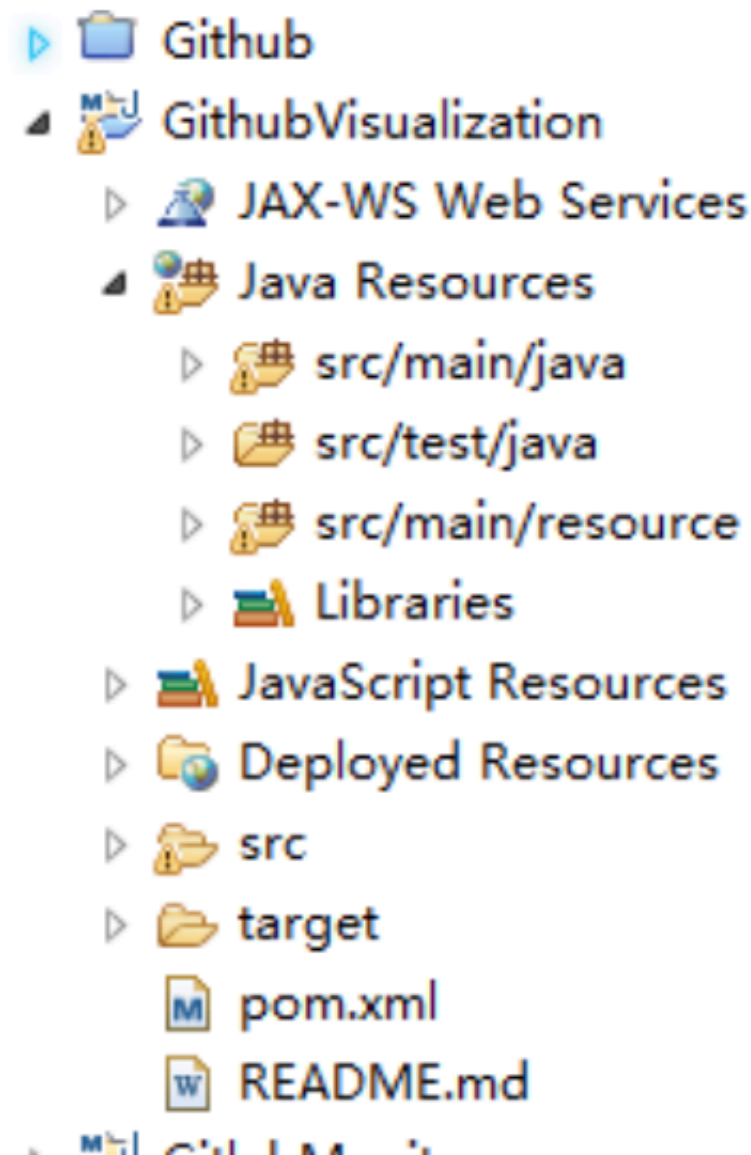


迭代三提交要求

- 项目规范
 - 提交项目必须为maven项目
 - 每次提交都必须编写与这次提交代码相关的单元测试
 - 在助教搭建的jenkins服务器上创建本小组的一个job，名称必须与gitlab上小组项目的名称一致，如141250053cseiiiAnyQuant / SEIII_AnyQuant
 - 配置完之后不要修改别的小组的配置，也不要修改jenkins系统配置，最好不要创建无用的job，测试的时候可以创建但测试完请删除
 - 提交代码的时候请只提交代码文件和配置文件，无用的数据文件，jar包，生成的class文件都不要提交。
- 提交内容
 - 在pom.xml文件中添加junit的依赖，可参考<https://github.com/gitminingOrg/GithubVisualization>下的pom文件。
 - 单元测试写在下图的src/test/java目录下，该目录是创建maven项目时存在的，若没有该目录可右击项目build path，点击libraries下的jre，选择default JRE即可。写在该目录下的单元测试将在构建的时候自动运行。



- 登陆jenkins服务器, <http://121.40.88.197:8080/>
- 点击新建, 创建一个maven项目, 名字为gitlab上项目名称

项目地址配置

☒ GitHub project
Project url

☐ 丢弃旧的构建 **千万不要勾这个**
☐ 参数化构建过程
☐ 关闭构建 (重新开启构建前不允许进行新的构建)
☐ 在必要的时候并发构建

高级项目选项 高级...

源码管理

☐ None
☐ CVS
☐ CVS Projectset
☒ Git

Repositories

Repository URL **地址同上**

Please enter Git repository.

Credentials Add **添加gitlab上的账号，如果上面url添加完报错，可先添加完账号再看看**

触发器配置，要求每次commit自动触发

构建触发器

☒ Build whenever a SNAPSHOT dependency is built
☐ Build after other projects are built
☐ Build periodically
☒ Build when a change is pushed to GitHub
☒ Build when a change is pushed to GitLab. GitLab CI Service URL: http://121.40.88.197:8080/project/testtt

Build on Merge Request Events ☒

Build on Push Events ☒ **勾选触发器**

Rebuild open Merge Requests

Enable [ci-skip] ☒

Set build description to build cause (eg. Merge request or Git Push) ☒

Add note with build status on merge requests ☒

Use GitLab CI features (GitLab 8.1 required!) ☐

Vote added to note with build status on merge requests ☒

Accept merge request on success ☐

☒ Allow all branches to trigger this job **?**
☐ Filter branches by name **?**
☐ Filter branches by regex **?**

Cobertura相关配置，重要！与你的测试覆盖率有关

Add pre-build step ▼

Build

Root POM ?

Goals and options ?

高级...

Post Steps

☐ Run only if build succeeds
 ☐ Run only if build succeeds or is unstable
 ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▼

构建设置

☐ E-mail Notification

构建后操作

Publish Cobertura Coverage Report

Cobertura xml report pattern

This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use `**/target/site/cobertura/coverage.xml`). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root

- 以上在jenkins上配置就完成了，然后还需要在gitlab上创建一个hook，以便每次提交能够在jenkins上自动进行构建。

Go to project

Project Settings

Deploy Keys

Web Hooks

Services

Protected Branches

Runners

Variables

Triggers

OwenChenx

Web hooks

Web hooks can be used for binding events when something is happening within the project.

URL

Trigger

☒ **Push events**
This url will be triggered by a push to the repository

☐ **Tag push events**
This url will be triggered when a new tag is pushed to the repository

☐ **Comments**
This url will be triggered when someone adds a comment

☐ **Issues events**
This url will be triggered when an issue is created/updated/merged

☐ **Merge Request events**
This url will be triggered when a merge request is created/updated/merged

☐ **Build events**
This url will be triggered when the build status changes

SSL verification ☒ **Enable SSL verification**

测试用例

- 单元测试代码统一基于junit4进行开发，请在pom文件中加入依赖：

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.8.2</version>
  <scope>test</scope>
</dependency>
```

- 同时，在中加入maven-surefire插件：

```
<build>
  <finalName>GitlabMonitor</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.8</version>
      <configuration>
        <includes>
          <!-- 包含所有后缀为Test.java的类 -->
          <include>**/*Test.java</include>
        </includes>
      </configuration>
    </plugin>
  </plugins>
</build>
```

- 测试文件包名为“开发包名Test”，文件名为“开发文件名Test”
- 单元测试要求逻辑和数据层的测试，代码覆盖率达到40%以上，展示层的测试不作硬性要求。

- 集成Cobertura插件

- 在pom文件中集成Cobertura插件，格式如下：

```
<reporting>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>cobertura-maven-plugin</artifactId>
      <version>2.5.2</version>
      <configuration>
        <formats>
          <!-- 这里很重要，一定要有xml -->
          <format>xml</format>
          <format>html</format>
        </formats>
      </configuration>
    </plugin>
  </plugins>
</reporting>
```

参考：<https://github.com/gitminingOrg/GitlabMonitor/blob/master/GitlabMonitor/pom.xml>

- 在jenkins配置页面，注意在build中添加cobertura相关配置：

Add pre-build step ▾

Build

Root POM ?

Goals and options ?

高级...

Post Steps

☐ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

构建设置

☐ E-mail Notification

构建后操作

☒ Publish Cobertura Coverage Report

Cobertura xml report pattern

This is a file name pattern that can be used to locate the cobertura xml report files (for example with Maven2 use ****/target/site/cobertura/coverage.xml**). The path is relative to the module root unless you have configured your SCM with multiple modules, in which case it is relative to the workspace root. Note that the module root is SCM-specific, and may not be the same as the workspace root

- 更多其他配置问题相关内容可见:

<http://www.ibm.com/developerworks/cn/java/j-lo-cobertura/index.html>

- 注: 在代码提交时, 请千万不要把target文件夹下生成的目标文件也一并提交, 大家可以在每一次jenkins构建完成时, 看到自己单元测试的代码覆盖率等数据, 这些数据将会纳入最终考核范围。