Spring Boot - Create Simple Custom Spring Boot Starter - AutoConfiguration

- **Create Simple Custom Spring Boot Starter/AutoConfiguration**
  First, lets get the Fluff out of the way.
  Spring Boot **Starter** is only to provide dependencies for **AutoConfiguration**. Itself only contains a POM file describing the jars needed. (Check the size and the content of the **Starter** jar file, you will know what I mean).
  It is AutoConfiguration that does the the "magic" and that is what we are going to do here.

  To Create Simple AutoConfiguration - that will automatically expose a Restful endpoint (just like Actuator) - where you can include it in other project as dependency and the endpoint will automatically be *enabled* for the application that includes it (*think of it as a plugin*).

  1. **Create a new maven project** - with ***spring-boot-starter-web*** dependency

     - NOTE: this will eventually create a JAR file named **myAutoConfigTest-1.0-SNAPSHOT.jar**

```
<?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.notimeforfluff</groupId>

<artifactId>myAutoConfigTest</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>


<dependencyManagement>

<dependencies>

<dependency>

<!-- Import dependency management from Spring Boot -->

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-dependencies</artifactId>

<version>1.5.1.RELEASE</version>
```

```
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<properties>
<java.version>1.8</java.version>
</properties>

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

</dependencies>

<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.2</version>
<configuration>
<source>${java.version}</source>
<target>${java.version}</target>
</configuration>
</plugin>
```

```
</plugins>
</build>


</project>
```

2. **Create a Configuration class** with conditions with the following annotations
   - `@Configuration` - Meta-annotated with @Component, so the classes will be a candidate for component scanning
   - `@AutoConfigureAfter` - Hint for that an auto-configuration should be applied after other specified auto-configuration classes
   - `@RestController` - marking class as REST controller

```
package com.notimeforfluff.rest;

@Configuration

@AutoConfigureAfter({WebMvcAutoConfiguration.class})

@RestController

public class MyRestAutoConfiguration{


@RequestMapping({"/autoConfigRESTHello"})

String sayHey() {

return "Hey  I've been auto discovered!! " ;

}

}
```

The above code
means: Configure/Enable ***"/autoConfigRESTHello"*** endpoint **AFTER**`WebMvcAutoConfiguration.class` has been loaded/configured.

3. **Tell Spring how/where to locate the auto-configuration candidates**

Create a file named **spring.factories** under `src/main/resources/META-INF` with the following content

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=com.notimeforfluff.rest.MyRestAutoConfiguration
```

And That Is It!!

4. Now Jar up the code with

```
mvn clean install
```

This will install the maven dependency in your local .m2 and you can include the following as dependency for any Spring Boot web project to automatically expose *"/autoConfigRESTHello"* endpoint.

```
<dependency>
<groupId>com.chris.test</groupId>
<artifactId>myAutoConfigTest</artifactId>
<version>1.0-SNAPSHOT</version>
</dependency>
```

Example: http://localhost:8080/autoConfigRESTHello