Git、Maven 的使用规范

爱旅行项目整个开发过程使用 Git 进行项目代码的版本管理，爱旅行后端项目使用 Maven 进行项目构建及多模块管理。

1.1 Git 使用说明

1.1.1 Git 环境准备

爱旅行项目在开源中国进行代码托管 https://git.oschina.net/（团队成员须自行注册账号），接下来由 Team Leader 完成 Git 环境的相关准备工作。

1. 创建项目，如图 1、2 所示，注意：由于该项目采用前后端分离开发，故应创建相应的前端项目（itripfront）和后端项目（itripbackend），分开进行代码版本管理。

图 1

图 2

2. 创建开发小组，如图 3 所示，把 team 成员拉进项目中，并赋予相应权限，一般开发人员授予开发者权限。

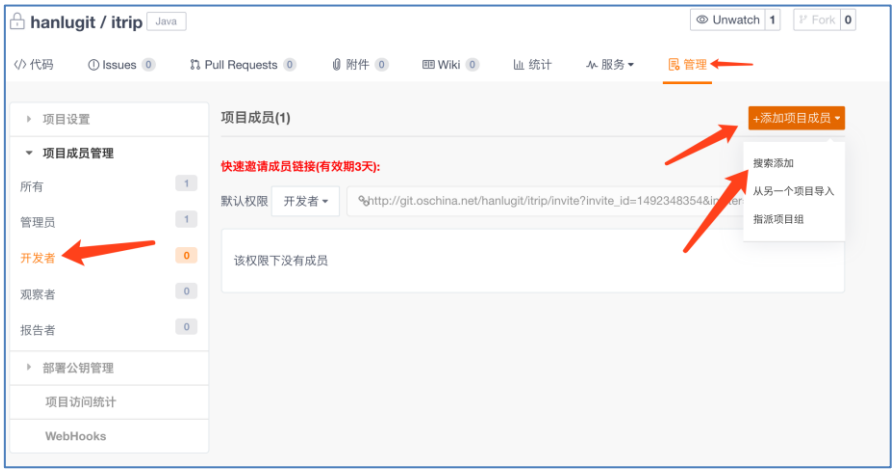

图 3

3. clone 项目 【前提：客户端已成功安装 git client 并保证可用】

首先复制 master 上项目路径到内存中，如图 4 所示：

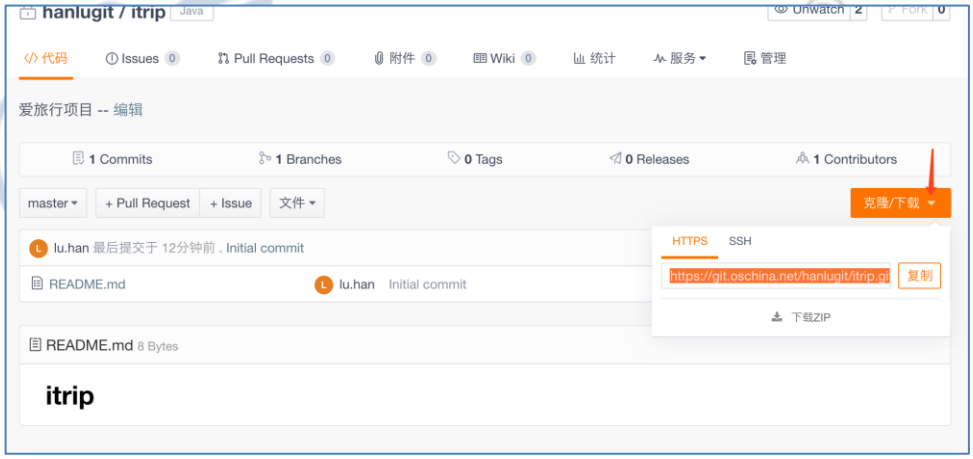

图 4

选择本地目录（e.g: D:\ideaworkspace），即：本地工程所在目录，然后进入 Git Bash，输入命令：

git clone https://git.oschina.net/hanlugit/itrip.git

clone 完成之后，可查看到本地已创建 itrip 项目，并且项目中只有一个 README.md 文件（注：项目开发者可以使用该文件进行项目记录等），通过命令：git branch 查看所在分支为 master

```
AppledeMacBook-Pro:backend apple$ cd itrip/
AppledeMacBook-Pro:itrip apple$ ls
README.md
AppledeMacBook-Pro:itrip apple$ git branch
* master
```

图 5

在本地完成项目框架搭建后，通过命令推送到远程主干（master）上。命令如下：

1> git add .

2> git commit -m ‘add pro framework’

3> git push

4. 创建分支 itripBranch1，并 push 到远程服务器

1> checkout 命令：git checkout -b itripBranch1 origin/master

```
AppledeMacBook-Pro:itrip apple$ git checkout -b itripBranch1 origin/master
Branch itripBranch1 set up to track remote branch master from origin.
Switched to a new branch 'itripBranch1'
AppledeMacBook-Pro:itrip apple$ git branch
* itripBranch1
  master
```

2> push 本地分支 itripBranch1 到远程服务器，完成远程分支的创建。

命令：git push origin itripBranch1

```
AppledeMacBook-Pro:itrip apple$ git push origin itripBranch1
Total 0 (delta 0), reused 0 (delta 0)
To https://git.oschina.net/hanlugit/itrip.git
 * [new branch]      itripBranch1 -> itripBranch1
```

注：分支创建除了通过命令之外，也可在页面上进行创建。

5. 完成项目框架的搭建，并上传项目代码到分支 itripBranch1 上

注意：.gitignore 文件对其所在的目录及所在目录的全部子目录均有效。通过将.gitignore 文件添加到仓库，其他开发者更新该文件到本地仓库，以共享同一套忽略规则。建议修改.gitignore 文件内容如图 5 所示：

```
**/target/
.settings/
.classpath
.mymetadata
.project
.gitconfig
.idea/
*.iml
**/classes/
```

图 5

6. 分支合并（注意：此操作由 Team Leader 完成）

一般情况下，当一个项目大版本完成，或者阶段完工，或者完整功能点开发完毕，需要部署到生产环境中，则会将开发中的分支（itripBranch1）合并到主干（master）上，注意：需合并的分支代码一定保证运行正常无报错，即：稳定运行版本。具体合并步骤如下：

1> 首先切换的 master 分支：git checkout master

2> 然后执行合并操作：git merge itripBranch1

3> 若有冲突，会提示，可调用 git status 查看冲突文件

3> 解决冲突，并调用 git add 或 git rm 将解决后的文件暂存

4> 所有冲突解决后，git commit 提交更改，git push

1.1.2 操作 Git

开发者操作 Git，可使用【推荐】Git Bash 等客户端工具或者使用 IDE（e.g：IntelliJ IDEA）集成工具进行代码的推送和拉取。具体操作过程如下：

1. git clone : 从 master clone，是必须的，如图 6：

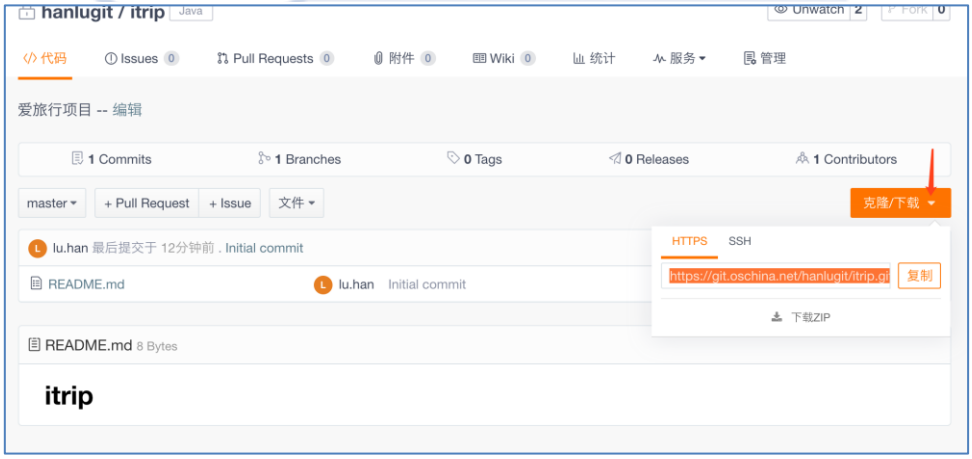

图 6

1> 命令：git clone https://git.oschina.net/hanlugit/itrip.git

2> 命令：ls 查看一下，会发现本地目录中已经有了 itrip

3> 命令：git branch 查看当前所在分支（目前应该是在 master 上）

2. checkout : 从项目管理者（Team Leader）处得知目前开发的分支，通过 checkout 远

程分支进行分支切换，即可在开发分支上工作。

命令：git checkout -b itripBranch01 remotes/origin/itripBranch01

通过 git branch 来查看当前所在分支（目前应该是在 itripBranch01 上），以后就在该分支上进行代码的开发，push 即可。

注意：开发者只在分支上进行开发，不操作 master！

1.1.3 Git 常用命令

1> git clone

2> git branch （注：作为初学者，应每次 push 之前，查看当前所在分支是否正确）

3> git status（注：每次 push 之前，要先执行该命令，确保提交正确）

4> git add .

5> git commit -m ‘填写 commit 信息描述’（注：添加 commit 信息必须完整正确）

6> git push

7> git checkout 分支名

8> git pull （注：每次 push 之前，要先 pull）

1.2 Maven 使用说明

1.2.1 Maven 环境准备

爱旅行项目使用 Maven 进行多 Module 管理， 将一个完整的项目拆成多个模块进行开发，而一些比较通用的模块（比如：itripbeans、itripdao、itriputils）就可以作为公用组件，在其他项目中直接依赖使用。

1. 新建 Maven 工程（itripbackend），该 Maven 工程只含一个 pom 文件，他是管理各个模块的父级 pom。

```
<groupId>cn.itrip</groupId>
<artifactId>itrip-project</artifactId>
<packaging>pom</packaging>
<version>1.0-SNAPSHOT</version>
<modules>
    <module>itripbiz</module>
    <module>itripsearch</module>
    <module>itripbeans</module>
    <module>itripdao</module>
    <module>itriputils</module>
    <module>itriptrade</module>
    <module>itripauth</module>
</modules>
```

图 7

2. 新建各个模块（itripbeans、itripdao、itriputils、itripbiz、itripsearch、itripauth、itriptrade），以 itripbiz 为例，如下图 8 所示：

```
<parent>
    <artifactId>itrip-project</artifactId>
    <groupId>cn.itrip</groupId>
    <version>1.0-SNAPSHOT</version>
    <relativePath>../pom.xml</relativePath>
</parent>
<modelVersion>4.0.0</modelVersion>
<artifactId>itrip-biz</artifactId>
<packaging>war</packaging>
```

图 8

注意：这里是需有 **parent** 节点的，说明是继承 itrip-project，其他跟普通的 pom 一样配置，另外由于 itripbiz 是 web 工程，需要打 war，与此相同的还有 itripsearch、itripauth、itriptrade 模块，但是 itripbeans、itripdao、itriputils 作为通用组件，需要打成 jar 包供其他项目模块依赖使用。当然上述的组件都是需要运行 install 命令（mvn clean install）才可以运行。

1.2.2 使用 Nexus 搭建 Maven 私服

Maven 有一个最重要的作用就是对 jar 文件的管理，爱旅行项目中我们使用 Nexus 搭建 Maven 私服（http://***.****.***/nexus/content/repositories/），整体项目的构建速度提高了很多。项目中所使用的 jar 包放到 Nexus 里管理，Git 只存储代码，可以省不少空间，对于 jar 的变更，也无须手动地替换 jar 包，只需要在项目 pom 文件里面进行修改就可以了，并且依赖包可以自动下载和更新，有效的节省开发时间。

注意：使用私服的原因是我们有自己的 jar 需要管理，若项目中自始至终都用不到自己

管理 jar，就不用搭建私服，若用到了比较特殊而又在中央仓库（推荐：http://repo.maven.apache.org/maven2 或者 https://mvnrepository.com）中没有的 jar，就需要使用私服进行上传。

在 pom 中指定中心仓库，如下图 9 所示：

```
<repositories>
    <repository>
        <id>nexus</id>
        <name>Nexus Repository</name>
        <url> http://[redacted]/nexus/content/groups/public/</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
    <repository>
        <id>snapshots</id>
        <url>http://[redacted]/nexus/content/repositories/snapshots</url>
    </repository>
</repositories>
```

图 9

1.3 补充说明

爱旅行项目中关于 Git 和 Maven 的使用，各中心可以根据具体情况，采用多种方式进行实施，比如：可以搭建 GitLab 服务器，或者在 GitHub 进行代码托管；Maven 的远程仓库可以使用 Apache 的中央仓库或者其他公共库，也可使用私服。只要达成项目目标即可。