30讲一个好的项目自动化应该是什么样子的

进入自动化这个模块，我准备从程序员的日常工作开始。介绍“迭代0”时，我提到构建脚本是项目准备的一个重要组成部分，但在那一讲中，我并没有具体说构建脚本长成什么样。

今天，我们以一个典型的 Java REST 服务为例，介绍一下最基本的构建脚本应该做到什么样子。这里我采用的 Java 技术中最为常见的 Spring Boot 作为基础框架，而构建工具，我选择了 Gradle。

估计很多 Java 程序员心中的第一个问题就是，为什么用 Gradle，而不是 Maven？Maven 明明是 Java 社区最经典的构建工具。答案是因为 Maven 不够灵活。

你可以回想一下，你有多少次用 Maven 实现过特定需求？估计大部分人的答案都是没有。随着持续集成、持续交付的兴起，构建脚本的订制能力会变得越来越重要，Maven 则表现得力有不逮。

其实，早在2012年，ThoughtWorks 技术雷达就将 Maven 放到了 **暂缓（HOLD）**里面，也就是说，能不用就不用。

为了配合这次的讲解，我写了一个 Demo，放在了 Github 上。它的功能非常简单：

- 通过向 /users POST 一个请求，实现用户注册；
- 访问 /users，查看已注册的用户。

如果方便的话，你最好把这个项目 clone 下来，以便参考。这里我主要是讲解自动化要做成什么样子，如果你想了解具体是怎么实现的，可以参考 Demo 里的代码。

好，我们开始！

基础准备

先把这个项目从 Github 上 clone 下来。

```
git clone  https://github.com/dreamhead/geektime-zero.git
```

然后，进入到项目所在的目录中。

```
cd geektime-zero
```

当你准备就绪，我们就开始进一步了解这个项目。

一般我们了解一个项目，都会用用一个 IDE 打开这个项目，这里我推荐使用 IntelliJ IDEA，这是目前行业中最好的Java IDE。自从它的社区版免费之后，它就成为了我向他人推荐的首选。

我知道，开发工具是除了程序设计语言之外，另外一个容易引起“宗教战争”的话题，如果你喜欢其他的 IDE，那就用你最喜欢的 IDE 打开好了，只不过，需要调整一下构建脚本中的配置。

怎么打开这个项目呢？我们先用 Gradle 命令生成一个 IDEA 工程。

```
./gradlew idea
```

这个命令会生成一个.ipr 文件，这就是 IDEA 的工程文件，用 IDEA 打开即可。

这里有两点需要说明一下。

第一，这里用的 gradlew，它是 Gradle 命令的一个封装，它会自动下载一个构建这个项目所需的Gradle，重点是通过这个命令锁定了 Gradle 的版本，避免因为构建脚本的差异，造成“你成功我失败”的情况。

第二，IDE 工程是生成的。很多人直觉的做法是用 IDE 直接打开。有一些团队的项目里有好多个构建文件，究竟用哪个打开，不去问人是根本不知道的，这对项目的新人是非常不友好的。

生成的做法与前面 Gradle 封装是类似的，它可以避免因为本地安装不同版本 IDE 造成各种问题。

另外，因为 IDE 的工程是生成的，如果项目里一旦增加了新的程序库依赖，你只需重新执行一次上面的命令就好了，现在的 IDE 都有很好的自动加载能力，当它检测到工程文件的变化，就会重新加载。

好，现在你可以用 IDE 打开，我们就可以进一步了解这个项目了。

初见项目

我们先来了解一点 Gradle 的配置文件，它也是我们做项目自动化的重点。

- build.gradle，它是 Gradle 的配置文件。因为 Gradle 是由 Groovy 编写而成，build.gradle 本质上就是一个 Groovy 的脚本，其中的配置就是 Groovy 代码，这也是 Gradle 能够灵活订制的基础。
- settings.gradle，这也是一个 Gradle 配置文件，用以支持多模块。如果说一个项目的每个模块都可以有一个 build.gradle，那整个项目只有一个 settings.gradle。

在 Gradle 里，许多能力都是以插件的形式提供的，比如，前面生成 IDEA 工程就是配置文件中的一句话。

```
apply plugin: 'idea'
```

所以，如果你是其他 IDE 的死忠粉，你可以把这句话，换成你喜欢的 IDE。

（注：这个项目采用 Lombok 简化代码，为了能让代码在你的 IntelliJ IDEA 编译运行，你可以安装 Lombok 插件，然后，在 “Build, Execution, Deployment”-> “Compiler” -> “Annotation Processors“”中，选中 Enable annotation processing）

好，有了基础知识之后，我们来了解一下代码组织。

首先是分模块。除非你的代码库规模非常小，否则，分模块几乎是一种必然。一种恰当的划分方式是根据业务划分代码。比如，把用户相关的内容放到一个模块里，把交易订单信息放到一个模块里，把物流信息放到另一个模块里。

如果你未来打算做微服务，那每一个模块就可以成为一个独立的服务。

在我们的项目里，我示例性地划分了两个模块：

- zero-identity，是用户信息的模块；
- zero-bootstrap，是多个模块打包成一个可部署应用的模块。

这两个模块的信息都配置在 settings.gradle 中。

```
include 'zero-bootstrap'
include 'zero-identity'
```

再来是目录结构。具体要怎么样组织代码，在 Java 世界里已经是一件约定俗成的事情了。

src/main/java 下放着你的源代码，src/main/resources 下放配置文件，src/test/java 放测试代码。这是约定优于配置（Convention over Configuration）思想的体现。如果你用的工具没有约定，你只能自己定好，让其他人遵守。

检查

在自动化过程中，一个最基本的工作是检查。检查的工作在我们的项目中通过一个 check 任务来执行。

```
./gradlew check
```

这个检查会检查什么呢？这取决于配置。在这个项目里，我们应用了 Java 插件，它就可以编译Java 文件，检查代码是否可以正常编译，运行测试，检查代码是否功能正常等等。但我要求更多。

讲“迭代0”时，我说过，最基本的代码风格检查要放在构建脚本中，这里我用了 CheckStyle 来做这件事。缺省情况下，你只要应用 Checkstyle 插件即可。

```
apply plugin: 'checkstyle'
```

在这个项目里，我做了一些订制，比如，指定某些文件可以不做检查。

```
style.excludePackages = [
]

style.excludeClasses = [
]
```

测试覆盖率也应该加入到构建脚本中，这里我用了 JaCoCo。同样，缺省情况下，只要应用 JaCoCo 插件即可。

```
apply plugin: 'jacoco'
```

我依然是做了一些订制，比如，生成结果的 HTML 报表，还有可以忽略某些文件不做检查。

```
coverage.excludePackages = [
]

coverage.excludeClasses = [
]
```

这里最特别的地方是，我将测试覆盖率固定在1.0，也就是100%的测试覆盖。这是我做新项目的缺省配置，也是我对团队的要求。

如果一个新项目，能把这几个检查都通过，腐坏的速度应该就不会那么快了。当然，你也可以根据自己的需要，添加更多的检查。

数据库迁移

讲“迭代0”时，我还提到了数据库迁移，也就是怎样修改数据库。在示例项目中，我选择的数据库迁移工具是 Flyway。

```
plugins {
    id "org.flywaydb.flyway" version "5.2.4"
}
```

下面先要做一些基本的配置，保证可以连接到数据库。（注：如果你想直接使用这里的配置，可以在本机的 MySQL 数据库上，创建一个 zero 的用户，密码是 geektime，然后，再创建一个 zero_test 的数据库。）

```
flyway {
    url = 'jdbc:mysql://localhost:3306/zero_test?useUnicode=true&characterEncoding=utf-8&useSSL=false'
    user = 'zero'
    password = 'geektime'
    locations = ["filesystem:$rootDir/gradle/config/migration"]
}
```

那修改数据库会怎么做呢？先添加一个数据库迁移文件，比如，在示例项目中，我创建一个迁移文件（gradle/config/migration/V2019.02.15.07.43__Create_user_table.sql），在其中创建了一个 User 表。

```
CREATE TABLE zero_users(
    id bigint(20) not null AUTO_INCREMENT,
    name varchar(100) not null unique,
    password varchar(100) not null,
    primary key(id)
);
```

这里的迁移文件版本，我选择了以时间戳的方式进行命名，还有一种方式是以版本号的方式，比如 V1、V2。

时间戳命名方式的好处是，不同的人可以同时开发，命名冲突的几率很小，而采用版本号命名的方式，命名冲突的概率会大一些。

添加好数据库迁移文件之后，只要执行下面这个命令就好：

```
./gradlew flywayMigrate
```

这样，对数据库的修改就在数据库里了，你可以打开数据库查看一下。

构建应用

做好了最基本的检查，数据库也准备就绪，接下来，我们就应该构建我们的应用了。

首先是生成构建产物，它只要一个命令。

```
./gradlew build
```

这个命令会在 zero-bootstrap/build/libs 下生成一个可执行 JAR 包，它就是我们最终的构建产物。此外，build 任务会依赖于 check 任务，也就是说，构建之前，会先对代码进行检查。

从前 Java 程序只是打出一个可部署的包，然后，部署到应用服务器上。感谢现在基础设施的进步，我们可以省去部署的环节，这个包本身就是一个可执行的。我们可以通过命令执行将 JAR 执行起来。

```
java -jar zero-bootstrap/build/libs/zero-bootstrap-*-boot.jar
```

在开发过程中，并不需要每次都将 JAR 包打出来，我们还可以直接通过 Gradle 命令将应用运行起来。

```
./gradlew bootRun
```

不过，我估计你更常用的方式是，在 IDE 中找到 Bootstrap 这个入口类，然后，直接运行它。

既然程序已经运行起来，我们不妨测试一下。我们通过一些工具，比如 Postman 或者 Curl，把下面的内容 POST 到 http://localhost:8080/users

```
{
 "username": "foo",
 "password": "bar"
}
```

然后，通过浏览器访问 http://localhost:8080/users
我们就可以看见我们刚刚注册的这个用户了。

总结时刻

总结一下今天的内容。今天我们通过一个具体的例子展示了一个最基本的项目自动化，包括了：

- 生成 IDE 工程；
- 编译；
- 打包；
- 运行测试；
- 代码风格检查；
- 测试覆盖率；
- 数据库迁移；
- 运行应用。

但这就是自动化的全部了吗？显然不是，我这里给出的只是一个最基本的示例。实际上，几乎每个重复的工作或是繁琐的工作，都应该自动化。我们不应该把时间和精力浪费在那些机器可以很好地替我们完成的工作上。

今天的基础设施已经让我们的自动化工作变得比以往容易了很多，比如，可执行 JAR 包就比从前部署到应用服务器上简化太多了。Gradle 也让订制构建脚本的难度降低了很多。

这里提到的项目自动化也是持续集成的基础，在持续集成服务上执行的命令，就应该是我们在构建脚本中写好的，比如：

```
./gradlew build
```

2011年，我在 InfoQ 上发表了一篇《软件开发地基》，讨论的就是一个项目的构建脚本应该是什么样子。虽然其中用到的工具今天已经不再流行，但一些基础内容今天看来，依然是有效的。如果有兴趣，你也可以看一下。

如果今天的内容你只能记住一件事，那请记住：**将你的工作过程自动化。**

最后，我想请你分享一下，在日常开发工作中，你还把哪些过程自动化了呢？欢迎在留言区写下你的想法。

感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给你的朋友。

精选留言

西西弗与卡夫卡

设想过这样的情景（还没实现，打算实践一把）：我们新招一名比较熟练的程序员，从TA入职拿到机器，到开发示意代码，再提交SCM，然后CI/CD，再发布到线上交付给用户，整个过程可以在入职当天的午饭之前完成。

这不光要求构建和集成自动化，甚至要求从入职开始的各个环节都能提前准备好，包括机器、开发环境、线上环境等，甚至连示范的需求都要能及时传递给TA。理想情况下，程序员只需要开发好程序，保证质量，提交到SCM即可，其他事情都应该交给机器。

要知道程序员都很贵，越早给用户交付价值越好

2019-03-20 00:20

作者回复

是这个意思，后面继续谈如何往线上送。

2019-03-20 08:14

246小言

老师，我有个疑问。公司最近有个技术老大说不准用lombok这个小工具，一定要我们手写set get？lombok真的那么差吗？

2019-03-23 08:25

hua168

老师，以后用idea导向建立Spring Boot，我是不是把默认的maven改为gradle?
自动化：gitlab CI/CD+jenkins的gradle+docker
其中docker是把脚本放在jenkins，根据tag，如果是稳定版本就用makeFile生成docker镜像
这样可以吧?

2019-03-20 16:22

作者回复

我的建议是用构建脚本生成 IDE 工程。

2019-03-21 22:58

孤星可

我一般会在需要专有的构建工具之上 再定义一个 Makrfile 封装 通用的 test build deploy 操作 统一各语言构建工具差异 比如 m

aven gradle npm

2019-03-20 09:59

作者回复

我也做过类似的事情，用 shell 脚本去封装。

2019-03-21 22:59

shniu

自动化在持续交付中得到了非常充分的体现，把频繁的打包、单测、集成测试、部分验收测试、镜像构建和发布、CI环境更新、服务可用性验证等过程全部流水线化，极大的提升了构建发布效率，当然自动化的意义不仅于此；同时，把一切自动化是思维的转变，高效工作的有利工具。

2019-03-20 09:35

作者回复

没错，我们下面就会谈到持续交付。

2019-03-21 23:00

shniu

郑老师是否可以分享一下自己在模块划分上面的经验呢?

2019-03-20 09:26

作者回复

《敏捷软件开发：原则、实践与模式》、《架构整洁之道》中都有关于模块划分的内容。

2019-03-22 20:27

hua168

idea没有直接集成gradle吧？我看maven就直接有，好不容易学完maven，就淘汰了

有gradle方面的书吗？idea方面的呢？看官方教程？

idea是不是用到什么功能再学？还是先看过大概，用到再仔细看？还是最好都看一遍？

目前idea只会一些基础的…

2019-03-20 08:25

作者回复

理解错了，IDEA 提供了很好的 Gradle 支持。我的建议是生成 IDE 工程，maven 同样适用。

我学 Gradle 和 IDEA 这种工具都是看官方文档。

IDEA 的学习，别的不说，先熟练适用快捷键。

2019-03-20 08:58

北天魔狼

老师，我想问下PHP有类似构建脚本的工具吗？还没开始学习JAVA

2019-03-20 06:46

作者回复

我并不擅长 PHP，你不妨用 PHP build tool 为关键词搜索一下。

2019-03-21 22:48

hua168

In china，很多都是用破解版的……比如网上一堆idea激活码

2019-03-20 02:23

作者回复

其实现在的开发模式，社区版足够用了，否则，就是开发做重了。

2019-03-20 08:13