# 使用spring boot

本章节是介绍如何使用spring boot。蕴含的主题诸如如何构建系统、自动化配置（auto-configuration）和如何去运行你的应用。我们还介绍了一些Spring Boot最佳实践（他仅仅是另外一个你可以使用的库），这仅仅是一些建议，如果使用的话，会使你开发变得更简单一点。

如果你开始使用Spring Boot，你应该在进入这一节之前阅读入门指南。

## 构建系统

这里强烈建议你构建系统的时候能支持依赖管理（dependency management），这可以消费发布到“Maven Central”仓库的组件。我们推荐你选择Maven或者Gradle。有可能让Spring Boot与其他系统（例如Ant）一起工作，但是它们并不是特别受支持的。

## 依赖管理（dependency management）

Spring Boot的每次发布都提供了一个经过精心编排的依赖项列表.。在实践中，您不需要为构建配置中的任何依赖项提供版本号，Spring Boot会去帮你管理好版本信息。当你更新Spring Boot版本的时候，这些依赖会跟着一致升级。

注意：如有必要，你依然可以自己指定一个版本来覆盖Spring Boot的推荐版本

这个精选的列表包含了所有可以使用Spring Boot的Spring模块，以及一个精挑细选的第三方库列表。这个列表可以作为标准的依赖列表（spring-boot-dependencies），可以同时使用Maven和Gradle。

警告：Spring Boot的每个版本都与Spring Framework的基本版本相关联。我们强烈建议您不要指定它的版本。

## Maven

Maven用户可以从spring boot-start-parent项目中继承来获得合理的默认值。父项目规定以下特性:

* Java1.8 作为默认的编译版本
* UTF-8为默认编码
* 部分依赖管理，继承自spring-boot-dependencies pom，管理常见依赖项的版本。这种依赖性管理可以让您在自己的pom中不需要配置<version>这个标签
* 合理的资源过滤
* 科学的插件配置（exec plugin, Git commit ID, and shade）
* 合理的资源过滤，例如：application.properties 和 application.yml；还包含特殊的文件（例如：application-dev.properties 和 application-dev.yml）

注意，application.properties 和 application.yml文件都能解析spring 风格的占位符（${…}），Maven需要将占位符换为@..@。（你可以通过设置一个Maven属性来覆盖它，这个属性是：resource.delimiter）

**继承自 Starter Parent**

将你的项目配置为继承自spring-boot-starter-parent, 按照以下方式设置父依赖:

<!-- Inherit defaults from Spring Boot -->

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.3.RELEASE</version>

</parent>

注意：您应该只需要在这个依赖项上指定Spring Boot版本号。如果你导入额外的starters, 您可以安全地省略版本号。

虽然有了这种配置，但是你依然可以在你的项目中指定个别版本的依赖。例如，升级Spring Data 为其他的版本，添加如下依赖到你的pom.xml里面即可：

<properties> <spring-data-releasetrain.version>Fowler-SR2</spring-data-releasetrain.version>

</properties>

提示：查看spring-boot-dependencies pom可以看到支持的列表。

**缺少Parent POM的方式使用Spring Boot**

并非所有人都喜欢继承spring-boot-starter-parent POM。你可能有你自己的公司标准的父Pom，你需要使用或者你可能更倾向于明确地声明你所有的Maven配置。

如果你不想使用spring-boot-starter-parent，您仍然可以通过使用scope=import依赖性来保持依赖性管理（而不是插件管理）的好处，如下所列：

<dependencyManagement>

<dependencies>

<dependency>

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

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

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

<version>2.0.3.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

**前面的示例配置不允许您通过使用属性来覆盖个别依赖项，如上所述。为了达到相同的结果，您需要在spring-boot-dependencies入口之前在项目的dependencyManagement中添加一个配置。例如，升级Spring Data版本的时候，你需要在你的pom.xml中添加下面的元素：**

<dependencyManagement>

<dependencies>

<!-- Override Spring Data release train provided by Spring Boot -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-releasetrain</artifactId>

<version>Fowler-SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

<dependency>

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

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

<version>2.0.3.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

注意：在前面的例子中，我们指定了BOM，但是任何依赖类型都可以以相同的方式覆盖。

**使用Spring Boot Maven Plugin**

Spring Boot提供了一个Maven plugin，可以用它来为项目打出一个可执行jar包。如果你需要使用这个plugin，只需要在<plugins>中添加即可，示例如下所示：

<!-- Package as an executable jar -->

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

注意：如果你使用Spring Boot starter parent pom，你只需要添加插件。不需要配置它，除非您想要更改parent中定义的设置。

## Gradle

## Ant

## Starters

Starters是一组方便使用的依赖，你可以在你的应用中引入并使用它们。你可以为所有的Spring和相关技术提供一站式服务而无需通过样本代码和复制粘贴的依赖去加载。例如，如果你需要使用spring和JPA去操作数据库，只需要在你的项目中引入spring-boot-starter-data-jpa依赖即可。

Starters包含许多依赖项，可以满足您需要快速地启动一个项目并使用一致的、受支持的管理传递依赖关系集。

**命名规则**

所有的官方starters遵循类似的命名规则；spring-boot-starter-\*，\*标识了特定的应用类型。这种命名规则是为了帮助你快速的找到你需要的starter。Maven在非常多的IDE里面集成了，可以让你根据名字查询依赖。例如，安装了适当插件的Eclipse或STS，你可以在POM编辑器中按ctrl+space键，并键入“spring boot-starter”来获取一个完整的依赖列表。

正如在“创建您自己的Starts”部分中所解释的那样，第三方的starters不应该使用spring-boot作为前缀，因为它是为官方Spring Boot artifacts预留的。相反，第三方启动器通常以项目的名称开始。例如，一个名为thirdpartyproject的第三方启动项目通常会被命名为thirdpartyproject-sprin-boot-starter。

下面的应用程序starters是由Spring Boot在groupId为org.springframework.boot下提供的:

Spring Boot应用的starters

|  |  |  |
| --- | --- | --- |
| 名字 | 描述 | POM |
| spring-boot-starter | 核心starter，包括自动配置支持、日志记录和YAML | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter/pom.xml) |
| spring-boot-starter-activemq | 使用Apache ActiveMQ的JMS消息传递starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-activemq/pom.xml) |
| spring-boot-starter-amqp | 使用了Spring AMQP和Rabbit MQ的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-amqp/pom.xml) |
| spring-boot-starter-aop | 使用Spring Aop和AspectJ实现面向切面编程的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-artemis/pom.xml) |
| spring-boot-starter-batch | Spring Batch的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-batch/pom.xml) |
| spring-boot-starter-cache | Spring Framework支持的缓存starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-cache/pom.xml) |
| spring-boot-starter-cloud-connectors | 使用Spring Cloud的starter，在Cloud Foundry和Heroku等云平台上简化连接到服务的连接器的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-cloud-connectors/pom.xml) |
| spring-boot-starter-data-cassandra | 使用了Cassandra和Spring Data Cassandra的分布式数据库starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-cassandra/pom.xml) |
| spring-boot-starter-data-cassandra-reactive | 用于使用分布式数据库Cassandra和Spring Data Cassandra Reactive 的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-cassandra-reactive/pom.xml) |
| spring-boot-starter-data-couchbase | 用于使用基于文档的数据库Couchbase和Spring Data Couchbase 的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-cassandra-reactive/pom.xml) |
| spring-boot-starter-data-couchbase-reactive | 用于使用基于文档的数据库Couchbase和Spring Data Couchbase Reactive 的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-couchbase-reactive/pom.xml) |
| spring-boot-starter-data-elasticsearch | 使用Elasticsearchsearch和Spring Data Elasticsearch实现分析搜索引擎的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-elasticsearch/pom.xml) |
| spring-boot-starter-data-jpa | 基于Hibernate实现的Spring Data JPA starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-jpa/pom.xml) |
| spring-boot-starter-data-ldap | Spring Data LDAP的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-ldap/pom.xml) |
| spring-boot-starter-data-mongodb | 用于使用基于文档的数据库MongoDB和Spring Data MongoDB 的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-mongodb/pom.xml) |
| spring-boot-starter-data-mongodb-reactive | 用于使用基于文档的数据库MongoDB和Spring Data MongoDB Reactive的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-mongodb-reactive/pom.xml) |
| spring-boot-starter-data-neo4j | 用于使用图形数据库Neo4j和Spring Data Neo4j的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-neo4j/pom.xml) |
| spring-boot-starter-data-redis | 用于使用Spring Data Redis和Lettuce客户端操作键-值存储的Redis的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-redis/pom.xml) |
| spring-boot-starter-data-redis-reactive | 用于使用Spring Data Redis reactive和Lettuce客户端操作键-值存储的Redis的starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-redis-reactive/pom.xml) |
| spring-boot-starter-data-rest | 用于使用Spring Data REST暴露基于REST的Spring Data仓库 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-rest/pom.xml) |
| spring-boot-starter-data-solr | 通过Spring Data Solr使用Apache Solr搜索平台 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-data-solr/pom.xml) |
| spring-boot-starter-freemarker | 用于使用FreeMarker模板引擎构建MVC web应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-freemarker/pom.xml) |
| spring-boot-starter-groovy-templates | 用于使用Groovy模板引擎构建MVC web应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-groovy-templates/pom.xml) |
| spring-boot-starter-hateoas | 用于使用Spring MVC和Spring HATEOAS实现基于超媒体的RESTful web应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-hateoas/pom.xml) |
| spring-boot-starter-integration | 用于使用Spring Integration | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-integration/pom.xml) |
| spring-boot-starter-jdbc | 对JDBC的支持（使用HikariCP连接池） | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jdbc/pom.xml) |
| spring-boot-starter-jersey | 用于使用JAX-RS和Jersey构建RESTful web应用，可使用spring-boot-starter-web替代 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jersey/pom.xml) |
| spring-boot-starter-jooq | 用于使用JOOQ访问SQL数据库，可使用spring-boot-starter-data-jpa或spring-boot-starter-jdbc替代 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jooq/pom.xml) |
| spring-boot-starter-json | 用于读写json | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-json/pom.xml) |
| spring-boot-starter-jta-atomikos | 用于使用Atomikos实现JTA事务 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jta-atomikos/pom.xml) |
| spring-boot-starter-jta-bitronix | 用于使用Bitronix实现JTA事务 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jta-bitronix/pom.xml) |
| spring-boot-starter-jta-narayana | 用于使用Narayana实现JTA事务 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jta-narayana/pom.xml) |
| spring-boot-starter-mail | 用于使用Java Mail和Spring Frameworke mail发送支持 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-mail/pom.xml) |
| spring-boot-starter-mustache | 用于使用Mustache模板引擎构建MVC web应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-mustache/pom.xml) |
| spring-boot-starter-quartz | 基于Quartz实现的任务调度 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-quartz/pom.xml) |
| spring-boot-starter-security | 对Spring Security的支持 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-security/pom.xml) |
| spring-boot-starter-test | 用于测试Spring Boot应用，支持常用测试类库，包括JUnit, Hamcrest和Mockito | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-test/pom.xml) |
| spring-boot-starter-thymeleaf | 用于使用Thymeleaf模板引擎构建MVC web应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-thymeleaf/pom.xml) |
| spring-boot-starter-validation | 用于使用Hibernate Validator实现Java Bean校验 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-validation/pom.xml) |
| spring-boot-starter-web | 用于使用Spring MVC构建web应用，包括RESTful。Tomcat是默认的内嵌容器 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-web/pom.xml) |
| spring-boot-starter-web-services | 对Spring Web服务的支持 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-web-services/pom.xml) |
| spring-boot-starter-webflux | 提供用Spring Framework创建webflux应用的支持 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-webflux/pom.xml) |
| spring-boot-starter-websocket | 用于使用Spring Framework的WebSocket支持构建WebSocket应用 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-websocket/pom.xml) |

除了应用级别的starters，下面的starters还可以用来添加生产就绪的特性：

Spring Boot生产starters

|  |  |  |
| --- | --- | --- |
| spring-boot-starter-actuator | 用于使用Spring Boot的Actuator，它提供了production ready功能来帮助你监控和管理应用程序 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-actuator/pom.xml) |

最后，Spring Boot还包括以下的starters，如果您想要排除或更换特定方面的技术，可以使用它们：

Spring Boot技术级别的starters

|  |  |  |
| --- | --- | --- |
| spring-boot-starter-jetty | 用于使用Jetty作为内嵌servlet容器，可使用spring-boot-starter-tomcat替代 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-jetty/pom.xml) |
| spring-boot-starter-log4j2 | 用于使用Log4j2记录日志，可使用spring-boot-starter-logging代替 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-log4j2/pom.xml) |
| spring-boot-starter-logging | 用于使用Logback记录日志，默认的日志starter | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-logging/pom.xml) |
| spring-boot-starter-reactor-netty | 使用Reactor Netty做为内嵌的HTTP服务器 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-reactor-netty/pom.xml) |
| spring-boot-starter-tomcat | 用于使用Tomcat作为内嵌servlet容器，spring-boot-starter-web使用的默认servlet容器 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-tomcat/pom.xml) |
| spring-boot-starter-undertow | 用于使用Undertow作为内嵌servlet容器，可使用spring-boot-starter-tomcat替代 | [POM](https://github.com/spring-projects/spring-boot/blob/v2.0.3.RELEASE/spring-boot-project/spring-boot-starters/spring-boot-starter-undertow/pom.xml) |

提示：对于另外一个社区贡献的starters，请参阅GitBub上面的spring-boot-starters模块的[README](https://github.com/spring-projects/spring-boot/blob/master/spring-boot-project/spring-boot-starters/README.adoc)文件。

# 构建你的代码

Spring Boot不需要任何额外的代码就能工作。然而，这里有一些非常好的实践能帮助到你。

## 使用“default”（默认）包

当一个类没有任何的包声明的时候，他就被认为是“default package”(默认包)。通常是不鼓励使用“default package”的。它可能会给使用@ComponentScan, @EntityScan或者@SpringBootApplication注解的Spring Boot应用程序带来特别的问题，因为每个jar中的每个类都被读取。

提示：我们建议您遵循Java推荐的包命名约定，并使用反向域名（例如：com.example.project）。

## 确定主应用程序类（Main Application Class）

我们通常建议您将主应用程序类放在根包，在所有的其他类之上。@SpringBootApplication注解通常就放在你的main类上，它隐式地为项目定义了一个基本的“search package”（包搜索范围）。例如，如果你写一个JPA的应用程序，@SpringBootApplication注解会被用来搜索所有蕴含@Entity注解的类。使用根包，也就仅仅允许程序扫描你的项目。

提示：如果你不愿意使用@SpringBootApplication注解，你也可以引入@EnableAutoConfiguration和@ComponentScan注解来代替，他们定义了同样的行为。

下面的清单显示了一个典型的布局：

com

+- example

+- myapplication

+- Application.java

|

+- customer

| +- Customer.java

| +- CustomerController.java

| +- CustomerService.java

| +- CustomerRepository.java

|

+- order

+- Order.java

+- OrderController.java

+- OrderService.java

+- OrderRepository.java

Application.java声明了一个主应用程序，包含了一个基本的注解@SpringBootApplication，如下所示：

**package** com.example.myapplication;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

# 配置类（Configuration Classes）

Spring Boot支持基于java的配置类。尽管可以使用SpringApplication XML的方式，我们通常建议你使用一个@Configuration类去配置。通常main 方法的类是@Configuration类的非常好的候选类。

提示：许多Spring配置示例已经在Internet上发布，它们使用XML配置。如果可能的话，使用等价的基于java的配置去替代。搜索Enable\*注解会是一个很好的切入点。

## 引入额外的配置类

你不需要在一个类中完成所有的@Configuration。@Import注解可以引入其他的配置类。你也可以用@ComponentScan注解让Spring自动注入所有的组件，包含@Configuration配置类。

## 引入XML配置

如果你需要使用基于xml的配置。我们建议你依然使用一个包含@Configuration的类，你可以在这个类上通过@ImportResource注解引入你的xml配置文件。

# 自动注入（Auto-configuration）

Spring Boot自动注入试图通过引入jar依赖的方式自动的为你的Spring应用完成配置。例如，你的classpath中有HSQLDB，你就不需要有任何手动配置数据库连接的beans了，Spring Boot的自动注入会自动帮你在内存中完成配置。

你需要在你的@Configuration类中选择自动注入的注解@EnableAutoConfiguration或者@SpringBootApplication。

提示：你仅仅需要引入@SpringBootApplication 或者@EnableAutoConfiguration注解。我们通常建议您仅将一个或另一个添加到主@Configuration类中。

## 逐步更换自动配置

自动注入是非侵入性的。在任何时候，你都可以通过你自定义的配置去替换自动配置的特定部分。例如，如果添加你自定义的DataSource bean，默认的嵌入式的数据库支持就会低于你的优先级。

如果你需要找出当前正在应用的自动配置，和为什么，可以用debug的方式启动你的应用程序。这样做可以为选择的核心loggers启用debug logs，并将条件输出到控制台。

## 禁用特定的自动配置类

如果你需要禁用特定的一系列自动配置类生效，你可以在@EnableAutoConfiguration注解中使用exclude属性去禁用它们，如下示例：

**import** org.springframework.boot.autoconfigure.EnableAutoConfiguration;

**import** org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.**class**})

**public** **class** MyConfiguration {

}

如果类没有在classpath中找到，你可以使用在注解中使用excludeName属性写上全类名来代替。最后，您还可以使用spring.autoconfigure.exclude属性控制要排除的自动配置类列表。

提示：您可以在注释级别和使用属性定义排除项。

# Spring Beans 和 依赖注入（Dependency Injection）

你可以免费使用任何Spring Framework技术去定义你的beans和注入它们的依赖项（injected dependencies）。例如，我们经常使用@ComponentScan（扫描你的beans）和使用@Autowired（作为构造函数注入），它们做的非常好。

如果您按照上面的建议来构造您的代码（在根包中定位您的应用程序类），你需要加入@ComponentScan注解，并且不需要任何参数。你应用程序中所有包含如下注解的组件(@Component,@Service, @Repository, @Controller 等等)都会自动注册成为Spring Beans.

如下例子可以看到@Service Bean可以通过构造函数注入它所需要的RiskAssessor 实体：

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** DatabaseAccountService **implements** AccountService {

**private** **final** RiskAssessor riskAssessor;

@Autowired

**public** DatabaseAccountService(RiskAssessor riskAssessor) {

**this**.riskAssessor = riskAssessor;

}

}

如果bean里面只有一个构造函数，你可以省略掉@Autowired注解，代码如下所示：

@Service

**public** **class** DatabaseAccountService **implements** AccountService {

**private** **final** RiskAssessor riskAssessor;

**public** DatabaseAccountService(RiskAssessor riskAssessor) {

**this**.riskAssessor = riskAssessor;

}

}

提示：注意使用构造函数注入如何将riskAssessor字段标记为final，表明无法随后更改。

# 使用@SpringBootApplication注解

很多Spring Boot开发者喜欢在它们的apps使用自动注入，组件扫描和在它们的“application class”上定义配置。使用一个@SpringBootApplication注解可以实现三个特性，那是：

* @EnableAutoConfiguration:开启Spring Boot自动配合机制
* @ComponentScan: 在应用程序所在的包上启用@component扫描（请参阅最佳实践）
* @Configuration:允许注册特别的beans到上下文或者引入额外的配置类

@SpringBootApplication 注解等价于使用@Configuration、@EnableAutoConfiguration和@ComponentScan注解和它们的默认属性，例子如下所示：

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

注意：@SpringBootApplication提供了别名来为@EnableAutoConfiguration 和 @ComponentScan 定制属性。

注意：这些特性都不是强制的，你可以选择任何的特性来替换这个注解。例如，你可以在你的程序不使用自动扫描组件：

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.EnableAutoConfiguration;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.annotation.Import;

@Configuration

@EnableAutoConfiguration

@Import({ MyConfig.**class**, MyAnotherConfig.**class** })

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

在这个例子里，应用和其他的Spring Boot应用程序一样，只是不自动检查@Component组件，需要用@Import注解来显式指定需要引入的配置。

# 启动你的应用程序

将应用程序打包成jar并使用嵌入式HTTP服务器的最大优势之一是，您可以像其他任何应用程序一样运行应用程序。Debugging Spring Boot应用将会变得非常简单，你不需要任何特别的IDE插件和拓展。

注意：这部分仅仅是包含基于jar方式打包，如果你需要将你的应用打包成war文件，您应该参考您的服务器和IDE文档。

## 在IDE运行

你可以在你的IDE运行Spring Boot应用就跟运行一个简单的java应用程序一样。当然，首先你得引入的项目，导入步骤取决于您的IDE和构建系统。很多IDE支持直接导入MAVEN项目。例如，Eclipse你可以通过如下步骤引入Import… → Existing Maven Projects from the File menu。