Spring Boot

Piya Lumyong

History

- 1999 J2EE 1.2

- 2001 J2EE 1.3

- 2003 J2EE 1.4

- 2004 Spring Framework 1.0

Dependency Injection

POJO oriented

AOP

- 2006 JavaEE 5

- 2013 Spring Boot

Defacto framework on Java

- Dependency Injection approach encourages writing testable code

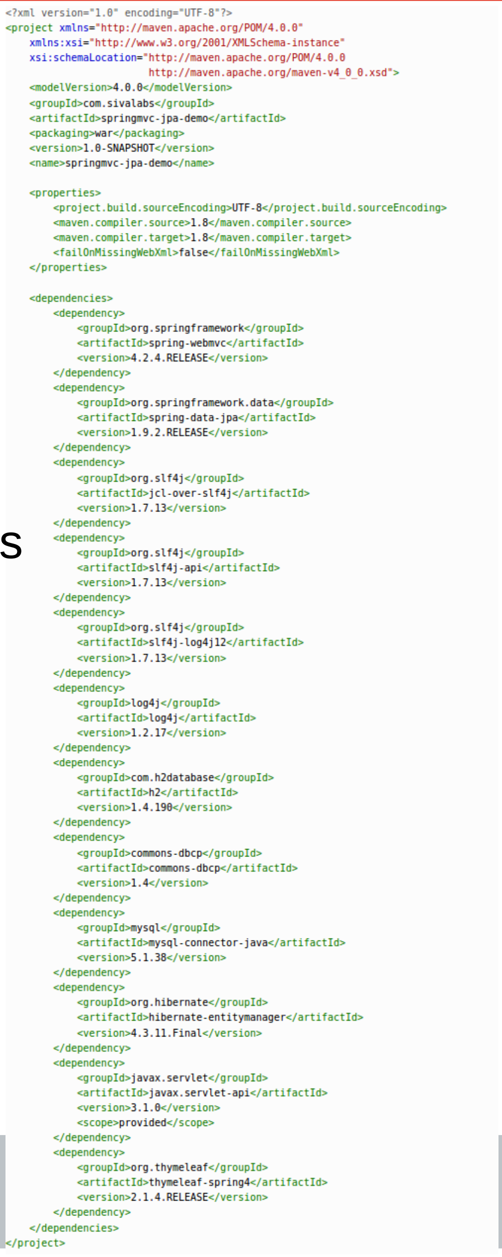
- Powerful database transaction management with AOP

- Spring simplifies integration with other Java frameworks

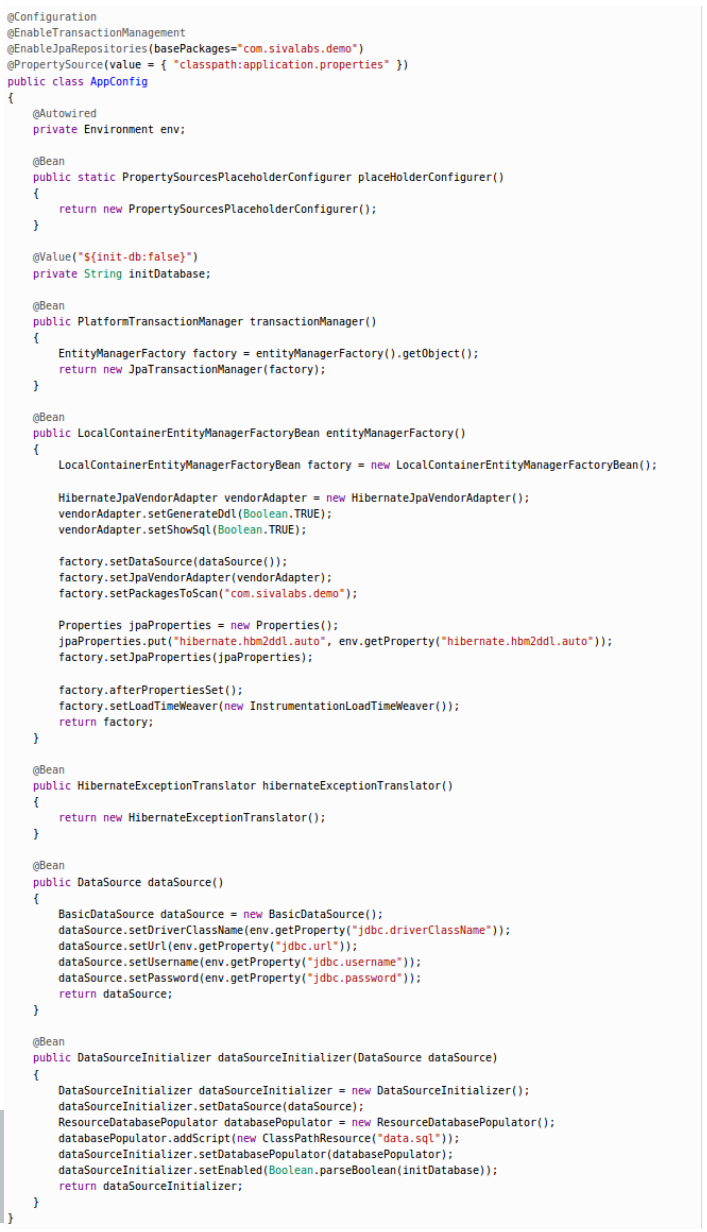
- Spring MVC

Before Boot

Dependencies



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                        http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.sivalabs</groupId>
    <artifactId>springmvc-jpa-demo</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>springmvc-jpa-demo</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <failOnMissingWebXml>false</failOnMissingWebXml>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>4.2.4.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.data</groupId>
            <artifactId>spring-data-jpa</artifactId>
            <version>1.9.2.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>jcl-over-slf4j</artifactId>
            <version>1.7.13</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.13</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.13</version>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <version>1.4.190</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.38</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-entitymanager</artifactId>
            <version>4.3.11.Final</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.thymeleaf</groupId>
            <artifactId>thymeleaf-spring4</artifactId>
            <version>2.1.4.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```



```java
@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(basePackages="com.sivalabs.demo")
@PropertySource(value = { "classpath:application.properties" })
public class AppConfig
{
    @Autowired
    private Environment env;

    @Bean
    public static PropertySourcesPlaceholderConfigurer placeHolderConfigurer()
    {
        return new PropertySourcesPlaceholderConfigurer();
    }

    @Value("${init-db:false}")
    private String initDatabase;

    @Bean
    public PlatformTransactionManager transactionManager()
    {
        EntityManagerFactory factory = entityManagerFactory().getObject();
        return new JpaTransactionManager(factory);
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory()
    {
        LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();

        HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
        vendorAdapter.setGenerateDdl(Boolean.TRUE);
        vendorAdapter.setShowSql(Boolean.TRUE);

        factory.setDataSource(dataSource());
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("com.sivalabs.demo");

        Properties jpaProperties = new Properties();
        jpaProperties.put("hibernate.hbm2ddl.auto", env.getProperty("hibernate.hbm2ddl.auto"));
        factory.setJpaProperties(jpaProperties);

        factory.afterPropertiesSet();
        factory.setLoadTimeWeaver(new InstrumentationLoadTimeWeaver());
        return factory;
    }

    @Bean
    public HibernateExceptionTranslator hibernateExceptionTranslator()
    {
        return new HibernateExceptionTranslator();
    }

    @Bean
    public DataSource dataSource()
    {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setDriverClassName(env.getProperty("jdbc.driverClassName"));
        dataSource.setUrl(env.getProperty("jdbc.url"));
        dataSource.setUsername(env.getProperty("jdbc.username"));
        dataSource.setPassword(env.getProperty("jdbc.password"));
        return dataSource;
    }

    @Bean
    public DataSourceInitializer dataSourceInitializer(DataSource dataSource)
    {
        DataSourceInitializer dataSourceInitializer = new DataSourceInitializer();
        dataSourceInitializer.setDataSource(dataSource);
        ResourceDatabasePopulator databasePopulator = new ResourceDatabasePopulator();
        databasePopulator.addScript(new ClassPathResource("data.sql"));
        dataSourceInitializer.setDatabasePopulator(databasePopulator);
        dataSourceInitializer.setEnabled(Boolean.parseBoolean(initDatabase));
        return dataSourceInitializer;
    }
}
```

Application Config

Ref. DZone

Before Boot

```java
public class SpringWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer
{

    @Override
    protected Class<?>[] getRootConfigClasses()
    {
        return new Class<?>[] { AppConfig.class};
    }

    @Override
    protected Class<?>[] getServletConfigClasses()
    {
        return new Class<?>[] { WebMvcConfig.class };
    }

    @Override
    protected String[] getServletMappings()
    {
        return new String[] { "/" };
    }

    @Override
    protected Filter[] getServletFilters() {
       return new Filter[]{ new OpenEntityManagerInViewFilter() };
    }
}
```

Servlet Initializer

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/test
jdbc.username=root
jdbc.password=admin
init-db=true
hibernate.dialect=org.hibernate.dialect.MySQLDialect
hibernate.show_sql=true
hibernate.hbm2ddl.auto=update
```

Application Properties

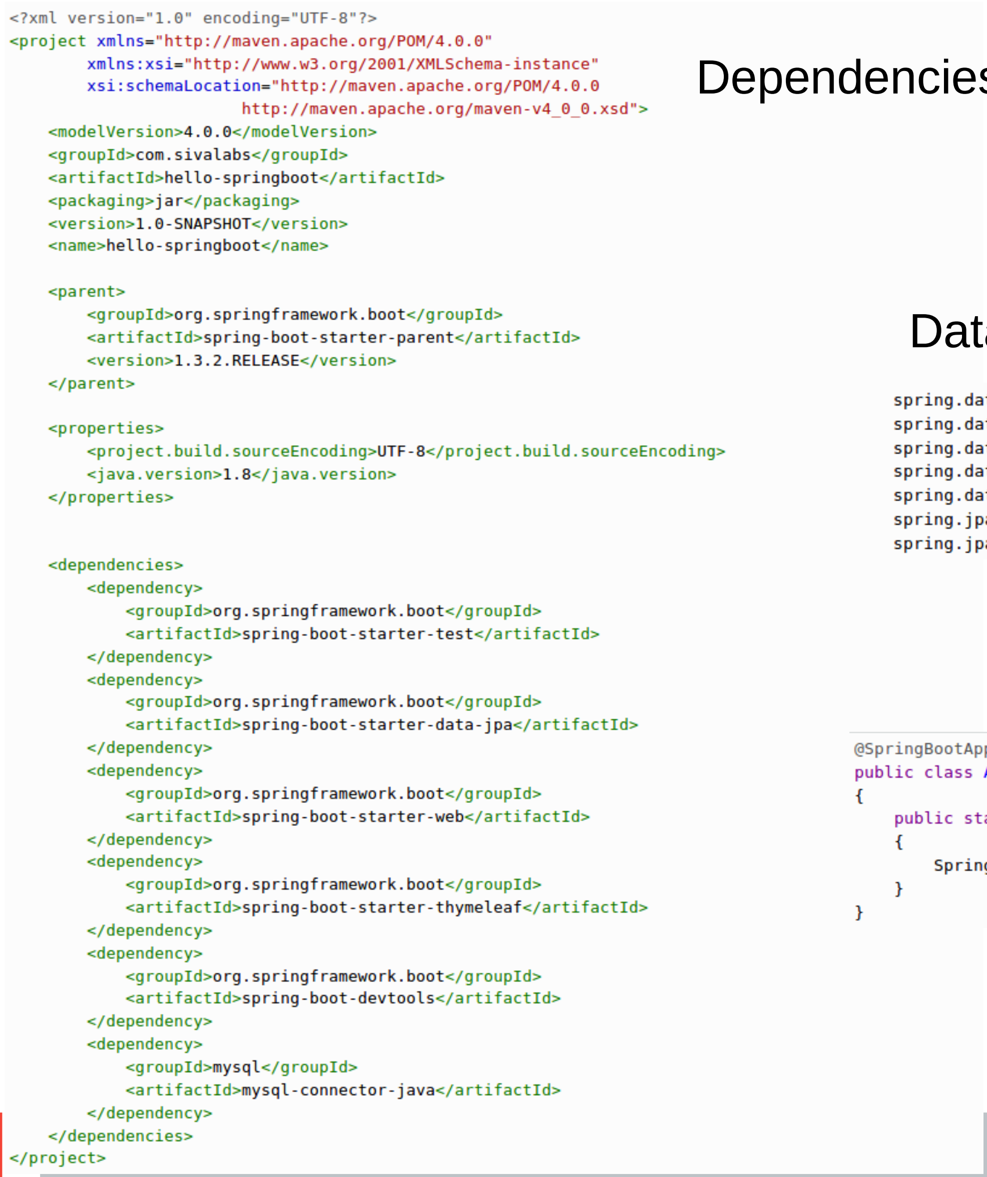
```
log4j.rootCategory=INFO, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p %t %c{2}:%L - %m%n

log4j.category.org.springframework=INFO
log4j.category.com.sivalabs=DEBUG
```

Log Config

Ref. DZone

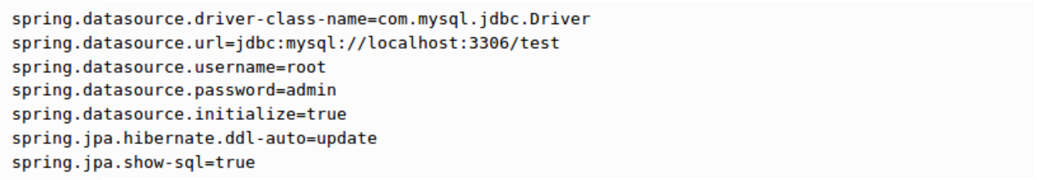
After Boot

Dependencies



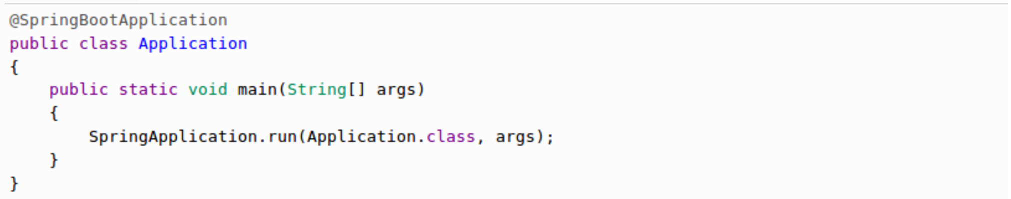
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                        http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.sivalabs</groupId>
    <artifactId>hello-springboot</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>hello-springboot</name>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.3.2.RELEASE</version>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
    </properties>


    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
    </dependencies>
</project>
```

Database, JPA, Logs and Application Properties



```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=admin
spring.datasource.initialize=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Startup Container Entry Point



```java
@SpringBootApplication
public class Application
{
    public static void main(String[] args)
    {
        SpringApplication.run(Application.class, args);
    }
}
```

Ref. DZone

Primary goals

- Faster getting started

- Out of the box

- Provided non-functional features that are common to large classes of projects

Embedded servers

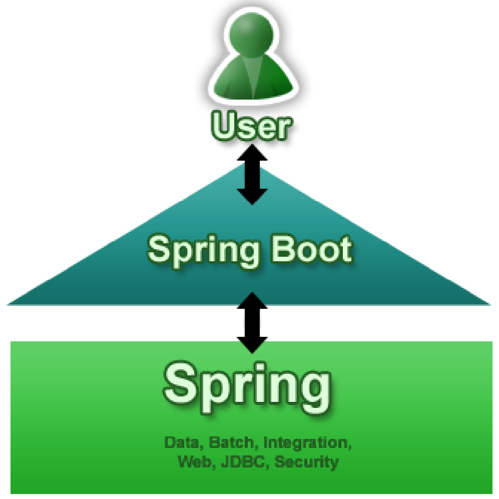
Security

Metrics

Health checks

Externalized configuration

etc.

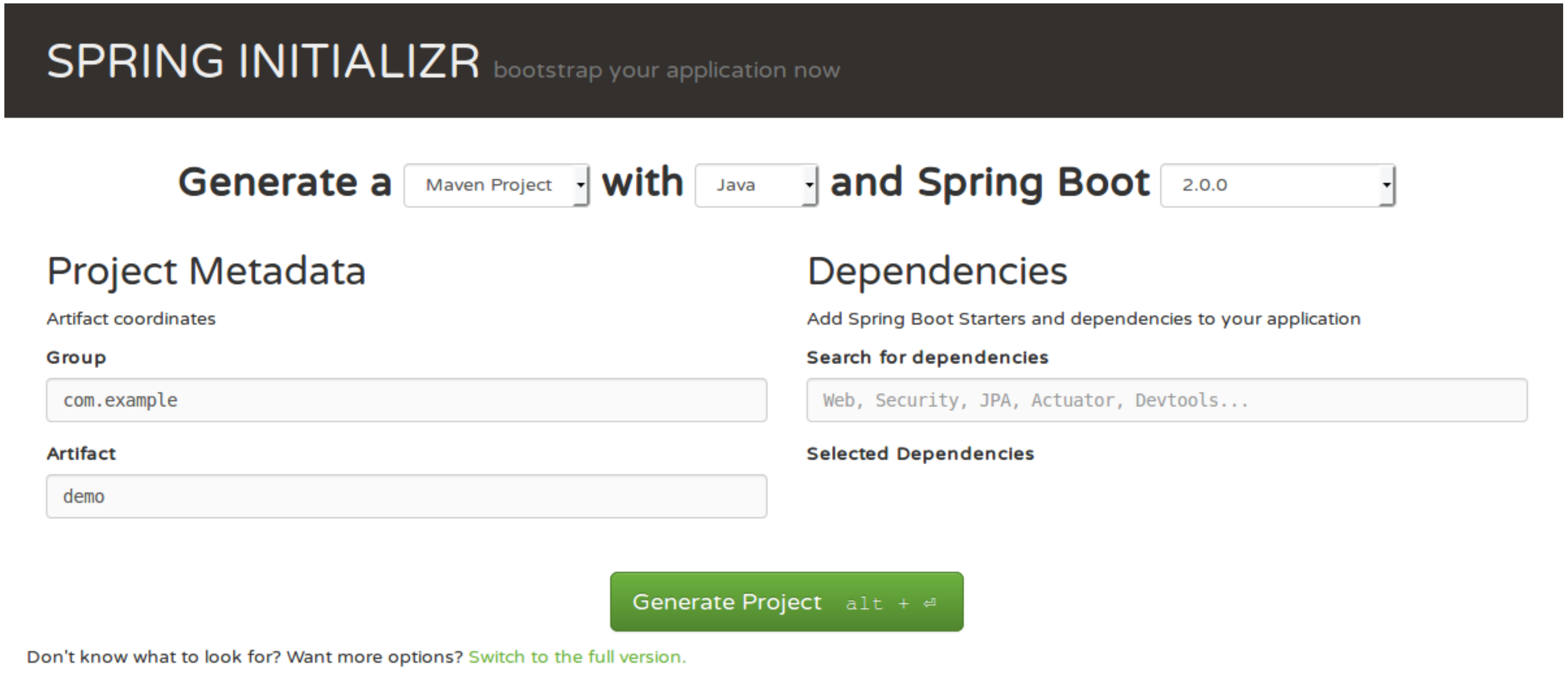


Key Feature

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

Spring Initializer

http://start.spring.io/


SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.0.0

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Don't know what to look for? Want more options? Switch to the full version.

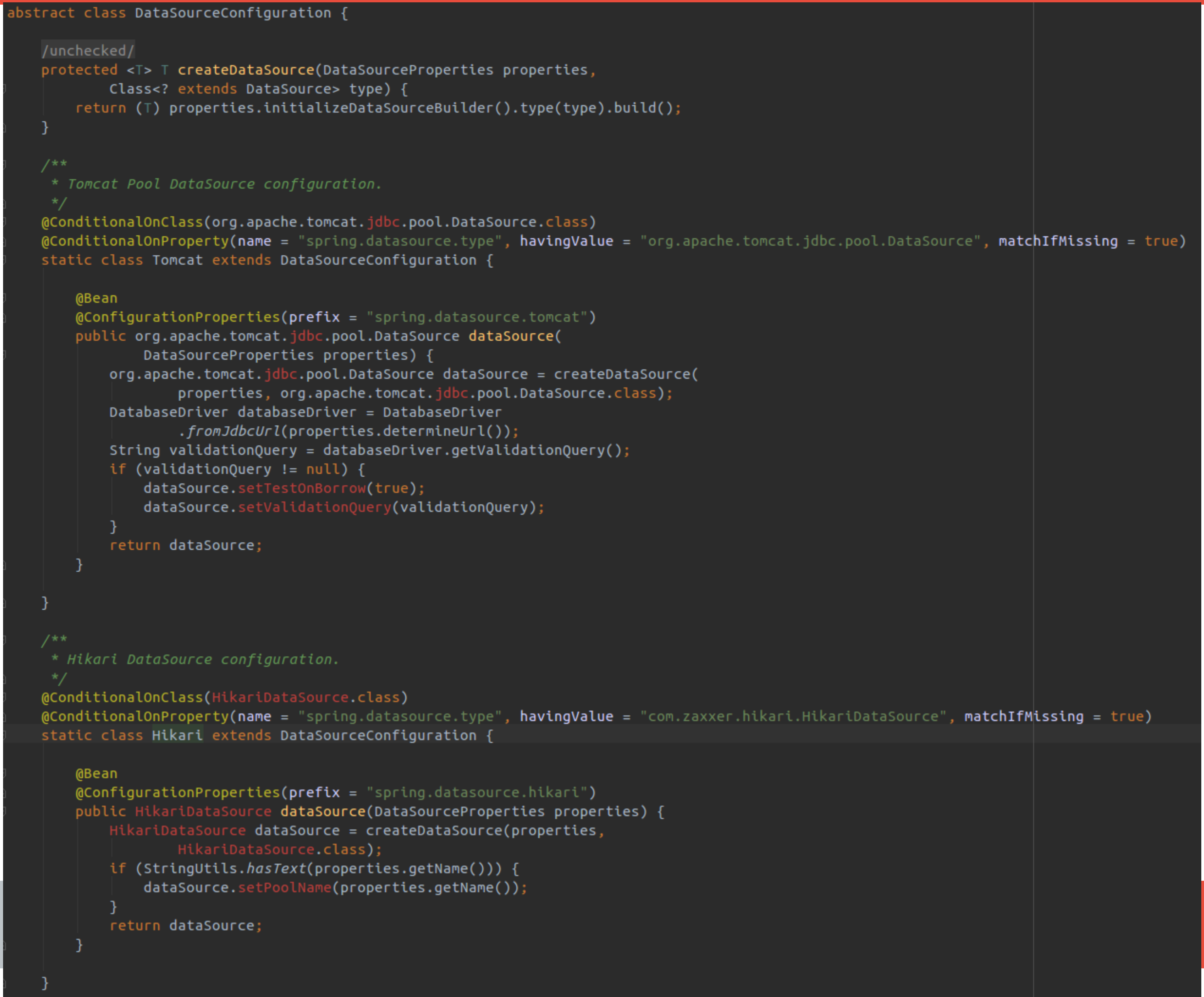
Starters

- spring-boot-starter
- spring-boot-starter-web
- spring-boot-starter-data-jpa
- spring-boot-starter-test
- spring-boot-starter-actuator

Spring Boot's @Conditional Annotations

- Class Conditions
- Bean Conditions
- Property Conditions
- Resource Conditions
- Custom Conditions
- Application Conditions

Example DataSource Auto Configuration



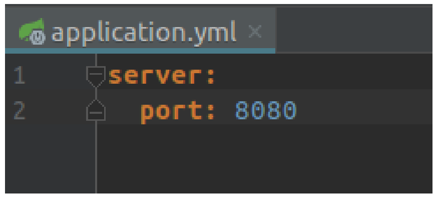
```java
abstract class DataSourceConfiguration {

    /unchecked/
    protected <T> T createDataSource(DataSourceProperties properties,
            Class<? extends DataSource> type) {
        return (T) properties.initializeDataSourceBuilder().type(type).build();
    }

    /**
     * Tomcat Pool DataSource configuration.
     */
    @ConditionalOnClass(org.apache.tomcat.jdbc.pool.DataSource.class)
    @ConditionalOnProperty(name = "spring.datasource.type", havingValue = "org.apache.tomcat.jdbc.pool.DataSource", matchIfMissing = true)
    static class Tomcat extends DataSourceConfiguration {

        @Bean
        @ConfigurationProperties(prefix = "spring.datasource.tomcat")
        public org.apache.tomcat.jdbc.pool.DataSource dataSource(
                DataSourceProperties properties) {
            org.apache.tomcat.jdbc.pool.DataSource dataSource = createDataSource(
                    properties, org.apache.tomcat.jdbc.pool.DataSource.class);
            DatabaseDriver databaseDriver = DatabaseDriver
                    .fromJdbcUrl(properties.determineUrl());
            String validationQuery = databaseDriver.getValidationQuery();
            if (validationQuery != null) {
                dataSource.setTestOnBorrow(true);
                dataSource.setValidationQuery(validationQuery);
            }
            return dataSource;
        }

    }

    /**
     * Hikari DataSource configuration.
     */
    @ConditionalOnClass(HikariDataSource.class)
    @ConditionalOnProperty(name = "spring.datasource.type", havingValue = "com.zaxxer.hikari.HikariDataSource", matchIfMissing = true)
    static class Hikari extends DataSourceConfiguration {

        @Bean
        @ConfigurationProperties(prefix = "spring.datasource.hikari")
        public HikariDataSource dataSource(DataSourceProperties properties) {
            HikariDataSource dataSource = createDataSource(properties,
                    HikariDataSource.class);
            if (StringUtils.hasText(properties.getName())) {
                dataSource.setPoolName(properties.getName());
            }
            return dataSource;
        }

    }
```

Change Application Property

```
java -jar app.jar --server.port=9000
```

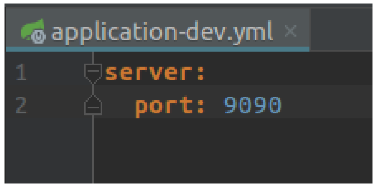
```
java -jar app.jar \
   --spring.config.location=classpath:/default.properties,classpath:/override.properties
```

Configure Multiple Profile


application.yml

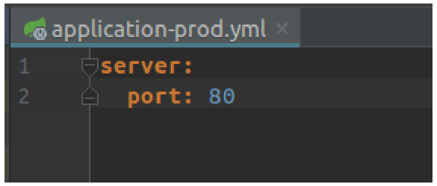
```yaml
server:
  port: 8080
```

Default port 8080


application-dev.yml

```yaml
server:
  port: 9090
```

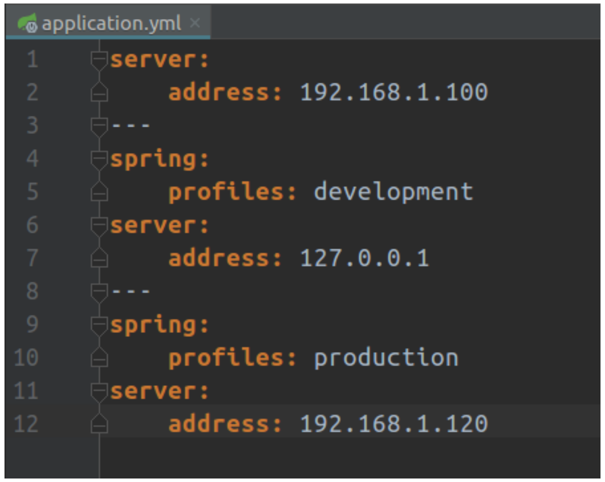
Dev port 9090


application-prod.yml

```yaml
server:
  port: 80
```

Prod port 80

Configure Multiple Profile


application.yml

```yaml
server:
    address: 192.168.1.100
---
spring:
    profiles: development
server:
    address: 127.0.0.1
---
spring:
    profiles: production
server:
    address: 192.168.1.120
```