

# Разработчик на Spring Framework

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

Длительность курса: 173 академических часа

## 1 Введение

### 1 Введение в Spring Framework

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.

Домашние задания

#### 1 Приложение по проведению тестирования студентов (только вывод вопросов)

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 задание:

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.

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", прикрепленные к материалам занятия).
2. Контекст описывается XML-файлом.
  3. Все зависимости должны быть настроены в IoC 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**

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.

### 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 **AOP, Spring AOP**

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

---

### 4 **"Чёрная магия" Spring Boot**

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.

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 Продвинутая конфигурация Spring-приложений

использовать Best Practices для конфигурирования Spring-приложений;  
максимально эффективно использовать аннотации конфигураций;  
писать приложения с использованием 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.

---

**6      Разбор домашних заданий, QnA**

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

### 1 DAO на Spring JDBC

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.

#### 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 обязателен, 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. Покрыть тестами, насколько это возможно.

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

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

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

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.

#### 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**

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

---

5 **"Белая магия" Spring Data: Spring Data JPA**

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

Домашние задания

1 Библиотеку на Spring Data JPA

Цель: Цель: максимально просто писать слой репозиторий с применением современных подходов  
Результат: приложение со слоем репозиторий 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 репозиторий (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**

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

---

7 **Spring Data для подключения к нереляционным БД**

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

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**

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



- |       |   |  |
|-------|---|--|
| 1     | <b>Введение в Spring MVC, Spring MVC на Spring Boot</b> | различать архитектуры MVC и Spring MVC;<br>создавать простые классические Web-приложения на основе Spring MVC и Spring Boot;<br>создавать REST-сервисы на основе Spring MVC и Spring Boot.   |
| <hr/> |   |  |
| 2     | <b>Spring MVC View</b>                                  | <p>разрабатывать классические Web-приложения на Spring MVC;<br/>разрабатывать слой View на Thymeleaf.</p> <p>Домашние задания</p> <p>1      CRUD приложение с Web UI и хранением данных в БД</p> <p>Цель: Цель: разрабатывать полноценные классические Web-приложения<br/>Результат: Web-приложение полностью на стеке Spring</p> <p>Необходимо:</p> <ol style="list-style-type: none"><li>1. Создать приложение с хранением сущностей в БД (можно взять библиотеку и DAO/репозитории из прошлых занятий)</li><li>2. Использовать классический View на Thymeleaf, classic Controllers.</li><li>3. Для книг (главной сущности) на UI должны быть доступные все CRUD операции. CRUD остальных сущностей - по желанию/необходимости.</li><li>4. Локализацию делать НЕ нужно - она строго опциональна.</li></ol> <p>Данное задание НЕ засчитывает предыдущие!</p> <p>Это домашнее задание частично будет использоваться в дальнейшем</p> |
| <hr/> |   |  |
| 3     | <b>Современные приложения на Spring MVC</b>             | <p>создавать сложные классические приложения с использованием Spring Web Flow;<br/>создавать современные приложения, как основанные на AJAX архитектуре и jQuery, так и SPA-приложения на Vue, React и Angular.</p> <p>Домашние задания</p> <p>1      Переписать приложение с использованием AJAX и REST-контроллеров</p> <p>Цель: Цель: использовать Spring MVC для разработки современных AJAX/SPA приложений с помощью Spring MVC<br/>Результат: современное приложение на стеке Spring</p> <p>Домашнее задание выполняется на основе предыдущего.</p> <ol style="list-style-type: none"><li>1. Переписать приложение с классических View на AJAX архитектуру и REST-контроллеры.</li><li>2. Минимум: получение одной сущности и отображение её на странице с помощью XMLHttpRequest, fetch api или</li></ol>   |

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**

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

---

5     **Reactive Spring Frameworks**

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

---

6     **Spring WebFlux**

создавать с помощью Spring WebFlux современные реактивные Web-приложения.

Домашние задания

1     Использовать WebFlux

Цель: Цель: разрабатывать Responsive и Resilient 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     | <b>Spring Security: Архитектура</b>              | различать аутентификацию и авторизацию;<br>различать задачи безопасности в Enterprise приложениях;<br>разбираться в архитектуре Spring Security.   |
| <hr/> |  |  |
| 2     | <b>Spring Security: Механизмы аутентификации</b> | <p>различать различные механизмы аутентификации;<br/>внедрять HTTP Basic аутентификацию для защиты Web-приложения;<br/>внедрять Form-based аутентификацию для защиты Web-приложения;<br/>использовать вторичные виды аутентификации (такие как Remember Me и Anonymous).</p> <p>Домашние задания</p> <p>1 В CRUD Web-приложение добавить механизм аутентификации</p> <p>Цель: Цель: защитить Web-приложение аутентификацией и простой авторизацией<br/>Результат: приложение с использованием Spring Security</p> <p>Внимание! Задание выполняется на основе неактивного приложения Spring MVC!</p> <p>1. Добавить в приложение новую сущность - пользователь. Не обязательно реализовывать методы по созданию пользователей - допустимо добавить пользователей только через БД-скрипты.<br/>2. В существующее CRUD-приложение добавить механизм Form-based аутентификации.<br/>3. UsersServices реализовать самостоятельно.<br/>4. Авторизация на всех страницах - для всех аутентифицированных. Форма логина - доступна для всех.</p> <p>Данное задание НЕ засчитывает предыдущие!</p> <p>Данное ДЗ будет использоваться в дальнейшем.</p> |
| <hr/> |  |  |
| 3     | <b>Spring Security: Авторизация</b>              | внедрять в приложение различные механизмы авторизации - на основе URL и методов сервисов, для глубокой защиты приложений.  |
| <hr/> |  |  |
| 4     | <b>Spring Security: ACL</b>                      | <p>внедрять в приложение безопасность на основе доменных сущностей - ACLs.</p> <p>Домашние задания</p> <p>1 Ввести авторизацию на основе URL и/или доменных сущностей</p> <p>Цель: Цель: научиться защищать приложение с помощью полноценной авторизации и разграничением прав доступа</p>   |

Результат: полноценное приложение с безопасностью на основе Spring Security

Внимание! Задание выполняется на основе нереактивного 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.

Данное задание НЕ засчитывает предыдущие!

---

## 5 Spring Batch

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

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!

---

## 6 Монолиты vs. Microservices Round 1, Messaging, Enterprise Integration Patterns (EIP)

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

---

## 7 Spring Integration: Messages и Channels

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.

---

## 8 **Spring Integration: Endpoints и Flow Components**

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

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!

---

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

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.

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

10	<b>REST-клиенты, SOAP, Spring WebServices и клиенты к ним</b>	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.
11	<b>Docker, оркестрация, облака, облачные хостинги</b>	<p>использовать Docker для запуска приложений в повседневной жизни, собрать собственные образы Docker с помощью Dockerfile для запуска собственных приложений, использовать docker-compose и kubernetes для запуска сложных систем</p> <p>Домашние задания</p> <p>1    Обернуть приложение в docker-контейнер</p> <p>Цель: Цель: деплоить приложение в современном DevOps-стеке Результат: обёртка приложения в Docker</p> <p>Внимание! Задание выполняется на основе любого сделанного Web-приложения</p> <p>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.)</p> <p>Данное задание НЕ засчитывает предыдущие!</p>
12	<b>Облака, Mongo DB Atlas cluster, Spring Cloud</b>	разобраться с облаками, их типами, конкретными реализациями; создать бесплатный кластер Mongo DB Atlas; ознакомиться с возможностями Spring для интеграции с облаками.
13	<b>Spring Cloud Config, Spring Cloud Bus, Spring Cloud Service Discovery с Eureka, Ribbon и Feign</b>	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.
14	<b>Zuul, Hystrix</b>	строить огромные системы с помощью Zuul, Hystrix, Sleuth,

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

Zipkin.

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!

---

15

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

быстро создавать современные приложения со всеми необходимыми примочками.



1	<b>Выбор темы и организация проектной работы</b>	<p>выбрать и обсудить тему проектной работы, спланировать работу над проектом, ознакомиться с регламентом работы над проектом</p> <p>Домашние задания</p> <p>1 Проектная работа</p> <p>Цель: Цель: реализовать собственный проект с применением Spring</p> <p>Проект должен быть сделан на основе Spring Boot, включать работу с DB с использованием Spring Data репозиторий и/или Spring JDBC.</p> <p>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</p> <p>Асинхронные части могут быть реализованы с помощью Spring Integration.</p> <p>Пектные обработки, утилиты поддержки должны быть реализованы с помощью Spring Batch + Spring Shell.</p> <p>Проект должен быть cloud-ready.</p>
2	<b>Консультация по проектам и домашним заданиям</b>	<p>получить ответы на вопросы по проекту, ДЗ и по курсу</p>
3	<b>Защита проектных работ №1</b>	<p>защитить проект и получить рекомендации экспертов</p>
4	<b>Защита проектных работ №2</b>	<p>защитить проект и получить рекомендации экспертов.</p>