SpringBoot

Takes an opinionated approach to building Spring applications
and
helps you “get up and running” as quickly as possible

스프링 부트는 생산성을 높인다

References

- Spring Boot Reference Documentation
  - https://docs.spring.io/spring-boot/docs/current/reference/html/

스프링 부트는 스프링의 개발을 쉽게 해준다

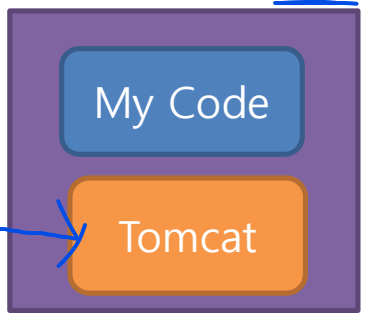
라이브러리 ≒ dependency

Spring Boot

- Make it easier to get started with Spring development
  1) Minimize the amount of manual configuration
     설정의 양을 대폭 줄인다, 프로젝트의 속성 파일과 JAR를
     - Perform auto-configuration based on props files and JAR classpath 기반으로 하여 자동 설정을 해준다
  2) Help to resolve dependency conflicts
     라이브러리 간 충돌을 해결하는데 도움을 준다
  3) Provide an embedded server so you can get started quickly
     - No need to install a server separately

내장 서버이므로 별도의 서버 설치 필요없음

mycoolapp.jar ← 대신 jar 파일이 뚱뚱해짐 (fat jar)



JAR file includes
my application code
AND
the server

Self-contained unit
Nothing else to install

Spring Boot FAQ

그럼 스프링 부트가 스프링 MVC를 대체 하나? 아니다.

- Does Spring Boot replace Spring MVC, Spring Rest, etc. ? 스프링 부트는 도움만 준다
  - No, Spring Boot actually uses those technologies
  - Behind the scenes, Spring Boot uses same code of Spring Framework

자동 설정

1) Auto Configuration

스프링 기반 애플리케이션에는 정말 많은 설정이 있었음

- Spring based applications have a lot of configuration

새로운 프로젝트마다
반복적인 설정!!!

web.xml

```
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

When we use Spring MVC,
we need to configure
a dispatcher servlet

Auto Configuration

Servlet-context.xml

```
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />

<context:component-scan base-package="kr.ac.hansung.web.controllers" />

<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <beans:property name="prefix" value="/WEB-INF/views/" />
   <beans:property name="suffix" value=".jsp" />
</beans:bean>

<resources mapping="/resources/**" location="/resources/" />
```

When we use Spring MVC,
we need to configure
component scan,
a view resolver, and resources

Auto Configuration

dao-context.xml

```
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
        destroy-method="close">
      <property name="driverClassName" value="${jdbc.driverClassName}" />
      <property name="url" value="${jdbc.url}" />
      <property name="username" value="${jdbc.username}" />
      <property name="password" value="${jdbc.password}" />
</bean>

<bean id="entityManagerFactory"
        class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="dataSource" ref="dataSource" />
      <property name="packagesToScan" value="kr.ac.hansung.cse.model" />
        ...
</bean>

<bean id="transactionManager"
        class="org.springframework.orm.jpa.JpaTransactionManager">
      <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

When we use JPA/Hibernate,
we would need to configure a datasource,
an entityManagerFactory and a transaction manager

Auto Configuration

복잡한 설정... 해야할 일이 너무 많다

- Problem: Complex configuration of Spring based apps
  - this is too much of work to do if you want to quickly get up and running
  - If you want to develop another SpringMVC application with similar technical stack, you copy-paste the configuration

자동화 하기 위해 스프링이 하도록 해보자

- Remember one thing: **if you have to do the same thing again and again, you should find an automated way to do it**

- Idea
  - What if Spring is capable of configuring beans automatically?
  - What if you can customize the automatic configuration using simple customizable properties?

자동으로 스프링이 bean을 설정

Auto Configuration

Hibernate가 있으면 데이터 소스도 알아서 설정

When a spring mvc jar is added into an application, can we auto configure some beans automatically?

- How about auto configuring a Dispatcher Servlet
  if Spring MVC jar is on the classpath?
- How about auto configuring a Data Source
  if Hibernate jar is on the classpath?

네가 쓰는 라이브러리를 보고 알아서 해주지만, 커마도 가능

Spring Boot looks at
a) Frameworks available on the CLASSPATH
b) Existing configuration for the application

Based on these, Spring Boot provides basic configuration needed to configure the application with these frameworks. This is called **Auto Configuration**

Auto Configuration

- For example,
  - if you have the spring-webmvc dependency in your classpath,
    - Spring Boot assumes you are trying to build a SpringMVC-based web application and automatically tries to register DispatcherServlet if it is not already registered
  - If you have any embedded database drivers in the classpath, such as H2 or HSQL, and if you haven't configured a DataSource bean explicitly,
    - then Spring Boot will automatically register a DataSource bean using in-memory database settings

Auto Configuration

pom.xml

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifa
    </dependency>

    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

이름 자유 X

↓

Application.properties

```properties
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp

spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/dbdemo
spring.datasource.username=root
spring.datasource.password=csedbadmin
```

의존성 추가 그리고
필요한 Property만 세팅하면 끝!!!

Spring Boot Autoconfiguration Works

추가하면
자동설정 켜기

```
@Configuration
@EnableAutoConfiguration
@ComponentScan
public class Application
{
    ...
}
```

enables the autoconfiguration of Spring ApplicationContext

쉬운 라이브러리 관리

2) Easy Dependency Management (SpringBoot Starters)

- We would need to identify the <u>dependencies</u> we want to use, which <u>versions</u> of dependencies to use

Dependencies in Spring MVC

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webmvc</artifactId>
   <version>4.2.2.RELEASE</version>
</dependency>

<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.5.3</version>
</dependency>

<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>5.0.2.Final</version>
</dependency>
```

아... 이 기능 쓰려면 어떤 라이브러리 써야 하지?

SpringBoot Starters 가 알아서 해줌

Easy Dependency Management
(SpringBoot Starters)

The "**spring-boot-starter-***" is pre-configured
with the most commonly used library dependencies
so that we don't have to search for the compatible library versions
and configure them manually

SpringBoot Starters 적용,
뒤에 web 붙으면 web 전용

Spring Boot Starters

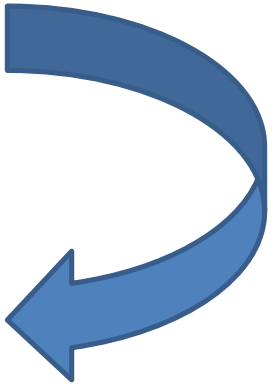
A collection of Maven dependencies
(Compatible versions)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Easy dependency Management
(SpringBoot Starters)

Dependencies in SpringBoot

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```



Added Dependencies

- Spring - core, beans, context, aop
- Web MVC - (Spring MVC)
- Jackson - for JSON Binding
- Validation - Hibernate Validator, Validation API
- Embedded Servlet Container - Tomcat
- Logging - logback, slf4j

Easy dependency Management
(SpringBoot Starters)

- There are 30+ Spring Boot Starters

Name	Description
spring-boot-starter-web	Build web apps, including validation, REST. Uses Tomcat as default embedded server
spring-boot-starter-security	Adding spring Security support
spring-boot-starter-data-jpa	Spring database support with JPA and Hibernate
spring-boot-starter-logging	For Logging using logback
…	…

We would not need to worry about
either these dependencies or their compatible versions

Tomcat이 내장되어 있다

3) Embedded Servlet Container Support

- When we run the **main()** method it started tomcat as an embedded container
- We don't have to deploy our application on any externally installed tomcat server

- Packaging type in pom.xml is 'jar' not 'war'

대신 jar이 뚱뚱하다 (fat jar)

그리고 java의 jar 과는 패키징 방법이 좀 다름

프로그램이 잘 동작하나 모니터링

4) SpringBoot Actuator

(사실 이것보다는 앞의 3개가 중요)

- Exposes endpoints to monitor and manage your application
- REST endpoints are <u>automatically</u> added to your application

모니터링하고 관리하도록 하는 endpoint를 제공

- Some actuator endpoints are:
  - The /beans endpoint shows all the beans registered in our application
  - The /mappings endpoint shows the application URL, mappings Environment details and configuration Parameter values
  - …

SpringBoot Actuator

- Adding the dependency to your POM file

추가해야 함

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

SpringBoot Actuator

- Exposing endpoints
  - By default, only **/health** is exposed
  - Endpoints are prefixed with: /actuator
    - health endpoint : /actuator/health

  - To expose all actuator endpoints over HTTP except the env and beans endpoints

src/main/resources/application.properties

management.endpoints.web.exposure.include=* ← 모든 endpoint를 열어라
management.endpoints.web.exposure.exclude=env,beans

SpringBoot Actuator

Health Endpoint

- /health checks the status of your app
- Normally used by monitoring apps to see if your app is up or down

localhost:8080/helloSpringBoot/actuator/health


```
{
    "status": "UP"
}
```

SpringBoot Actuator

- To expose **/info**

  src/main/resources/application.properties

management.endpoints.web.exposure.include=*

management.info.env.enabled=true

앱 정보

info.app.name=My Super Cool App
info.app.description=A crazy and fun app, yoohoo!
info.app.version=1.0.0

Properties starting with "info" will be used by /info

localhost:8080/helloSpringBoot/actuator/info


```
{
  "app": {
    "name": "My Super Cool App",
    "description": "A crazy and fun app, yoohoo!",
    "version": "1.0.0"
  }
}
```

SpringBoot Actuator

endpoint

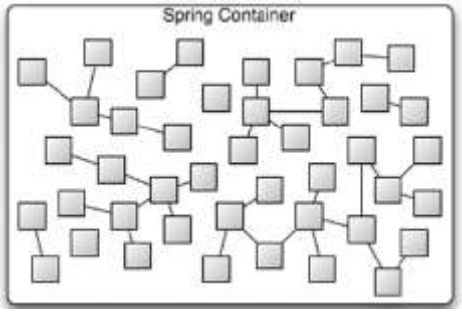
Endpoint ID	Description
beans	List of all beans registered in the Spring application context
mappings	List of all @RequestMapping paths
metrics	Shows various metrics information of your application
env	Displays current environment properties.
loggers	Displays and modifies the configured loggers.
…	

Three ways to configure bean

bean을 생성하고 설정하기

(3가지 방법)

Configuring beans



① XML based configuration

```
<bean id="userService" class="kr.ac.hansung.myapp.service.UserService">
    <property name="userDao" ref="userDao"/>
</bean>

<bean id="userDao" class="kr.ac.hansung.myapp.dao.JdbcUserDao">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
            destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost:3306/test"/>
    <property name="username" value="root"/>
    <property name="password" value="secret"/>
</bean>
```

Configuring beans

② Annotation based configuration

```
@Service
public class UserService
{

    @Autowired
     private UserDao userDao;

      ...
}
```

```
@Repository
public class JdbcUserDao
{

    @Autowired
     private DataSource dataSource;

     ...

}
```

<가장 많이 사용>

Configuring beans

③ Java based configuration

클래스

Spring Java-based Configuration

Registering beans

```
@Configuration
public class AppConfig
{
    @Bean
    public UserService userService(UserDao dao){
        return  new UserService(dao);
    }
    @Bean
     public UserDao userDao(DataSource dataSource){
        return  new JdbcUserDao(dataSource);
    }
    @Bean
    public DataSource dataSource(){
        BasicDataSource dataSource = new BasicDataSource();

        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/test");
        dataSource.setUsername("root");
        dataSource.setPassword("secret");
        return dataSource;
    }
}
```

메소드명을 bean 이름이 됨

컨테이너에 bean으로 등록

Configuring beans

- @Configuration
  - Used to annotate Configuration classes which can contain bean definition methods annotated with @Bean
  - @Configuration annotation is a more specialized version of the @Component annotation
  - *Note: @Controller, @Service, @Repository, and @Configuration* are all meta-annotations of *@Component* 애너 처럼, @Component 의 종류이다.
- @Bean
  - tells Spring that a method annotated with @Bean will return an object that should be registered as a bean
  - the method name works as bean ID 리턴된 객체는 bean이 되고, 메소드명이 bean ID가 됨

    - @Configuration annotates a configuration class
    - @Bean annotates methods that create Spring beans
    - This allows centralized management of application beans using Java-based configuration