов, книг и жанров.
3. Предполагается отношение многие-к-одному (у книги один автор и жанр). Опциональное усложнение - отношения многие-ко-многим (у книги может быть много авторов и/или жанров).
4. Интерфейс выполняется на Spring Shell (CRUD книги обязательны, операции с авторами и жанрами - как будет удобно).
5. Скрипт создания таблиц и скрипт заполнения

данными должны автоматически запускаться с помощью spring-boot-starter-jdbc.

6. Написать тесты для всех методов DAO и сервиса работы с книгами.

Рекомендации к выполнению работы:

1. НЕ делать AbstractDao.
2. НЕ делать наследования в тестах

Это домашнее задание является основой для следующих.

---

## 2 Основы ORM, JPA, Hibernate как провайдер JPA

### Цели занятия:

применять JPA для описания маппинга классов-entities на таблицы реляционной БД;  
использовать Hibernate в качестве JPA Vendor.

### Краткое содержание:

JPA, как прослойку между приложением и ORM-фреймворком;  
JPA mapping сущности (entity) на БД.

---

## 3 JPQL, Spring ORM, DAO на основе Spring ORM + JPA

### Цели занятия:

писать ORM DAO с помощью Spring ORM + JPA + Hibernate (в качестве JPA Vendor-a) в Spring приложениях;  
использовать JPQL (аналог HQL) для построения SQL-подобных запросов.

### Краткое содержание:

интерфейсы JPA;  
JPQL;  
функционал Spring ORM для JPA.

### Домашние задания

- 1 Переписать приложение для хранения книг на ORM

Цель: Цель: полноценно работать с JPA + Hibernate для подключения к реляционным БД посредством ORM-фреймворка  
Результат: Высокоуровневое приложение с JPA-маппингом сущностей

Домашнее задание выполняется переписыванием предыдущего на JPA.

Требования:

1. Использовать JPA, Hibernate только в качестве JPA-провайдера.
2. Для решения проблемы N+1 можно использовать специфические для Hibernate аннотации @Fetch и @BatchSize.
3. Добавить сущность "комментария к книге", реализовать CRUD для новой сущности.
4. Покрыть репозитории тестами, используя H2 базу данных и соответствующий H2 Hibernate-диалект для тестов.
5. Не забудьте отключить DDL через Hibernate
6. @Transactional рекомендуется ставить только на методы сервиса.

Это домашнее задание будет использоваться в качестве основы для других ДЗ  
Данная работа не засчитывает предыдущую!

#### 4 Транзакции, Spring Tx

##### Цели занятия:

применять особенности транзакции в реляционных БД для правильной разработки слоя DAO;  
использовать декларативное и императивное управление транзакциями в Spring-приложениях с помощью Spring Tx.

##### Краткое содержание:

ACID;  
уровни изоляции транзакций;  
декларативное управление транзакциями в Spring Tx;  
propagation транзакций в Spring Tx;  
императивное управление транзакциями в Spring Tx.

5 "Белая магия"  
Spring Data:  
Spring Data JPA

**Цели занятия:**

использовать абстракции Spring Data для реализации собственных репозиториев в общем виде;  
использовать "белую магию" Spring Data JPA для создания репозиториев для JPA сущностей.

**Краткое содержание:**

проекты Spring Data;  
репозитории на Spring Data JPA;  
кастомизация (@Query, custom repositories).

**Домашние задания**

1 Библиотеку на Spring Data JPA

Цель: Цель: максимально просто писать слой репозиториев с применением современных подходов

Результат: приложение со слоем репозиториев на Spring Data JPA

Домашнее задание выполняется переписыванием предыдущего на JPA.

Требования:

1. Переписать все репозитории по работе с книгами на Spring Data JPA репозитории.
2. Используйте spring-boot-starter-data-jpa.
3. Кастомные методы репозиториев (или с хитрым @Query) покрыть тестами, используя H2.
4. @Transactional рекомендуется ставить на методы сервисов, а не репозиториев.

Это домашнее задание будет использоваться в качестве основы для других ДЗ

Данная работа не засчитывает предыдущую!

---

## 6 SQL и NoSQL базы данных

### Цели занятия:

использовать особенности нереляционных (NoSQL) БД;  
выбирать NoSQL БД для решения задач.

### Краткое содержание:

принципы на которых строятся нереляционные  
(NoSQL) базы данных;  
анализ некоторых из них;  
запросы к ним.

---

## 7 Spring Data для подключения к нереляционным БД

### Цели занятия:

разрабатывать репозитории для хранения данных в  
NoSQL БД;  
использовать другие проекты Spring Data.

### Краткое содержание:

Spring Data Key-Value для подключения к Key-Value  
базам данных (и даже к SynchronizedMap);  
MondoDB;  
MongoDB с помощью Spring Data MongoDB.

### Домашние задания

- 1 Использовать MongoDB и spring-data для хранения информации о книгах

Цель: Цель: После выполнения ДЗ вы сможете использовать Spring Data MongoDB и саму MongoDB для разработки приложений с хранением данных в нереляционной БД.  
Результат: Приложение с использованием MongoDB

Задание может выполняться на основе предыдущего, а может быть выполнено самостоятельно

Требования:

1. Использовать Spring Data MongoDB репозитории, а если не хватает

функциональности, то и \*Operations

2. Тесты можно реализовать с помощью Flapdoodle Embedded MongoDB

3. Hibernate, равно, как и JPA, и spring-boot-starter-data-jpa не должно остаться в зависимостях, если ДЗ выполняется на основе предыдущего.

4. Как хранить книги, авторов, жанры и комментарии решать Вам. Но перенесённая с реляционной базы структура не всегда будет подходить для MongoDB.

Данное задание НЕ засчитывает предыдущие!

Это задание может использоваться в дальнейшем, а может не использоваться - на Ваше дальнейшее усмотрение

---

8 **Разбор домашних заданий, QnA**

**Цели занятия:**

писать код с учётом Best Practices;  
не допускать частых ошибок;  
получить ответы на вопросы.

**Краткое содержание:**

типичные ошибки при выполнении ДЗ;  
наставники ответят на ваши вопросы.

### 1 Введение в Spring MVC, Spring MVC на Spring Boot

#### Цели занятия:

различать архитектуры MVC и Spring MVC;  
создавать простые классические Web-приложения на основе Spring MVC и Spring Boot;  
создавать REST-сервисы на основе Spring MVC и Spring Boot.

#### Краткое содержание:

архитектуры MVC и Spring MVC;  
Web-слой на Spring MVC;  
простой REST-сервис (без View).

---



## 2 Spring MVC View

### Цели занятия:

разрабатывать классические Web-приложения на Spring MVC;  
разрабатывать слой View на Thymeleaf.

### Краткое содержание:

подробнее о View в Spring MVC;  
Thymeleaf и его преимущество по сравнению с JSP;  
возможности Thymeleaf в процессе написания полноценного приложения с View.

### Домашние задания

- 1 CRUD приложение с Web UI и хранением данных в БД

Цель: Цель: разрабатывать полноценные классические Web-приложения

Результат: Web-приложение полностью на стеке Spring

Необходимо:

1. Создать приложение с хранением сущностей в БД (можно взять библиотеку и DAO/репозитории из прошлых занятий)
2. Использовать классический View на Thymeleaf, classic Controllers.
3. Для книг (главной сущности) на UI должны быть доступные все CRUD операции. CRUD остальных сущностей - по желанию/необходимости.
4. Локализацию делать НЕ нужно - она строго опциональна.

Данное задание НЕ засчитывает предыдущие!

Это домашнее задание частично будет использоваться в дальнейшем

## 3 Современные приложения на Spring MVC

### Цели занятия:

создавать сложные классические приложения с использованием Spring Web Flow;

создавать современные приложения, как основанные на AJAX архитектуре и jQuery, так и SPA-приложения на Vue, React и Angular.

### **Краткое содержание:**

Spring Web Flow, как высокоуровневую настройку над Spring MVC;  
приложения на jQuery с использованием AJAX;  
современные приложения на Vue;  
варианты написания приложений с использованием React и Angular.

### **Домашние задания**

- 1 Переписать приложение с использованием AJAX и REST-контроллеров

Цель: Цель: использовать Spring MVC для разработки современных AJAX/SPA приложений с помощью Spring MVC  
Результат: современное приложение на стеке Spring

Домашнее задание выполняется на основе предыдущего.

1. Переписать приложение с классических View на AJAX архитектуру и REST-контроллеры.
2. Минимум: получение одной сущности и отображение её на странице с помощью XMLHttpRequest, fetch api или jQuery
3. Опционально максимум: полноценное SPA приложение на React/Vue/Angular, только REST-контроллеры.

В случае разработки SPA - рекомендуется вынести репозиторий с front-end. Используйте проху при разработке (настройки CORS не должно быть).

Данное задание, выполненное в виде SPA засчитывает ДЗ №9 (Занятие №15).  
Если Вы хотите засчитать, то обязательно пришлите ссылку в чат соответствующего предыдущего занятия.

Данное ДЗ будет использоваться в дальнейшем

## 4 Реактивное программирование

### Цели занятия:

применять принципы, на которых построено Reactive программирование;  
пользоваться библиотекой RxJava для построения реактивных приложений;  
применять различные Rx-операторы для необходимых ситуаций.

### Краткое содержание:

принципы построений реактивных систем;  
RxJava;  
различные операторы RxJava.

---

## 5 Reactive Spring Frameworks

### Цели занятия:

ориентироваться в реактивных фреймворках в стеке Spring;  
использовать Reactive-версию Spring Data репозиторийев.

### Краткое содержание:

различные реактивные API и фреймворки;  
реактивный стек Spring и принципы его работы для построения реактивных приложений;  
особенности работы с БД в реактивном стеке;  
Reactive Spring Data Repositories.

---

## 6 Spring WebFlux

### Цели занятия:

создавать с помощью Spring WebFlux современные реактивные Web-приложения.

### Краткое содержание:

Reactive-стек Spring;  
Spring WebFlux;  
приложение на полностью реактивном стеке Spring с

использованием Spring WebFlux + Reactive Spring Data Repositories.

## Домашние задания

### 1 Использовать WebFlux

Цель: Цель: разрабатывать Responsive и Resilent приложения на реактивном стеке Spring с помощью Spring Web Flux и Reactive Spring Data Repositories

Результат: приложение на реактивном стеке Spring

1. За основу для выполнения работы можно взять ДЗ с Ajax + REST (классическое веб-приложение для этого лучше не использовать).
2. Но можно выбрать другую доменную модель (не библиотеку).
3. Необходимо использовать Reactive Spring Data Repositories.
4. В качестве БД лучше использовать MongoDB или Redis. Использовать PostgreSQL и экспериментальную R2DBC не рекомендуется.
5. RxJava vs Project Reactor - на Ваш вкус.
6. Вместо классического Spring MVC и embedded Web-сервера использовать WebFlux.
7. Опционально: реализовать на Functional Endpoints

Данное задание НЕ засчитывает предыдущие!

Рекомендации:

Старайтесь избавиться от лишних архитектурных слоёв. Самый простой вариант - весь flow в контроллере.

**1 Вебинар-  
собеседование****Цели занятия:**

проверить полученные на курсе знания;  
выявить сильные и слабые места;  
ознакомиться с форматом собеседования.

**Краткое содержание:**

На примере решения основными участниками небольших практических задач будут проверены знания слушателей в разрезе пройденного на курсе материала, а также знания в Java core. Кроме того, подразумевается участие "зрителей". Некоторые вопросы будут задаваться "в зал".

---

**2 Spring Security:  
Архитектура****Цели занятия:**

различать аутентификацию и авторизацию;  
различать задачи безопасности в Enterprise приложениях;  
разбираться в архитектуре Spring Security.

**Краткое содержание:**

задачи безопасности в Enterprise приложениях;  
система фильтров Spring Security;  
внутреннее устройство Spring Security и его основных классов.

---

**3 Spring Security:  
Механизмы  
аутентификации****Цели занятия:**

различать различные механизмы аутентификации;  
внедрять HTTP Basic аутентификацию для защиты Web-приложения;  
внедрять Form-based аутентификацию для защиты Web-приложения;  
использовать вторичные виды аутентификации (такие как Remember Me и Anonymous).

## Краткое содержание:

различные механизмы первичной аутентификации - HTTP Basic и Form-based и как их реализовывать с помощью Spring Security;  
вторичные механизмы аутентификации и как они поддерживаются Spring Security.

## Домашние задания

- 1 В CRUD Web-приложение добавить механизм аутентификации

Цель: Цель: защитить Web-приложение аутентификацией и простой авторизацией  
Результат: приложение с использованием Spring Security

Внимание! Задание выполняется на основе неактивного приложения Spring MVC!

1. Добавить в приложение новую сущность - пользователь. Не обязательно реализовывать методы по созданию пользователей - допустимо добавить пользователей только через БД-скрипты.
2. В существующее CRUD-приложение добавить механизм Form-based аутентификации.
3. UsersServices реализовать самостоятельно.
4. Авторизация на всех страницах - для всех аутентифицированных. Форма логина - доступна для всех.
5. Написать тесты контроллеров, которые проверяют, что все необходимые ресурсы действительно защищены.

Данное задание НЕ засчитывает предыдущие!

Данное ДЗ будет использоваться в дальнейшем.

---

## 4 Spring Security: Авторизация

### **Цели занятия:**

внедрять в приложение различные механизмы авторизации - на основе URL и методов сервисов, для глубокой защиты приложений.

### **Краткое содержание:**

классы Spring Security, участвующие в авторизации;  
авторизация на основе URL с помощью встроенного DSL;  
механизмы для авторизации на методах сервисов.

---

## 5 Spring Security: ACL

### Цели занятия:

внедрять в приложение безопасность на основе доменных сущностей - ACLs.

### Краткое содержание:

авторизация на основе доменных сущностей;  
структура ACL на примере реализации в БД;  
основные классы Spring Security, участвующие в авторизации на основе доменных сущностей;  
пример приложения с ACLs.

### Домашние задания

- 1 Ввести авторизацию на основе URL и/или доменных сущностей

Цель: Цель: научиться защищать приложение с помощью полноценной авторизации и разграничением прав доступа

Результат: полноценное приложение с безопасностью на основе Spring Security

Внимание! Задание выполняется на основе неактивного приложения Spring MVC!

1. Минимум: настроить в приложении авторизацию на уровне URL.
2. Максимум: настроить в приложении авторизацию на основе доменных сущностей и методов сервиса.

Рекомендации по выполнению:

1. Не рекомендуется выделять пользователей с разными правами в разные классы - т.е. просто один класс пользователя.
2. В случае авторизации на основе доменных сущностей и PostgreSQL не используйте GUID для сущностей.
3. Написать тесты контроллеров, которые проверяют, что все необходимые ресурсы действительно защищены

Данное задание НЕ засчитывает предыдущие!

---



## Цели занятия:

использовать Spring Batch для организации пакетной обработки данных в приложении;  
применять Spring Batch не только в больших проектах.

## Краткое содержание:

абстракции, предоставляемые Spring Batch;  
основной принцип абстракций;  
различные абстракции и их реализации в Spring Batch.

## Домашние задания

- 1 На основе Spring Batch разработать процедуру миграции данных из реляционного хранилища в NoSQL или наоборот

Цель: Цель: мигрировать данные с помощью Spring Batch

Результат: приложение для пакетных обработок данных на Spring Batch

1. Задание может быть выполнено в отдельном репозитории, с сущностями из ДЗ JPA и MongoDB.
2. Вы можете выбрать другую доменную модель
3. Не обязательно добавлять процесс миграции в веб-приложение. Приложение может быть оформлено в виде отдельной утилиты.
3. Используя Spring Batch, следите, чтобы связи сущностей сохранились.
4. Опционально: Сделать restart задачи с помощью Spring Shell.

Данное задание НЕ засчитывает предыдущие!

---

7 **Монолиты vs. Microservices Round 1, Messaging, Enterprise Integration Patterns (EIP)**

**Цели занятия:**

различать два подхода к разработке Enterprise-приложений - монолиты и микросервисы;  
видеть проблемы, возникающие при создании монолитов;  
использовать Best Practices при разработке монолитов.

**Краткое содержание:**

микросервисная и монолитная архитектуры;  
преимущества и недостатки обеих архитектур;  
различные виды интеграций;  
EIP (Enterprise Intergration Patterns) и Spring Integration.

---

8 **Spring Integration: Messages и Channels**

**Цели занятия:**

использовать различные многообразие сообщения для работы со Spring Integration;  
использовать различные семантики каналов там, где необходима нужная семантика интеграции;  
пользоваться встроенный DSL для настройки связей в Spring Integration;  
различать базовые Endpoints и Flow Components.

**Краткое содержание:**

паттерн Message;  
реализация Message в SPring Integration;  
классификация каналов с точки зрения семантики, буферизации и т.д.;  
реализация разных каналов в Spring Integration.

---

## 9 Spring Integration: Endpoints и Flow Components

### Цели занятия:

использовать Endpoints и Flow Components для разработки сложных Enterprise-приложений с почти любой интеграцией.

### Краткое содержание:

различные паттерны Endpoint и их реализации в Spring Integration;

на примере узнаем, как с помощью Flow Components управлять потоками интеграции.

### Домашние задания

- 1 Реализовать обработку доменной сущности через каналы Spring Integration

Цель: Участники смогут осуществлять "интеграцию" частей приложения с помощью EIP  
Результат: приложение с применением EIP на Spring Integration

1. Выберите другую доменную область и сущности. Пример: превращение гусеницы в бабочку.
2. Опишите/сконфигурируйте процесс (IntegrationFlow) с помощью инструментария предоставляемого Spring Integration.
3. Желательно использование MessagingGateway и subflow (при необходимости).

Задание выполняется в другом репозитории/в другой папке.

Данное задание НЕ засчитывает предыдущие!

---

## 10 Монолиты vs. Microservices (Round 2), Spring Boot Actuator - must have в микросервисах

### Цели занятия:

использовать Best Practices для разработки приложений на микросервисной архитектуре;  
использовать изобилие возможностей Spring Boot Actuator для создания production-grade приложений и микросервисов;  
применять HATEOAS подход для разработки REST-сервисов.

### Краткое содержание:

микросервисная архитектура и Best Practices;  
приложение Spring Boot Actuator;  
HATEOAS подход;  
Spring Data REST, как самый простой способ создания CRUD микросервисов.

### Домашние задания

#### 1 Использовать метрики, healthchecks и logfile

Цель: Цель: реализовать production-grade мониторинг и прозрачность в приложении  
Результат: приложение с применением Spring Boot Actuator

Данное задание выполняется на основе одного из реализованных Web-приложений

1. Подключить Spring Boot Actuator в приложение.
2. Включить метрики, healthchecks и logfile.
3. Реализовать свой собственный HealthCheck индикатор
4. UI для данных от Spring Boot Actuator реализовывать не нужно.
5. Опционально: переписать приложение на HATEOAS принципах с помощью Spring Data REST Repository

Данное задание НЕ засчитывает предыдущие!

---

## 11 REST-клиенты, SOAP, Spring WebServices и клиенты к ним

### Цели занятия:

писать REST-клиенты с различными дополнениями для построения полноценных микросервисных систем; писать на Spring WebServices SOAP-сервисы и клиенты к ним для разработки систем на SOA и SOAP.

### Краткое содержание:

основной класс для написания REST-клиентов - RestTemplate и напишем клиента с помощью него; Spring Cache и Spring Retry; SOA, SOAP, XML, XSD, WSDL; Code-first и Contract-first разработку; SOAP-сервис и SOAP-клиент к нему с помощью Spring Web Services.

---

## 12 Docker, оркестрация, облака, облачные хостинги

### Цели занятия:

использовать Docker для запуска приложений в повседневной жизни, собирать собственные образы Docker с помощью Dockerfile для запуска собственных приложений, использовать docker-compose и kubernetes для запуска сложных систем.

### Краткое содержание:

"Matrix from the Hell";  
уровень виртуализации Docker;  
приложение с помощью Docker;  
свой image с помощью Dockerfile;  
docker-compose и kubernetes.

### Домашние задания

#### 1 Обернуть приложение в docker-контейнер

Цель: Цель: деплоить приложение в современном DevOps-стеке

Результат: обёртка приложения в Docker

Внимание! Задание выполняется на основе любого сделанного Web-приложения

1. Обернуть приложение в docker-контейнер. Dockerfile принято располагать в корне репозитория. В image должна попадать JAR-приложения. Сборка в контейнере рекомендуется, но не обязательна.
2. БД в собственный контейнер оборачивать не нужно (если только Вы не используете кастомные плагины)
3. Настроить связь между контейнерами, с помощью docker-compose
4. Опционально: сделать это в локальном кубе.
5. Приложение желательно реализовать с помощью всех Best Practices Docker (логгирование в stdout и т.д.)

Данное задание НЕ засчитывает предыдущие!

---

13 **Облака, Mongo DB Atlas cluster, Spring Cloud**

**Цели занятия:**

разобраться с облаками, их типами, конкретными реализациями;  
создать бесплатный кластер Mongo DB Atlas;  
ознакомиться с возможностями Spring для интеграции с облаками.

**Краткое содержание:**

внутри облаков и вокруг них.

---

14 **Spring Cloud Config, Spring Cloud Bus, Spring Cloud Service Discovery с Eureka, Ribbon и Feign**

**Цели занятия:**

организовать взаимодействие микросервисов с Config Server с помощью Spring Cloud;  
изучить возможности обновления данных микросервисов "на лету" с помощью Spring Cloud Bus;  
научиться строить огромные системы на Cloud Service Discovery с использованием Eureka, Ribbon и Feign.

**Краткое содержание:**

взаимодействие участников микросервисной архитектуры в облаках с помощью Eureka;  
конфигурирование сервиса с помощью Spring Cloud Config.

---

15 **Zuul, Hystrix  
Circuit Breaker,  
Sleuth, Zipkin,  
Hystrix  
Dashboard,  
Secure  
Configuration  
Properties**

**Цели занятия:**

строить огромные системы с помощью Zuul, Hystrix, Sleuth, Zipkin.

**Краткое содержание:**

множество технологий для построения огромных систем;  
Hystrix как решение для изоляции внешних вызовов;  
логистика и мониторинг в Spring Cloud.

**Домашние задания**

**1** Обернуть внешние вызовы в Hystrix

Цель: Цель: сделать внешние вызовы приложения устойчивыми к ошибкам

Результат: приложение с изолированными с помощью Hystrix внешними вызовами

1. Обернуть все внешние вызовы в Hystrix, Hystrix Javanica.
  2. Возможно использование Resilient4j
  3. Возможно использование Feign Client
- Опционально: Поднять Turbine Dashboard для мониторинга.

Данное задание НЕ засчитывает предыдущие!

---

16 **Обзор  
дополнительных  
технологий  
Spring, быстрая  
разработка  
приложений**

**Цели занятия:**

быстро создавать современные приложения со всеми необходимыми примочками.

**Краткое содержание:**

принципы Rapid Application Development, Convention over Configuration;  
история создания, развитие и необычный мир Spring Roo;  
молодой конкурент Spring Roo - JHipster;  
плюсы и минусы рассматриваемых технологий.



## 1 Выбор темы и организация проектной работы

### Цели занятия:

выбрать и обсудить тему проектной работы, спланировать работу над проектом, ознакомиться с регламентом работы над проектом

### Краткое содержание:

правила работы над проектом и специфика проведения итоговой защиты;  
требования к результату проекта и итоговой документации.

### Домашние задания

#### 1 Проектная работа

Цель: Цель: реализовать собственный проект с применением Spring

Проект должен быть сделан на основе Spring Boot, включать работу с DB с использованием Spring Data репозитория и/или Spring JDBC.

Проект должен иметь UI построенный на современных принципах разработки Web-приложений (AJAX и/или SPA).

Приложение должно содержать механизмы аутентификации и авторизации с использованием Spring Security

Асинхронные части могут быть реализованы с помощью Spring Integration.

Пектные обработки, утилиты поддержки должны быть реализованы с помощью Spring Batch + Spring Shell.

Проект должен быть cloud-ready.

---

**2**    **Консультация  
по проектам и  
домашним  
заданиям**

**Цели занятия:**

получить ответы на вопросы по проекту, ДЗ и по курсу

**Краткое содержание:**

вопросы по улучшению и оптимизации работы над проектом;  
затруднения при выполнении ДЗ;  
вопросы по программе.

---

**3**    **Защита  
проектных  
работ №1**

**Цели занятия:**

защитить проект и получить рекомендации экспертов

**Краткое содержание:**

презентация проектов перед комиссией;  
вопросы и комментарии по проектам.

---

**4**    **Защита  
проектных  
работ №2**

**Цели занятия:**

защитить проект и получить рекомендации экспертов.

**Краткое содержание:**

презентация проектов перед комиссией;  
вопросы и комментарии по проектам.