Auto-configuration

Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added. For example, if `HSQLDB` is on your classpath, and you have not manually configured any database connection beans, then Spring Boot auto-configures an in-memory database.

You need to opt-in to auto-configuration by adding the `@EnableAutoConfiguration` or `@SpringBootApplication` annotations to one of your `@Configuration` classes.

TIP

You should only ever add one `@SpringBootApplication` or `@EnableAutoConfiguration` annotation. We generally recommend that you add one or the other to your primary `@Configuration` class only.

configuration. For example, if you add your own `DataSource` bean, the default embedded database support backs away.

If you need to find out what auto-configuration is currently being applied, and why, start your application with the `--debug` switch. Doing so enables debug logs for a selection of core loggers and logs a conditions report to the console.

Disabling Specific Auto-configuration Classes

If you find that specific auto-configuration classes that you do not want are being applied, you can use the exclude attribute of `@SpringBootApplication` to disable them, as shown in the following example:

Java Kotlin

```java
@SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })
public class MyApplication {

}
```

If the class is not on the classpath, you can use the `excludeName` attribute of the annotation and specify the fully qualified name instead. If you prefer to use `@EnableAutoConfiguration` rather than `@SpringBootApplication`, `exclude` and `excludeName` are also available. Finally, you can also control the list of auto-configuration classes to exclude by using the `spring.autoconfigure.exclude` property.

TIP

You can define exclusions both at the annotation level and by using the property.

NOTE

Even though auto-configuration classes are `public`, the only aspect of the class that is considered public API is the name of the class which can be used for disabling the auto-configuration. The actual contents of those classes, such as nested configuration classes or bean methods are for internal use only and we do not recommend using those directly.

Auto-configuration Packages

Auto-configuration packages are the packages that various auto-configured features look in by default when scanning for things such as entities and Spring Data repositories. The `@EnableAutoConfiguration` annotation (either directly or through its presence on `@SpringBootApplication`) determines the default auto-configuration package. Additional packages can be configured using the `@AutoConfigurationPackage` annotation.