



Spring Aufbau



Cegos Group

inspire
qualify
change



Inhalts- verzeichnis



Spring Basics mit Spring Boot



Spring Data



Context und Dependency Injection



Test und Anwendungsstart



Weitere Features






Spring Boot



Spring Basics mit Spring Boot



-  Setup mit Spring Boot
-  IDE Unterstützung
-  Ein erstes Beispiel



Setup mit Spring Boot



Build-Prozess

- Spring ist ein sehr mächtiges Framework
 - mit sehr vielen Abhängigkeiten zu weiteren Bibliotheken derCommunity
- Spring Boot stellt vordefinierte Maven POMs zur Verfügung
 - Parent POM
 - "Starter" POMs definieren einen kompletten Technologie-Stack
 - Web applications
 - JPA
 - ...



Beispiel: Ein Auszug einer Spring-POM

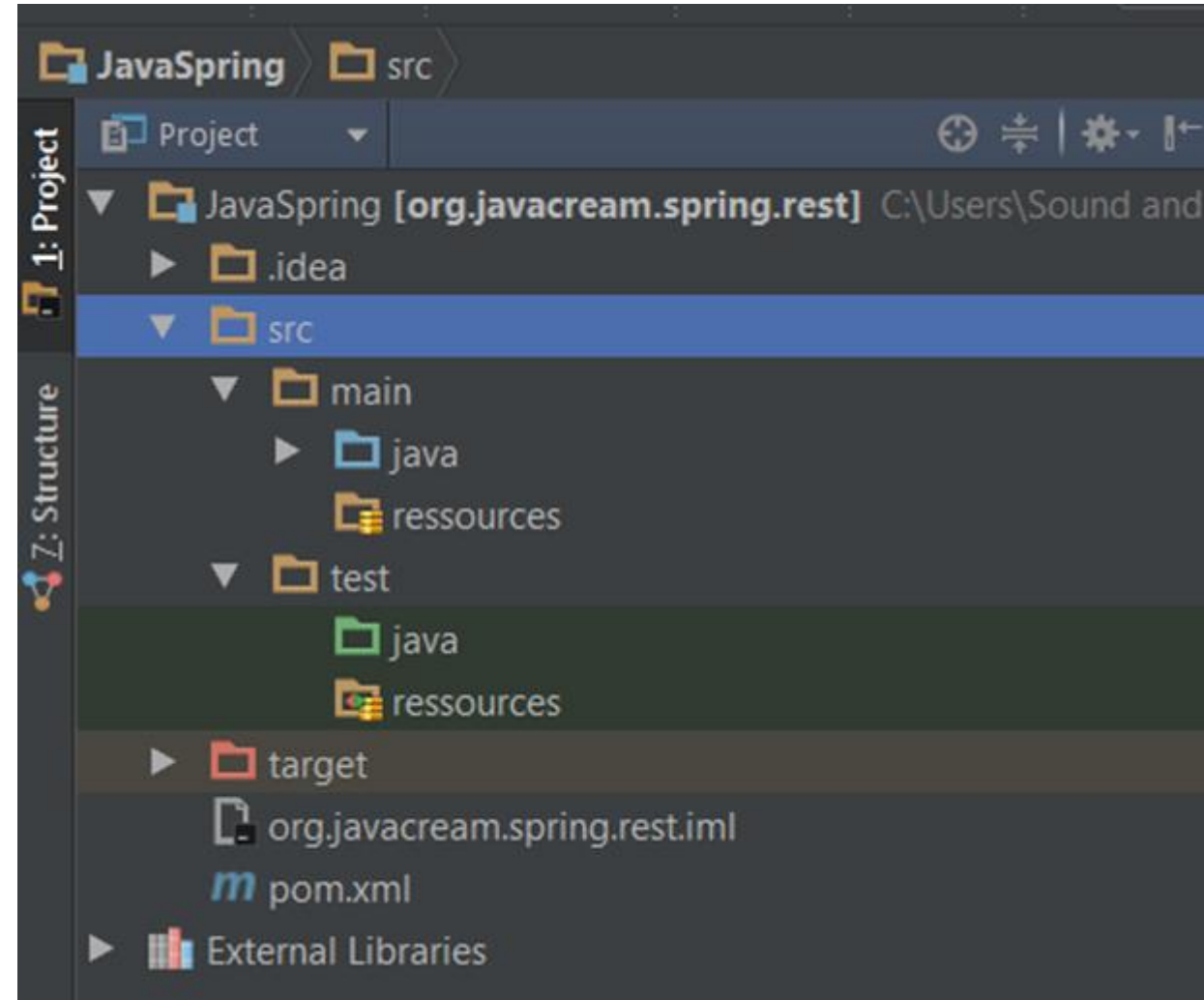
```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.javacream.training</groupId>
  <artifactId>org.javacream.training.spring.core.boot</artifactId>
d>
  <version>0.0.1-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.3.RELEASE</version>
    <relativePath />
  </parent>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter</artifactId>
    </dependency>
  </dependencies>
</project>
```



IDE Unterstützung

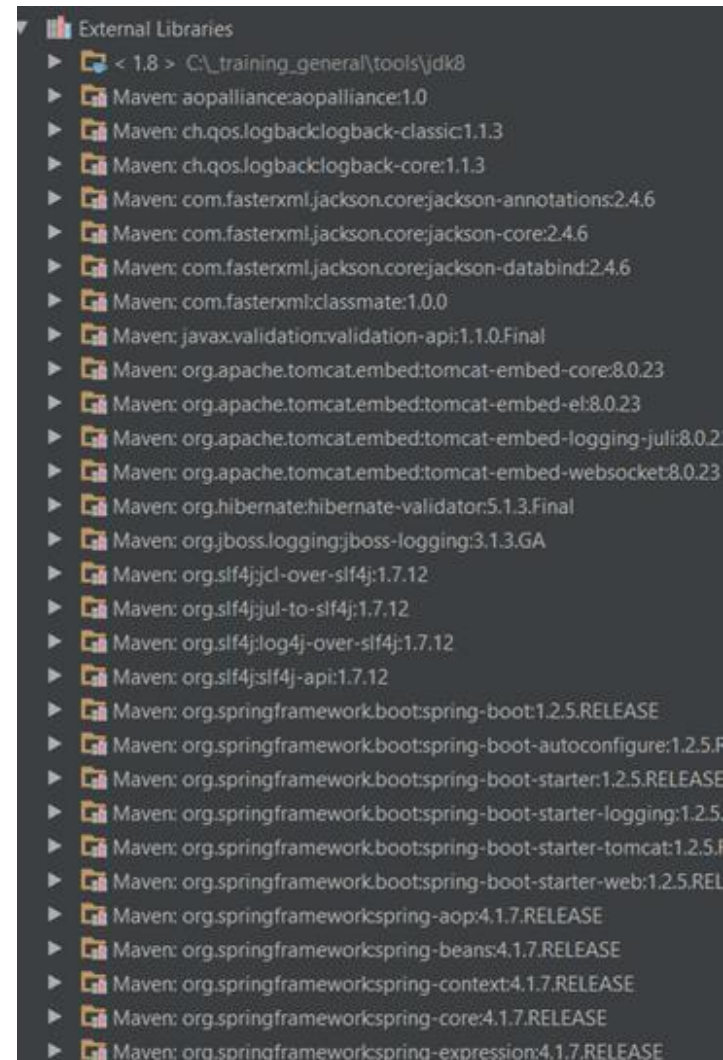


IntelliJ-Project





Maven- Dependencies in IntelliJ





Eclipse mit STS





Maven- Dependencies in Eclipse

▼ Maven Dependencies

- ▶ spring-boot-starter-data-jpa-2.0.6.RELEASE.jar - /home/rainer/.m2/repos
- ▶ spring-boot-starter-2.0.6.RELEASE.jar - /home/rainer/.m2/repository/org
- ▶ spring-boot-2.0.6.RELEASE.jar - /home/rainer/.m2/repository/org/spring
- ▶ spring-boot-autoconfigure-2.0.6.RELEASE.jar - /home/rainer/.m2/reposito
- ▶ spring-boot-starter-logging-2.0.6.RELEASE.jar - /home/rainer/.m2/reposit
- ▶ logback-classic-1.2.3.jar - /home/rainer/.m2/repository/ch/qos/logback/l
- ▶ logback-core-1.2.3.jar - /home/rainer/.m2/repository/ch/qos/logback/lo
- ▶ log4j-to-slf4j-2.10.0.jar - /home/rainer/.m2/repository/org/apache/loggir
- ▶ log4j-api-2.10.0.jar - /home/rainer/.m2/repository/org/apache/logging/lc
- ▶ jul-to-slf4j-1.7.25.jar - /home/rainer/.m2/repository/org/slf4j/jul-to-slf4j/
- ▶ javax.annotation-api-1.3.2.jar - /home/rainer/.m2/repository/javax/annol
- ▶ snakeyaml-1.19.jar - /home/rainer/.m2/repository/org/yaml/snakeyaml/
- ▶ spring-boot-starter-aop-2.0.6.RELEASE.jar - /home/rainer/.m2/repository
- ▶ spring-aop-5.0.10.RELEASE.jar - /home/rainer/.m2/repository/org/spring
- ▶ aspectjweaver-1.8.13.jar - /home/rainer/.m2/repository/org/aspectj/asp
- ▶ spring-boot-starter-jdbc-2.0.6.RELEASE.jar - /home/rainer/.m2/repository
- ▶ HikariCP-2.7.9.jar - /home/rainer/.m2/repository/com/zaxxer/HikariCP/2
- ▶ spring-jdbc-5.0.10.RELEASE.jar - /home/rainer/.m2/repository/org/spring
- ▶ javax.transaction-api-1.2.jar - /home/rainer/.m2/repository/javax/transa
- ▶ hibernate-core-5.2.17.Final.jar - /home/rainer/.m2/repository/org/hibern
- ▶ jboss-logging-3.3.2.Final.jar - /home/rainer/.m2/repository/org/jboss/log
- ▶ hibernate-jpa-2.1-api-1.0.2.Final.jar - /home/rainer/.m2/repository/org/hi
- ▶ javassist-3.22.0-GA.jar - /home/rainer/.m2/repository/org/javassist/javas
- ▶ antlr-2.7.7.jar - /home/rainer/.m2/repository/antlr/antlr/2.7.7



Ein erstes Beispiel



Eine Spring Boot Rest Application

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class RestApplication {

    @RequestMapping("/hello")
    public String doHello(){
        return "Hello!";
    }
    @RequestMapping("/exit")
    public void doExit(){
        System.exit(0);
    }

    public static void main(String[] args){
        SpringApplication.run(RestApplication.class, args);
    }
}
```



Start der Application

- Einfach die Starter-Klasse aufrufen
 - es ist tatsächlich so einfach
- Öffnen eines Browsers mit
 - <http://localhost:8080/hello>



Context und Dependency Injection



Überblick



Spring Core



Überblick



Context

- Der Context ist verantwortlich dafür, "relevante" Objekte zu erzeugen und zu verwalten
 - Die Lebensdauer ist abhängig vom angegebenen "Scope"
 - singleton/application
 - prototype/request
 - session
 - ...
 - In erster Näherung ist der Context eine (ziemlich) smarte Map
 -



Ein "Live" Context

```
mapBooksService  
Scope = singleton
```

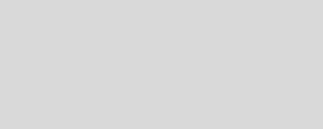
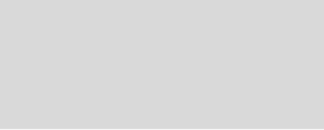
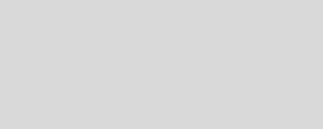
```
randomIsbnGenerator  
Scope = singleton
```

```
simpleStoreservice  
Scope = singleton
```

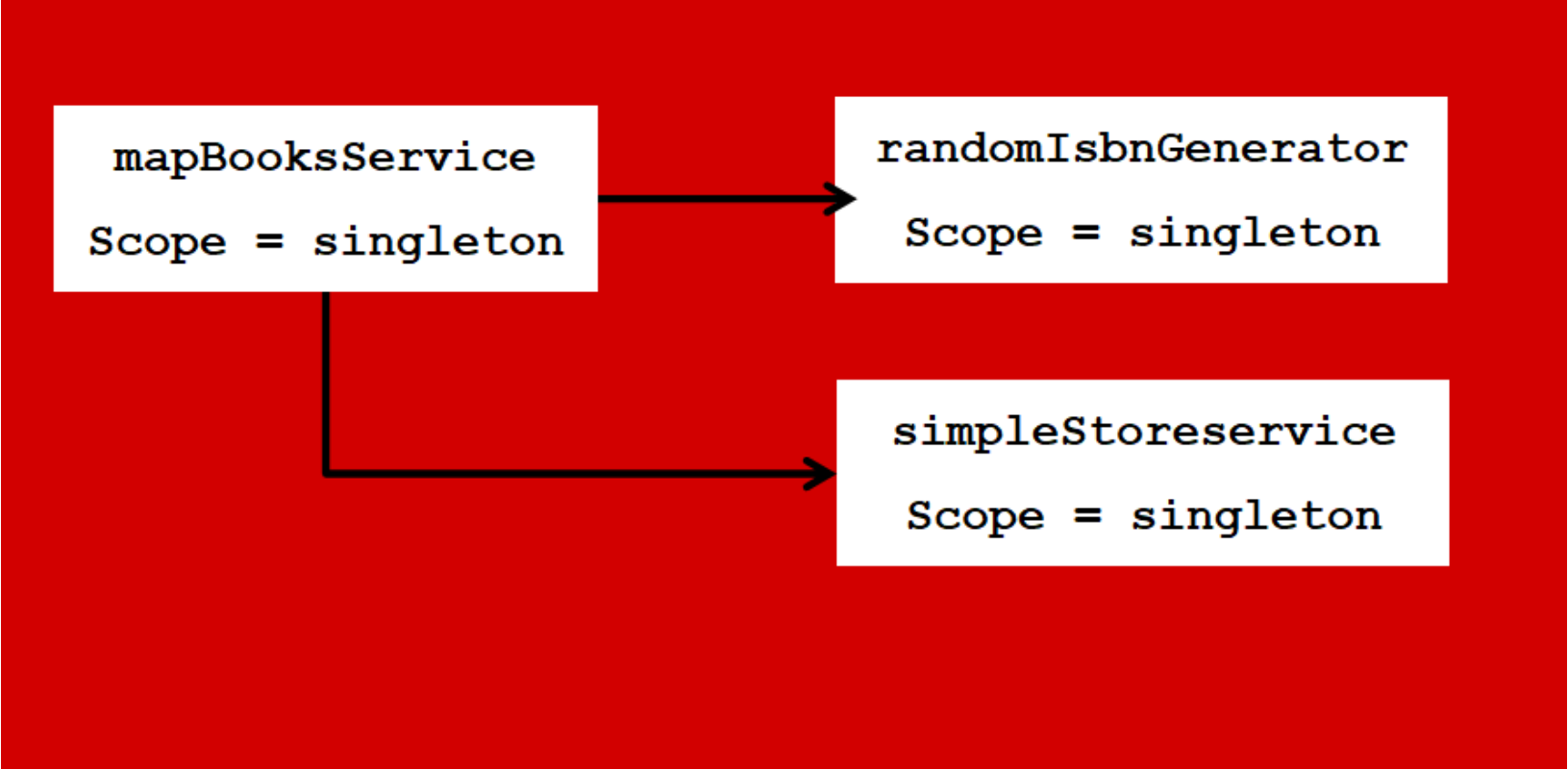


Dependency Injection

- Ein relevantes Objekt definiert neben dem Scope auch Abhängigkeiten auf andere Objekte
 - Im OOP-Modell ist dies eine Assoziation
- Der Context ist ebenfalls verantwortlich:
 - Dependencies zu identifizieren und
 - diese zu setzen



Ein "Live" Context nach Dependency Injection



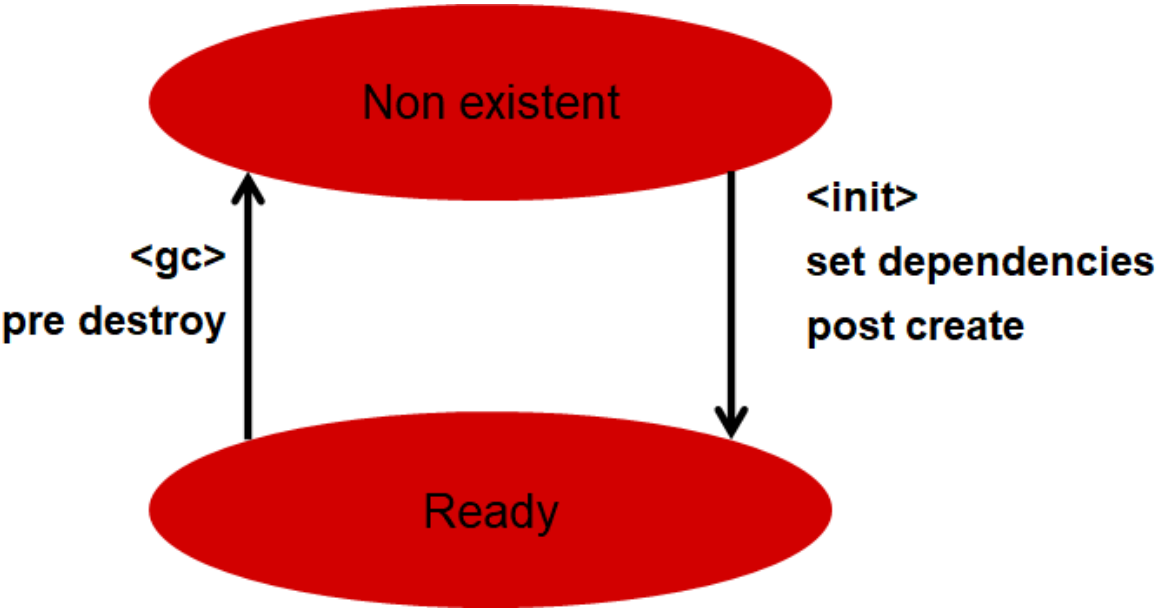


Lebenszyklus

- Every business object has a defined lifecycle
 - instantiation
 - dependency injection
 - post create
 - pre destroy
 - destroy (Garbage collection)



State diagram





Spring Core



The Spring Context

- Spring definiert Scope und Lifecycle mit
 - XML
 - Java Annotations
 - JavaConfig
 - eine Factory-Klasse
 - Eine Mischung aus allen VErfahren ist möglich



Context-Definition: Spring XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean class="org.javacream.store.business.SimpleStoreService" id="simpleStoreService">
        <property name="stock" value="42"></property>
    </bean>
    <bean class="org.javacream.keygeneration.business.RandomKeyGeneratorImpl"
          id="randomKeyGeneratorImpl" init-method="initThekeyGenerator" destroy-method="destroyThekeyGenerator">
        <property name="countryCode" value="-de"></property>
        <property name="prefix" value="ISBN:"></property>
    </bean>
    <bean class="org.javacream.books.warehouse.business.MapBooksService"
          id="mapBooksService">
        <property name="keyGenerator" ref="randomKeyGeneratorImpl"></property>
        <property name="storeService" ref="simpleStoreService"></property>
    </bean>
</beans>
```



Context-Definition: Spring Annotations

```
@Repository
public class MapBooksService implements BooksService {
    @Autowired
    @Qualifier("sequence")
    private KeyGenerator randomKeyGeneratorImpl;

    @Autowired
    private StoreService storeService;

    private Map<String, BookValue> books;

    {
        books = new HashMap<String, BookValue>();
    }
}
```



Context-Definition: Spring JavaConfig

```
@Configuration
public class BooksWarehouseConfig {

    @Bean public BooksService booksService() {
        MapBooksService mapBooksService = new MapBooksService();
        mapBooksService.setKeyGenerator(keyGenerator());
        mapBooksService.setStoreService(storeService());
        return mapBooksService;
    }

    @Bean public OrderService orderService() {
        OrderServiceImpl orderServiceImpl = new OrderServiceImpl();
        orderServiceImpl.setBooksService(booksService());
        orderServiceImpl.setStoreService(storeService());
        orderServiceImpl.setKeyGenerator(keyGenerator());
        return orderServiceImpl;
    }

    @Bean public StoreService storeService() {
        SimpleStoreService simpleStoreService = new SimpleStoreService();
        simpleStoreService.setStock(42);
        return simpleStoreService;
    }

    @Bean public KeyGenerator keyGenerator() {
        RandomKeyGeneratorImpl randomKeyGeneratorImpl = new RandomKeyGeneratorImpl();
        randomKeyGeneratorImpl.setPrefix("ISBN:");
        randomKeyGeneratorImpl.setCountryCode("-is");
        return randomKeyGeneratorImpl;
    }
}
```



Test und Anwendungsstart



Spring Unit Tests



Main



Spring Unit Tests



Unit Tests

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/books-service.xml")
public class BooksServiceSpringTest {

    @Autowired
    private BooksService booksService;

    @Test
    public void testSpring() {
        TestActor.doTest(booksService);
    }
}
```



Ablauf

- Durch die Angabe des Runners wird der angegebene Spring-Kontext geladen
- Das zu testende Objekt steht dann über normale Dependency Injection zur Verfügung
 - Hier mit Autowiring
- Weitere Features
 - Profiles definieren nur die Objekte, die dem angegebenen Profil entsprechen



Main



Main: Klassisch

```
public static void main(String[] args){  
    ClasspathXmlApplicationContext context =  
new  
    ClasspathXmlApplicationContext("/books-  
service.xml");  
    BooksService bs  
context.getBean(BooksService.class);  
}
```



Main: Spring Boot

```
public static void main(String[] args){  
  
    SpringApplication.run(BooksWarehouseConfig.class,  
                           args) ;  
}
```



Weitere Features



Spring AOP



Konfiguration



Utilities und weitere weitere
Bibliotheken



Spring AOP



Spring AOP

- Aspect Oriented Programmming führt "Cross Cutting Concerns" ein
- Beispiel:
 - "Jeder Aufruf einer Business-Funktion muss eine Audit-Meldung generieren"
 - Authentication
 - Transaction Management
 - Profiling
 - Tracing
- Spring stellt ein ausgefeiltes AOP Framework zur Verfügung
 - XML Konfiguration
 - Der `aop`-namespace
 - Annotations
 - `@AspectJ` und `@Around`



Konfiguration



Konfiguration

- Die Konfiguration von Spring Applications ist einfach
 - Lesen von Properties Files und System Properties
 - Bei Spring Boot wird automatisch die application.properties gelesen
 - auch das YAML-Format wird unterstützt
- Zum Zugriff auf die Properties wird die Spring Expression Language genutzt
 - in den meisten Fällen genügt eine einfache Expression
 - `${propertyKey}`
 - Ein komplexeres Beispiel
 - `${(bean.list[i5] + bean2.foo.goo) > 42}`
- Property-Values werden injected
 - value-Attribut im XML
 - @Value-Annotation



Utilities und weitere weitere Bibliotheken



Utilities

- Die Spring Distribution enthält umfangreiche Zusatzbibliotheken
 - Eingebunden jeweils über den entsprechenden Starter
- Beispiele
 - `JdbcTemplate` für direkten Datenbankzugriff
 - JPA-Integration
 - Das Web Framework Spring MVC
 - Klassische Web-Anwendungen
 - RESTful Web Services
 - Monitoring mit dem Spring Actuator
 - Developer Tools



Frameworks

- Spring Data
- Spring Security
- Spring Batch
- Spring Integration
- ...



Spring Boot



Was ist Spring Boot?



Dokumentation und Beispiele



Was ist Spring Boot?



Aufgaben

- Build-Werkzeug
 - Parent-POM mit allen unterstützten auch externen Bibliotheken
 - Konkrete Einbindung in ein Projekt mit Starter-POMs
- Laufzeitumgebung
 - Anwendungen können als einfacher Java-Prozess gestartet werden
 - Damit sehr einfach Container-fähig!
 - Die benötigte Infrastruktur wird mit integriert
 - Web Server
 - Bei Bedarf Embedded Database
 - ...
- Anwendungs-Konfiguration
 - Autoconfig zur Erstellung von Prototypen oder einfachen Anwendungen
 - Explizite Konfiguration über Datei oder zentralen Config-Server
- Vereinfachte Anwendungsentwicklung
 - Zusätzliche Annotationen
 - Mächtiges Test-Framework



Dokumentation und Beispiele

https://spring.io/projects/spring-boot

by Pivotal

PROJECTS

GUIDES

BLOG

TRAINING & CERTIFICATION

Projects

Spring Boot

Spring Framework

Spring Data >

Spring Cloud >

Spring Cloud Data Flow >

Spring Security >

Spring Session >

Spring Integration

Spring HATEOAS

Spring REST Docs

Spring Batch

Spring IO Platform

Spring AMQP

Spring for Android

Spring CredHub

Spring Flo

Spring for Apache Kafka

Spring LDAP

Spring Mobile

Spring Roo

Spring Boot 2.1.5

Overview

Learn

Samples

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration

You can also [join the Spring Boot community on Gitter!](#)

Talks and videos

- [It's a Kind of Magic: Under the Covers of Spring Boot](#)
- [What's New in Spring Boot 2.0](#)
- [Introducing Spring Boot 2.0 webinar](#)
- [Test Driven Development with Spring Boot](#)
- [From Zero to Hero with Spring Boot 2.0](#)

https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples

spring-projects / spring-boot

Watch 3,274 Star 38,021 Fork 24,799

Code Issues 380 Pull requests 41 Projects 0 Wiki Insights

Branch: master spring-boot / spring-boot-samples / Create new file Upload files Find file History

philwebb Update copyright header of changed files Latest commit 3a9ca5f 4 days ago

..		
spring-boot-sample-activemq	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-actuator-custo...	Update copyright header of changed files	4 days ago
spring-boot-sample-actuator-log4j2	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-actuator-noweb	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-actuator-ui	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-actuator	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-amqp	Update copyright header of changed files	2 months ago
spring-boot-sample-animated-ban...	Merge branch '1.5.x' into 2.0.x	2 months ago
spring-boot-sample-ant	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-aop	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-atmosphere	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-batch	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-cache	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-custom-layout	Merge branch '2.0.x' into 2.1.x	2 months ago
spring-boot-sample-data-cassandra	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-data-couchbase	Update copyright header of changed files	4 days ago
spring-boot-sample-data-elasticse...	Validate our own tests work with JUnit5 and the vintage engine	11 days ago
spring-boot-sample-data-jdbc	Update copyright header of changed files	4 days ago
spring-boot-sample-data-jpa	Remove reference to spring.datasource.jmx-enabled	6 days ago



Spring Data



Was ist Spring Data?



Dokumentation und Beispiele



Was ist Spring Data?




Aufgaben

- Ein Umbrella-Projekt für den Zugriff auf verschiedenste Backend-Systeme
 - Datenbanken
 - NoSQL-Produkte
 - ...
- Allerdings wird im Common-Projekt eine allgemein gültige Architektur festgelegt
 - Definition einer ID für Datensätze
 - Zugriff auf das Backend-System über ein Repository
 - CRUD-Operationen
 - Definition von Queries über Annotationen bzw. über Namenskonventionen



Dokumentation und Beispiele

<https://spring.io/projects/spring-data>




PROJECTS

GUIDES

BLOG

TRAINING & CERTIFICATION



Projects

Spring Boot

Spring Framework

Spring Data ▾

• Spring Data JDBC

• Spring Data JDBC Extensions

• Spring Data JPA

• Spring Data LDAP

• Spring Data MongoDB

• Spring Data Redis

• Spring Data R2DBC

• Spring Data REST

• Spring Data for Apache Cassandra

• Spring Data for Apache Geode

• Spring Data for Apache Solr

• Spring Data for Pivotal GemFire

• Spring Data Couchbase

• Spring Data Elasticsearch

Spring Data Lovelace-SR8

Overview

Learn

Spring Data's mission is to provide a familiar and consistent, Spring-based programming model for data access while still retaining the special traits of the underlying data store.

It makes it easy to use data access technologies, relational and non-relational databases, map-reduce frameworks, and cloud-based data services. This is an umbrella project which contains many subprojects that are specific to a given database. The projects are developed by working together with many of the companies and developers that are behind these exciting technologies.

Features

- Powerful repository and custom object-mapping abstractions
- Dynamic query derivation from repository method names
- Implementation domain base classes providing basic properties
- Support for transparent auditing (created, last changed)
- Possibility to integrate custom repository code
- Easy Spring integration via JavaConfig and custom XML namespaces
- Advanced integration with Spring MVC controllers
- Experimental support for cross-store persistence

Main modules

- **Spring Data Commons** - Core Spring concepts underpinning every Spring Data module.
- **Spring Data JDBC** - Spring Data repository support for JDBC.
- **Spring Data JDBC Ext** - Support for database specific extensions to standard JDBC including support for Oracle RAC fast connection failover, AQ JMS support and support for using advanced data types.

https://github.com/spring-projects/spring-data-examples

spring-projects / spring-data-examples

Watch 311Star 2,828Fork 2,127

CodeIssues 20Pull requests 9Projects 0WikiInsights

Spring Data Example Projects

420 commits24 branches0 releases26 contributorsApache-2.0

Branch: masterNew pull requestCreate new fileUpload filesFind FileClone or download

mp911de #502 - Upgrade to Spring Boot R2DBC 0.1.0.M1. Latest commit 7f9fbaa 4 days ago

bom	#443 - Show updated usage of Gradle with Maven BOMs.	23 days ago
cassandra	#491 - URL Cleanup.	2 months ago
couchbase	#491 - URL Cleanup.	2 months ago
elasticsearch	#491 - URL Cleanup.	2 months ago
jdbc	#491 - URL Cleanup.	2 months ago
jpa	#494 - Fix spelling mistake.	2 months ago
ldap	#491 - URL Cleanup.	2 months ago
map	#491 - URL Cleanup.	2 months ago
mongodb	#491 - URL Cleanup.	2 months ago
multi-store	#491 - URL Cleanup.	2 months ago
neo4j	#491 - URL Cleanup.	2 months ago
r2dbc	#502 - Upgrade to Spring Boot R2DBC 0.1.0.M1.	4 days ago
redis	#491 - URL Cleanup.	2 months ago
rest	#491 - URL Cleanup.	2 months ago
solr	#491 - URL Cleanup.	2 months ago
web	#491 - URL Cleanup.	2 months ago
.gitignore	Added IntelliJ artefacts to gitignore.	5 years ago