



Spring Boot

CS544: Enterprise Architecture

Spring Boot

CS544 Enterprise Architecture

Wholeness

- Spring Boot tries to get your project setup as quick as possible.
- It takes “an opinionated view of the Spring platform and third party libraries so you can get started with minimal fuss”.
- Do Less and Accomplish more.

Spring Boot:

DEPENDENCY MANAGEMENT

Typical pom.xml: Starter Parent

```
<?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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>myproject</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <!-- Inherit defaults from Spring Boot -->
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
  </parent>

  <!-- Add typical dependencies for a web application -->
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>

  <!-- Package as an executable jar -->
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>

</project>
```

(Almost) all other
Dependency version
numbers set by
Spring Boot

Plugin packages it as
an executable JAR

Many Spring Boot Starters

Table 13.1. Spring Boot application starters

Name	Description	
<code>spring-boot-starter</code>	Core starter, including auto-configuration support, logging and YAML	
<code>spring-boot-starter-activemq</code>	Starter for JMS messaging using Apache ActiveMQ	
<code>spring-boot-starter-amqp</code>	Starter for using Spring AMQP and Rabbit MQ	
<code>spring-boot-starter-aop</code>	Starter for aspect-oriented programming with Spring AOP and AspectJ	Pom
<code>spring-boot-starter-artemis</code>	Starter for JMS messaging using Apache Artemis	Pom
<code>spring-boot-starter-batch</code>	Starter for using Spring Batch	Pom
<code>spring-boot-starter-cache</code>	Starter for using Spring Framework's caching support	Pom
<code>spring-boot-starter-cloud-connectors</code>	Starter for using Spring Cloud Connectors which simplifies connecting to services in cloud platforms like Cloud Foundry and Heroku	Pom
<code>spring-boot-starter-data-cassandra</code>	Starter for using Cassandra distributed database and Spring Data Cassandra	Pom
<code>spring-boot-starter-data-cassandra-reactive</code>	Starter for using Cassandra distributed database and Spring Data Cassandra Reactive	Pom
<code>spring-boot-starter-data-couchbase</code>	Starter for using Couchbase document-oriented database and Spring Data Couchbase	Pom
<code>spring-boot-starter-data-couchbase-reactive</code>	Starter for using Couchbase document-oriented database and Spring Data Couchbase Reactive	Pom
<code>spring-boot-starter-data-elasticsearch</code>	Starter for using Elasticsearch search and analytics engine and Spring Data Elasticsearch	Pom
<code>spring-boot-starter-data-jpa</code>	Starter for using Spring Data JPA with Hibernate	Pom
<code>spring-boot-starter-data-ldap</code>	Starter for using Spring Data LDAP	Pom
<code>spring-boot-starter-data-mongodb</code>	Starter for using MongoDB document-oriented database and Spring Data MongoDB	Pom
<code>spring-boot-starter-data-mongodb-reactive</code>	Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive	Pom
<code>spring-boot-starter-data-neo4j</code>	Starter for using Neo4j graph database and Spring Data Neo4j	Pom
<code>spring-boot-starter-data-redis</code>	Starter for using Redis key-value data store with Spring Data Redis and the Lettuce	Pom
...		

So that you manage less individual dependencies

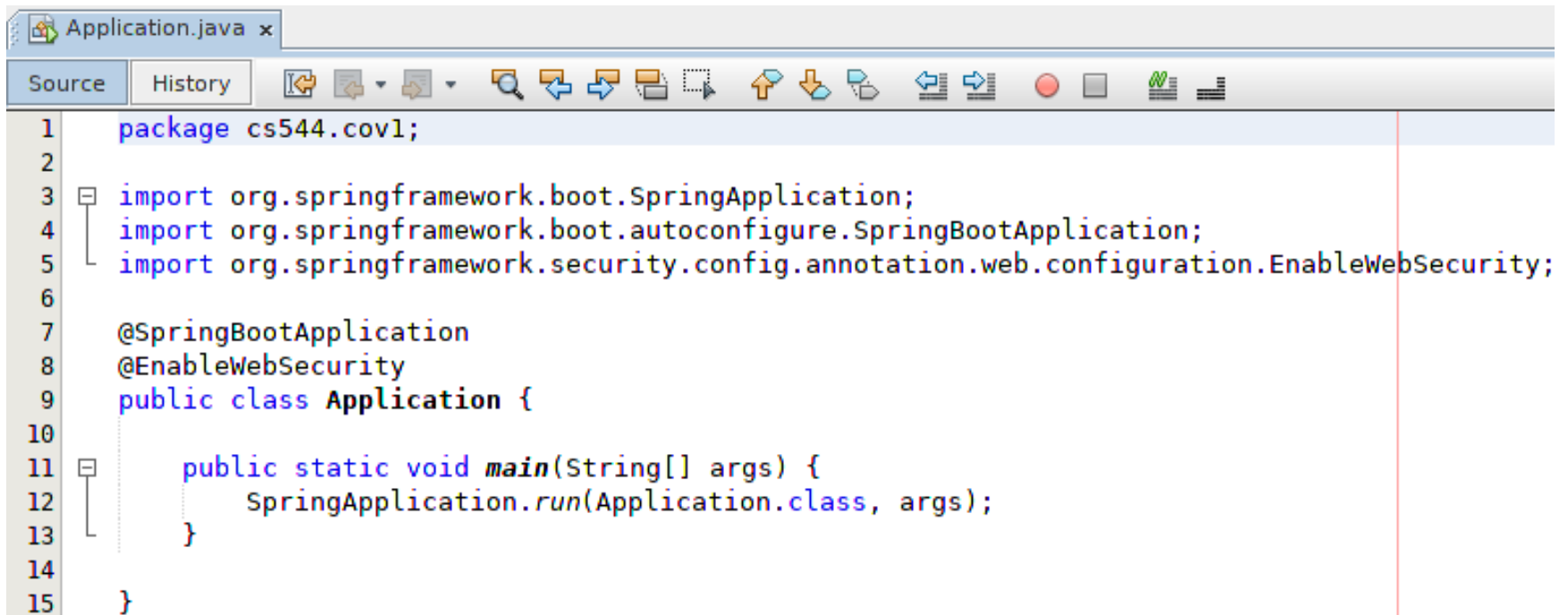
Spring Boot

CODE & CONFIGURATION

Java Config / Main Class

- Spring Boot favors Java-based configuration. Although it is possible to use `SpringApplication` with XML sources, we generally recommend that your primary source be a single `@Configuration` class. Usually the class that defines the main method is a good candidate as the primary `@Configuration`.
- Many Spring configuration examples have been published on the Internet that use XML configuration. If possible, always try to use the equivalent Java-based configuration. Searching for `Enable*` annotations can be a good starting point.

From COV2



The screenshot shows an IDE window titled "Application.java x". The window has a "Source" tab and a "History" tab. Below the tabs is a toolbar with various icons for file operations, search, and execution. The code is as follows:

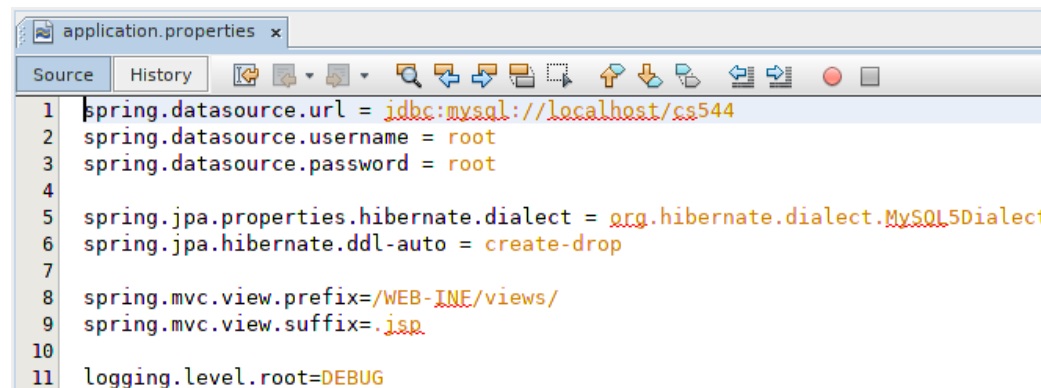
```
1 package cs544.cov1;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
6
7 @SpringBootApplication
8 @EnableWebSecurity
9 public class Application {
10
11     public static void main(String[] args) {
12         SpringApplication.run(Application.class, args);
13     }
14
15 }
```

Auto Configuration

- `@SpringBootApplication`
- Same as writing the following 3:
 - `@EnableAutoConfiguration`
 - `@ComponentScan`
 - `@Configuration`
- `Application.java` therefore is a `JavaConfig` class

Properties or YML

- Autoconfiguration still needs certain values
 - Such as database username and password
- These can be stored in:
application.properties
Or application.yml (YAML is a superset of JSON)

A screenshot of an IDE window titled 'application.properties'. The window has a toolbar with various icons for editing and navigation. The text inside the window is as follows:

```
1 spring.datasource.url = jdbc:mysql://localhost/cs544
2 spring.datasource.username = root
3 spring.datasource.password = root
4
5 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
6 spring.jpa.hibernate.ddl-auto = create-drop
7
8 spring.mvc.view.prefix=/WEB-INF/views/
9 spring.mvc.view.suffix=.jsp
10
11 logging.level.root=DEBUG
```

Additional Configuration

- **Importing Additional Configuration Classes**
 - You need not put all your `@Configuration` into a single class. The `@Import` annotation can be used to import additional configuration classes. Alternatively, you can use `@ComponentScan` to automatically pick up all Spring components, including `@Configuration` classes.
- **Importing XML Configuration**
 - If you absolutely must use XML based configuration, we recommend that you still start with a `@Configuration` class. You can then use an `@ImportResource` annotation to load XML configuration files.

Spring Boot

RUNNING AN APPLICATION

Running

- For our purposes you just run it from your IDE
 - Can also run it from its own Maven target
 - Or run the executable JAR it creates
- You can also include devtools
 - Automatically disabled when running the JAR

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
  </dependency>
</dependencies>
```

DevTools Benefits

- Automatically restarts when it senses changes
- On restart logs changes to autoconfig (delta)
- Option to configure resources that should not trigger a restart when changed
- Also the ability to watch extra resources / paths for changes to trigger restart

Active Learning

- What 3 things does `@SpringApplication` do?
- What would you say is the main benefit of the Spring Boot Dev Tools?

Main Point

- Spring Boot tries to setup as much as possible for you.
- Science of Consciousness: take the right angle and let go.