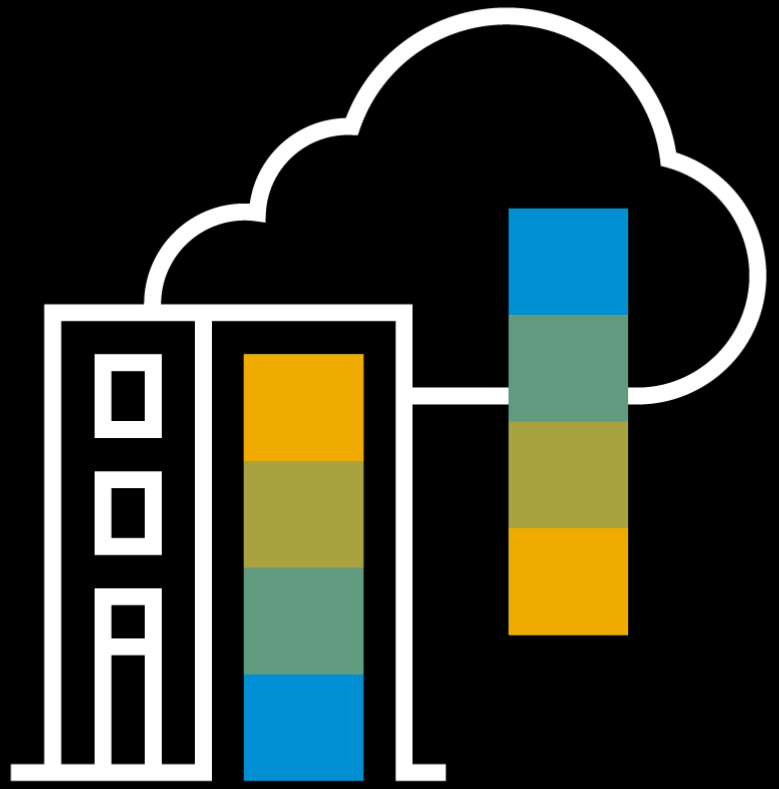
Spring Boot
Development Camp



Robin Herder, SAP
July 5, 2020

Agenda

Agenda Prerequisites

- Spring Framework - What is Spring?

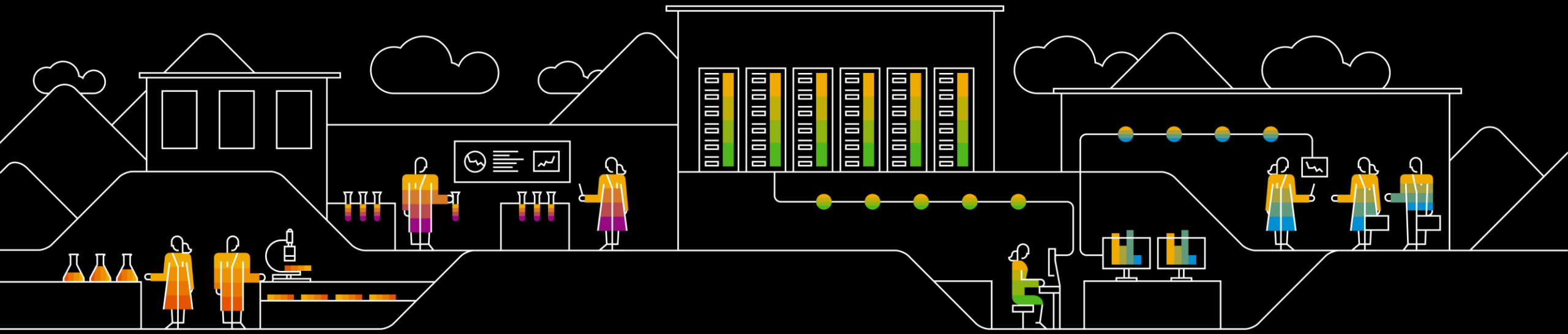
Agenda Spring Boot

- What is Spring Boot
- Web Starter – Rest API
- Datasource (H2) + Java Persistence API (JPA)
- Basic Procedure for Implementing a Spring Boot Application

Agenda Prospects

- Spring Boot on the SAP Cloud Platform – Cloud Foundry

Spring Framework 5



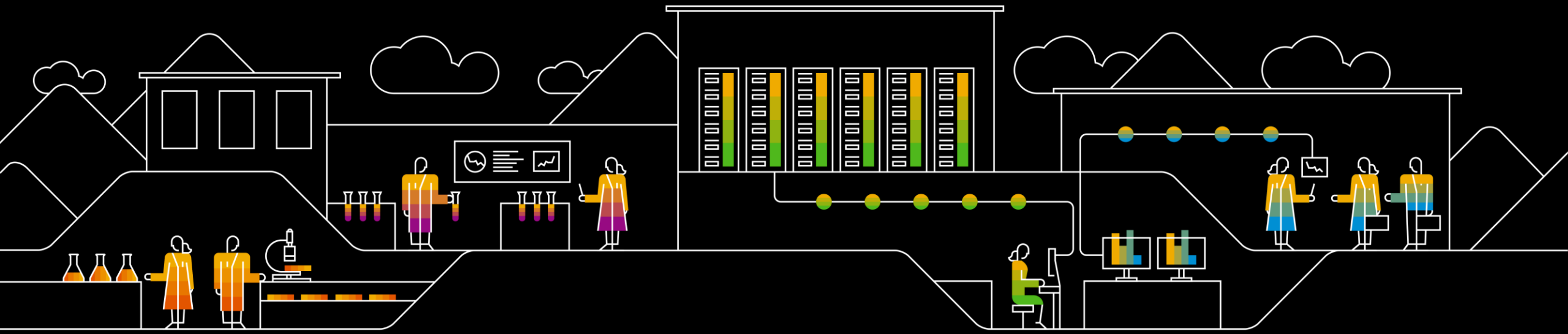
What is the Spring Framework

- The Spring Framework is a open Framework for the java development Platform
- The target of the Spring Framework is to Simplify the process of creating platform independent Enterprise Applications
- The Spring Framework consist of countless dependencies (ex. Jackson Databind) sometime bundled to Packages like:
  - Spring Data JPA
  - Spring Web MVC
  - …
- The Spring Context is the Heart of Springs own functionality
  - Bean Creation
  - Bean Lifecycle Management
  - Dependency Injection (implementation of IoC)

Features of Spring Framework 5

- Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP
- Testing: mock objects, TestContext framework, Spring MVC Test, WebTestClient
- Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML
- Spring MVC and Spring WebFlux web frameworks
- Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache
- Languages: Kotlin, Groovy, dynamic languages

Spring Boot 2



What is Spring Boot 2

- Over time the Spring Framework became increasingly complex and hard to configure. Spring Boot is the Solution
- Spring Boot is based on the Spring Framework with the Targets:
  - Shorten code lengths
  - Create standalone applications (ex. Comes with embedded Tomcat)
  - XML Configs are not required but allowed
  - Production ready features
  - Speed up Set Up via auto configuration (reduce amount of configuration needed)
- Spring Boots Projects are also Based on Maven
- Spring Boot Projects can be created manually or vie sites like https://start.spring.io/
- Spring Boot is not a competitor of the Spring Framework

Spring Boot – spring-boot-starter-web

- **spring-boot-starter-web** is a Spring Boot Dependency for Web Development
- The Dependency enables Spring Boot to
  - start the Tomcat server
  - serve static content
  - serve dynamic content via ‘controllers’
- It comes with:
  - The Spring Boot Starter
  - The Embedded Spring Boot Tomcat (Jetty & Undertow also supported)
  - The Spring Boot Validation
  - Jackson Databind
  - Spring Web
  - Spring Web MVC

Spring Boot – How to create a Controller

```java
@Controller
@RequestMapping (path = BookController.PATH)
public class BookController {

    private static final String PATH_PART_BOOK = "books";
    public static final String PATH = IRestController.REST_API_PATH + PATH_PART_BOOK;

    private final IBookService bookService;

    public BookController (IBookService bookService) { this.bookService = bookService; }

    @GetMapping (path = "/all", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<Book>> getAllBooks() { return new ResponseEntity<>(bookService.getAllBooks(), HttpStatus.OK); }

    @PutMapping (path = "/book", consumes = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity saveBook(@RequestBody Book book) {...}

    @PostMapping (path = "/book", consumes = MediaType.APPLICATION_JSON_VALUE, produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity updateBook(@RequestBody Book book) { return new ResponseEntity(bookService.updateBook(book), HttpStatus.OK); }

    @DeleteMapping (path = "/book/{isbn}")
    public ResponseEntity removeBook(@PathVariable String isbn) {...}

}
```

Spring Boot – How to create a Controller

```java
@GetMapping (path = "/book/{isbn}", produces = MediaType.APPLICATION_JSON_VALUE)
public ResponseEntity<Book> getBook(@PathVariable("isbn") String isbn) {
    return new ResponseEntity<>(bookService.getBook(isbn), HttpStatus.OK);
}
```

- Mapping: <HOST>/<CONTROLLER_BASE_MAPPING>/book/{isbn}
- **@PathVariable(“isbn”)** reads isbn from Url as Parameter for Method
- **@RequestBody** reads RequestBody (ex. Serialized JSON)
- ResponseEntity is the Response for the Request. It contains:
  - Serialized version of given Object (JSON)
  - HTML Response Code (ex. 200 – OK, 400 – Bad Request, 500 – Internal Server Error)

Spring Boot – spring-boot-starter-data-jpa

- **spring-boot-starter-data-jpa** is a Spring Boot Dependency for working with embedded (ex. H2) / standalone external Datasources (ex. SAP Hana)
- The Dependency enables Spring Boot to
  - Connect to external / embedded Datasources
  - Work with JPA / Hibernate
- It comes with:
  - The Spring Boot AOP Dependency
  - The Spring Boot JDBC Dependency
  - Spring Data JPA
  - Spring Aspects
  - Hibernate core
  - Jakarta

Spring Boot – Configure a H2 Datasource via application.properties / What is the application.properties

```
#Datasourc configuration
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

- The **application.properties** File exists in every Spring Boot Project. Its located in the resources
- It is used to configure Spring Boot Parameters like the **Datasource** or **Logging**
- Also can be used to create **custom parameters**, to later use them in code
- Can also be written in YAML as **application.yml**
- Can be profiled like **application-profile.properties**
- Setting up **H2**:
  - Add Dependency
  - Set Datasource Parameters
  - Set JPA Parameters

Spring Boot – JPA Entity

- **@Entity(‘<TABLE>‘)** – Marks class as Data Entity
- **@Id** – Marks Field as Primary Key
- **@Column(<PARAMS>)** – Marks Field as Column, declare column name …

- JPA Entities are used to map a Database Table as Java Object for (De-)Serialization

```
@Data
@ToString
@NoArgsConstructor
@EqualsAndHashCode
@Entity (name = "BOOK")
@JsonNaming (PropertyNamingStrategy.UpperCamelCaseStrategy.class)
public class Book {

    @Id
    @Column (name = "ISBN")
    private String isbn;

    @Column (name = "TITLE", nullable = false)
    private String title;

    @Column (name = "AUTHOR", nullable = false)
    private String author;

    @Column (name = "YEAR")
    private Date   year;

    @Column (name = "EDITOR")
    private String editor;

    @Column (name = "STOCK")
    private int stock = 0;

    @Column (name = "PRICE")
    private float price = 0.0f;

}
```

Spring Boot – JPA Repositories

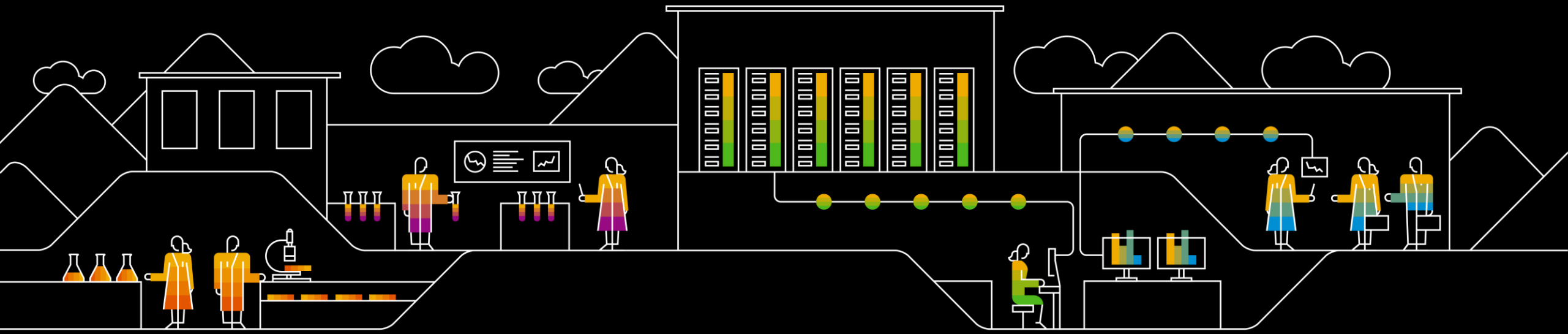
- The **JPA Repository** is used for accessing the DB
- Methods in the JPA Repo. are named:
  - **As you want** – Use **@Query('<Query>')** at the method and write a native or JPQL query
  - **After Convention** – the Identifier of the Method is the Query
    - **findBookByIsbn(String isbn)** – finds a book by its ISBN
    - **deleteBookByIsbn(Sring isbn)** – deletes a Book from the DB by its ISBN
- A JPA Repository contains a lot of **Standard functions**:
  - **findAll()** – Returns all Entities in Table
  - **save(Entity T)** – Persists Entities
  - …
- A JPA Repository is created by:
  - Creating a **Interface** (Best Practice identifier ends with Repository)
  - Extending the class **JpaRepository<Entity, IdField>**
  - At compile stage the Implementation of the Repo is generated

```
@Repository
public interface IBookRepository extends JpaRepository<Book, String> {

    @Query ("SELECT b FROM BOOK b ORDER BY b.title ASC")
    List<Book> getAll();

    @Query ("SELECT b FROM BOOK b WHERE b.author LIKE :AUTHOR")
    List<Book> findBooksByAuthorLike (@Param("AUTHOR") String author);

    Book findBookByIsbn (@Param("ISBN") String isbn);

    void deleteBookByIsbn (@Param("ISBN") String isbn);

    boolean existsBookByIsbn (@Param("ISBN") String isbn);
}
```

Spring Boot – Basic Procedure for Implementing a Spring Boot Application

- Think of the Features you want to Implement and note the Spring Boot Dependencies you need to Implement them
- Create the Project (ex. Via https://start.spring.io/)
- Create the Datamodel you need for your Application (ex. ERM) and create your Database
- Model your Entities & create Repositories for them (One Repository per Entity)
- Create your Services / Business Logic (Recommended use of Interfaces)
- Create Controllers

Spring Boot – SAP Cloud Platform CF



Spring Boot & The SAP Cloud Platform – Cloud Foundry

- Spring Boot 2 Applications natively Support Cloud Foundry
- A Spring Boot Application only needs a Manifest (manifest.yml) to run in the Cloud
- Spring Boot Applications are Deployed via the Cloud Foundry Command Line Interface (CF CLI)
- Alternatively Applications can be created as Multi Target Application (MTA)
- The MTA Configures
  - Everything the Manifest Configures
  - Create / Update Services in CF
- The Manifest Configures
  - RAM
  - Name
  - Route
  - …

Useful Links

- Maven in 5 Minutes - https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- What are Spring Beans - https://www.baeldung.com/spring-bean
- Building an Application with Spring Boot - https://spring.io/guides/gs/spring-boot/
- Introduction to Spring Data JPA - https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa
- Bootstrap a Simple Application - https://www.baeldung.com/spring-boot-start
- Spring Boot Overview & Learnings - https://spring.io/projects/spring-boot
- Baeldung Spring / Spring Boot Articles - https://www.baeldung.com/
- Stackoverflow - https://stackoverflow.com/

Thank you.

Contact information:

Robin Herder
Student of Business Information Technology 2018