如何使用 Maven 搭建 Spring Boot 父子项目?

> 大家好，我是磊磊落落，目前我在技术上主要：Java、Golang、架构设计、云原生和自动化测试。欢迎来我的博客（[leileiluoluo.com](http://cxyroad.com/ "https://leileiluoluo.com/posts/spring-boot-parent-child-projects-with-maven.html")）获取我的最近更新!

本文探索如何使用 Maven 来搭建 Spring Boot 父子项目，方便我们在搭建 Spring Boot 微服务时作参考。

使用 Maven 搭建 Spring Boot 微服务项目时，最直接的做法是在每个项目的`pom.xml`中直接引用 Spring Boot Starter 父项目`org.springframework.boot:spring-boot-starter-parent`，并配置各项依赖。

但有多个微服务项目时，若要升级它们各自`pom.xml`中的依赖会产生大量的重复工作。因此，若将这些公共依赖抽取到一个父项目中，而这些子项目都引用这个父项目，那么依赖可在父项目中统一管理；这样，子项目若需升级，只需升级一下引用的父项目版本，升级过程将会变得非常简单。

本文接下来即会创建一个示例 Spring Boot 父项目，并尝试在子项目中引用并使用它。

写作本文时，用到的 Java、Maven 和 Spring Boot 的版本分别为：

* Java: 1.8
* Maven: 3.9.0
* Spring Boot: 2.7.9

1 创建父项目

首先，开始搭建父项目`starter-parent`，父项目可包括多个子模块，本文的例子仅含`common-utils`一个子模块，该子模块用于编写项目用到的工具类。

下面使用`mvn archetype:generate`来生成项目的脚手架。

首先，使用如下命令生成`starter-parent`空项目，生成后删除不用的`src`文件夹，并修改`pom.xml`中的`<packaging>`为`pom`。

```
mvn archetype:generate  \
    -DgroupId=com.leileiluoluo \
    -DartifactId=starter-parent \
    -Dversion=1.0-SNAPSHOT \
    -DinteractiveMode=false
```

然后，进入`starter-parent`文件夹下，生成子模块`common-utils`，命令如下：

```
cd starter-parent/

mvn archetype:generate  \
    -DgroupId=com.leileiluoluo \
    -DartifactId=common-utils \
    -Dversion=1.0-SNAPSHOT \
    -DinteractiveMode=false
```

这样，项目脚手架就出来了，目录结构如下：

```
starter-parent
 |-- common-utils
 |    |-- src/main/java
 |    `-- pom.xml
 `-- pom.xml
```

修改下两个`pom.xml`文件，使配置更简单紧凑；并在`common-utils`下加一个工具类`DataUtil.java`。

修改后，项目目录结构如下：

```
starter-parent
 |-- common-utils
 |   |-- src/main/java
 |   |   `-- com/leileiluoluo/common/util/DataUtil.java
 |   `-- pom.xml
 `-- pom.xml
```

下面看一下该项目下几个文件的源码。

1.1 pom.xml

根 POM，注意`<parent>`引用了`spring-boot-starter-parent`；`<packaging>`类型为`pom`；`<dependencyManagement>`定义了所有的公共依赖；`<pluginManagement>`定义了所有的公共插件。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.9</version>
        <relativePath/>
    </parent>

    <groupId>com.leileiluoluo</groupId>
    <artifactId>starter-parent</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
```

```
<properties>
    <java.version>1.8</java.version>
    <spring.boot.version>2.7.9</spring.boot.version>
</properties>

<modules>
    <module>common-utils</module>
</modules>

<dependencyManagement>
    <dependencies>
        <!-- modules -->
        <dependency>
            <groupId>com.leileiluoluo</groupId>
            <artifactId>common-utils</artifactId>
            <version>${version}</version>
        </dependency>

        <!-- spring boot starters -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
            <version>${spring.boot.version}</version>
        </dependency>

        <!-- test -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <version>${spring.boot.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.9.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
```

```
            </plugin>
        </plugins>
    </pluginManagement>
</build>
</project>
```

1.2 common-utils/pom.xml

`common-utils`子模块 POM，注意`<groupId>`与`<parent>`中的一致，所以可以省略；`<dependency>`中的`<version>`已在父 POM 中声明，可以省略。

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
         xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>com.leileiluoluo</groupId>
        <artifactId>starter-parent</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <artifactId>common-utils</artifactId>

    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

1.3 common-utils 下的 DataUtil.java

`DataUtil.java`为`common-utils`子模块下的 Java 类，供后面的项目使用。

```
package com.leileiluoluo.common.util;

import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;

public class DateUtil {

    public static String getCurrentTimeStr() {
        DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        return ZonedDateTime.now().format(formatter);
    }

}
```

接下来尝试创建一个项目来引用这个父项目。尝试之前，使用 Maven 将该项目打包并安装到本地。

命令如下：

```
mvn clean install

...
[INFO] Reactor Summary for starter-parent 1.0-SNAPSHOT:
[INFO]
[INFO] starter-parent ..................................... SUCCESS [  0.170 s]
[INFO] common-utils ....................................... SUCCESS [  1.330 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.658 s
[INFO] Finished at: 2023-03-14T10:34:43+08:00
[INFO] ------------------------------------------------------------------------
```

2 使用父项目

下面创建子项目`app-child`来使用前面的父项目`starter-parent`。

创建后的子项目目录结构如下：

```
app-child
 |-- src/main/java
 |    `-- com/leileiluoluo/app/DemoApplication.java
 `-- pom.xml
```

下面接着看一下该项目下的几个文件，然后尝试启动项目并做测试。

2.1 pom.xml

该项目 POM，注意`<parent>`引用了前面搭建好的`starter-parent`；`<packaging>`采用默认类型`jar`；`<dependencies>`下引用的公共依赖均无需指定`<version>`；`plugins`下引用的公共插件也无需指定`<version>`。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.leileiluoluo</groupId>
        <artifactId>starter-parent</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <artifactId>app-child</artifactId>

    <properties>
```

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

    <dependencies>
        <dependency>
            <groupId>com.leileiluoluo</groupId>
            <artifactId>common-utils</artifactId>
        </dependency>

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

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

2.2 DemoApplication.java

仅写了一个 API `hello`，在其方法内调用了父项目模块`common-utils`中的`DateUtil.getCurrentTimeStr()`方法。

```
package com.leileiluoluo.app;

import com.leileiluoluo.common.util.DateUtil;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

    @GetMapping("/hello")
    public String hello(@RequestParam(value = "name", defaultValue =
"World") String name) {
        return String.format("Hello %s! %s", name,
DateUtil.getCurrentTimeStr());
    }

}
```

2.3 启动与测试

下面使用 Maven 将该项目打包，然后运行打包好的`jar`包，命令如下：

```
cd app-child/
mvn clean package

java -jar target/app-child-1.0-SNAPSHOT.jar
```

启动完成后，使用`curl`请求一下，会得到如下内容。

```
curl 'http://localhost:8080/hello?name=Larry'

Hello Larry! 2023-03-13 16:31:23
```

综上，本文探索了如何使用 Maven 来搭建 Spring Boot 父子项目，方便我们

在搭建 Spring Boot 微服务时作参考。本文中涉及的父项目`starter-parent`与子项目`app-child`的完整代码已托管至 [GitHub](http://cxyroad.com/”https://github.com/olzhy/java-exercises/tree/main/spring-boot-parent-child-maven-project”)，欢迎或 Fork。

> 参考资料
>
>
> [1] [Introduction to the POM | Maven – maven.apache.org](http://cxyroad.com/”https://maven.apache.org/guides/introduction/introduction-to-the-pom.html”)
>
>
> [2] [Introduction to the Dependency Mechanism | Maven – maven.apache.org](http://cxyroad.com/”https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html”)
>
>
> [3] [Guide to Working with Multiple Modules | Maven – maven.apache.org](http://cxyroad.com/”https://maven.apache.org/guides/mini/guide-multiple-modules.html”)
>
>
> [4] [Spring Boot | Spring – spring.io](http://cxyroad.com/”https://spring.io/projects/spring-boot”)
>
>
> [5] [Maven by Example: A Multi-Module Project | TheNEXUS – books.sonatype.com](http://cxyroad.com/”https://books.sonatype.com/mvnex-book/reference/multimodule.html”)
>
>
> [6] [Multi-Module Project with Maven | Baeldung – www.baeldung.com](http://cxyroad.com/”https://www.baeldung.com/maven-multi-module”)
>
>
> [7] [Spring Boot Multi-Module Maven Project | HowToDoInJava – howtodoinjava.com](http://cxyroad.com/”https://howtodoinjava.com/spring-boot2/sb-multi-module-maven-project/”)