## Spring MVC - Controller

- **@Controller**
  Annotation to indicate that the class is a controller class.

- **@RestController**
  A convenience annotation that is itself annotated with @Controller and @ResponseBody. Used in controllers that will behave as RESTful resources.

- **@RequestMapping**
  Annotation used to be used on methods in @RestController classes. You can provide an URI to be served as RESTful service.

- **@ModelAttribute**
  Annotation used to bind values present in views.

## Spring Boot - Auto Configuration

- **@Configuration**
  Annotation used to provide configurations.

- **@Bean**
  Annotation that acts like a `@Provider` where you can define how the bean is instantiated when a `@Inject`ion of that type is requested. Instances of `@Bean` annotated methods will act as singletons.

## Properties Evaluation Sequence

- **Command-line arguments**
  `java -Dproject.name=Test -jar app.jar`

- **System properties**
  `System.getProperties()`

- **Environment Variable**
  `export PROJECT_NAME=Test`

- **External properties/yml file**
  `project.name=Test`

- **Internal properties/yml file**
  `project.name=Test`

The default properties/yml files are `application.properties` and `application.yml` and they are located in `/src/resources`.

## Spring Boot Initializer

- **http://start.spring.io**
  Web service that allows the user to specify the project metadata and dependencies as well as download the initial structure.

- **Spring CLI**
  A CLI tool that interacts with [http://start.spring.io](http://start.spring.io) service to scaffold a new project.

- **Spring Tool Suit**
  Eclipse-based IDE that also interacts with [http://start.spring.io](http://start.spring.io) to scaffold a new project.

- **IntelliJ IDEA**
  IntelliJ also provides a way of creating a new project via [http://start.spring.io](http://start.spring.io).

## Configuration (cont)

- **@ConditionalOnClass**
  Only available if the Tomcat class is found in the classpath.

- **@ConditionalOnProperty**
  Only available if the property `tomcat.version` is set to true.

Auto configuration is just the combination of `@Configuration` and `@ConditionalOn` annotations in order to correctly register beans.
**Dependency Injection**

@Resource  
Annotation used to inject an object that is already in the Application Context. It searches the instance by name. It also works on setter methods.

@Autowired  
Annotation used to inject objects in many possible ways, such as: instance variable, constructor and methods. It does not rely on name as @Resource, so, for multiple concrete implementations, the @Qualifier annotation must be used with it.

@Qualifier  
Annotation used to distinguish between multiple concrete implementations. Used alongside with @Autowired annotation that does not rely on name.

@Primary  
Annotation used when no name is provided telling Spring to inject an object of the annotated class. Used along with @Component.

@Component  
Generic stereotype annotation used to tell Spring to create an instance of the object in the Application Context. It's possible to define any name for the instance, the default is the class name as camel case.

@Controller  
Stereotype annotation for presentation layer.

@Repository  
Stereotype annotation for persistence layer.

@Service  
Stereotype annotation for service layer.

**Profile**

spring.profiles.active  
Property to be set in application.properties in order to tell Spring what profiles are active.

@Profile("dev")  
Annotation used to define which profile can execute the annotated method.

**Spring Boot - Basics**

@SpringBootApplication  
Initial annotation that comprises the following annotations: @SpringBootConfiguration, @EnableAutoConfiguration and @ComponentScan.

@SpringBootConfiguration  
Indicates that a class provides Spring Boot application configuration.

@EnableAutoConfiguration  
Enable auto-configuration of the Spring Application Context, attempting to guess and configure beans that you are likely to need.

@ComponentScan  
Configures component scanning directives for use with @Configuration classes.

Most of the time you will need only to declare the @SpringBootApplication annotation.

**Spring Boot - Example**

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