# 声纳扫描(sonarqube) 使用参考

说明

声纳扫描工具对于代码质量改进有重要的意义，本参考手册对声纳服务器的安装、本地使用FindBugs 静态扫描案例进行了简单的介绍。 如果想快速使用 FindBugs快速完成本地工程的代码质量扫描，请重点关注 《4 findBugs插件》章节。

| 作者 | 时间 | 备注 | 密级 |
| --- | --- | --- | --- |
| 姜鹏 | 2018.12.27 | V0.2版；新增FindBugs 插件使用。 | 内部使用 |
| | | | |
| | | | |
| | | | |

# 1 Docker 安装、运行sonarqube服务

Step1: 安装数据库

$docker run --name postgresqllatest -v /data/postgresql/:/var/lib/postgresql/data/ -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -p 54321:5432 -d docker.io/postgres:latest运行docker run 经常会报错，提示容器名称已经占用，需要移除掉先前的或者使用新的容器名。

$docker ps -a

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
3ad95254c368        postgres            "docker-entrypoint..."   6 minutes ago       Created                                         mypostgresql
94ca011a1e21        postgres            "docker-entrypoint..."   About an hour ago   Exited (0) 21 minutes ago                       postgresqldb
```

$docker stop ID

$docker rm -f ID

再次运行容器命令：

docker run --name postgresqllatest -v /data/postgresql/:/var/lib/postgresql/data/ -e POSTGRES_USER=postgres -e POSTGRES_PASSWORD=postgres -p 54321:5432 -d docker.io/postgres:latest

在WINDOWS客户端，连接云端通过docker启动的postgresql：

```
C:\Program Files\PostgreSQL\10\bin>psql -h 39.108.210.27 -p 54321 -U postgres
psql (10.1, 服务器 11.0 (Debian 11.0-1.pgdg90+2))
WARNING: psql major version 10, server major version 11.
         Some psql features might not work.
输入 "help" 来获取帮助信息.

postgres=#
```

CREATE USER sonar WITH PASSWORD 'sonar';

Step2:

$docker run -d --name mysonarqube --link postgresqllatest -v /data/sonarqube/:/var/lib/sonarqube/data/ -p 9001:9000  -e SONARQUBE_JDBC_URL=jdbc:postgresql://39.108.210.27:54321/sonar docker.io/sonarqube:latest

清理此容器的网络占用

$docker network disconnect --force bridge mysonarqube

查看是否还有同名容器占用

$docker network inspect mysonarqube

查看容器日志：

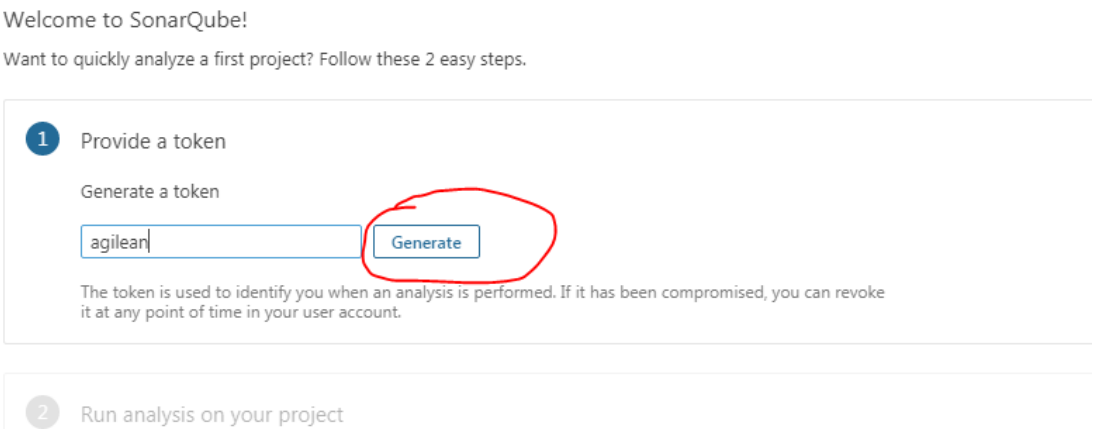
$docker logs -f -t --tail 600 mysonarqube

```
2018-10-26T05:04:01.139307000Z 2018.10.26 05:04:01 INFO  ce[][o.sonar.db.Database] Create JDBC data source for jdbc:postgresql://39.108.210.27:54321/sonar
2018-10-26T05:04:02.658058000Z 2018.10.26 05:04:02 INFO  ce[][o.s.s.p.ServerFileSystemImpl] SonarQube home: /opt/sonarqube
2018-10-26T05:04:02.832400000Z 2018.10.26 05:04:02 INFO  ce[][o.s.c.c.CePluginRepository] Load plugins
2018-10-26T05:04:03.643697000Z 2018.10.26 05:04:03 INFO  ce[][o.s.c.q.PurgeCeActivities] Delete the Compute Engine tasks created before 1524978243642
2018-10-26T05:04:03.657381000Z 2018.10.26 05:04:03 INFO  ce[][o.s.c.q.PurgeCeActivities] Delete the Scanner contexts tasks created before 1538111043656
2018-10-26T05:04:03.690125000Z 2018.10.26 05:04:03 INFO  ce[][o.s.ce.app.CeServer] Compute Engine is operational
2018-10-26T05:04:04.142011000Z 2018.10.26 05:04:04 INFO  app[][o.s.a.SchedulerImpl] Process[ce] is up
2018-10-26T05:04:04.142274000Z 2018.10.26 05:04:04 INFO  app[][o.s.a.SchedulerImpl] SonarQube is up
```

# 2 maven将项目提交到远程声纳服务器扫描

浏览SONAR服务：

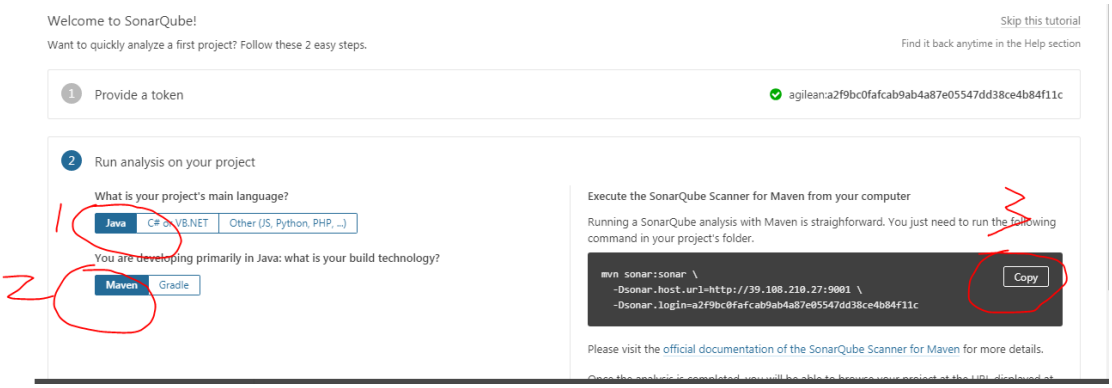
http://39.108.210.27:9001/projects




生成私有的Token.

**a2f9bc0fafcab9ab4a87e05547dd38ce4b84f11c**

选择分析的语言和构建：

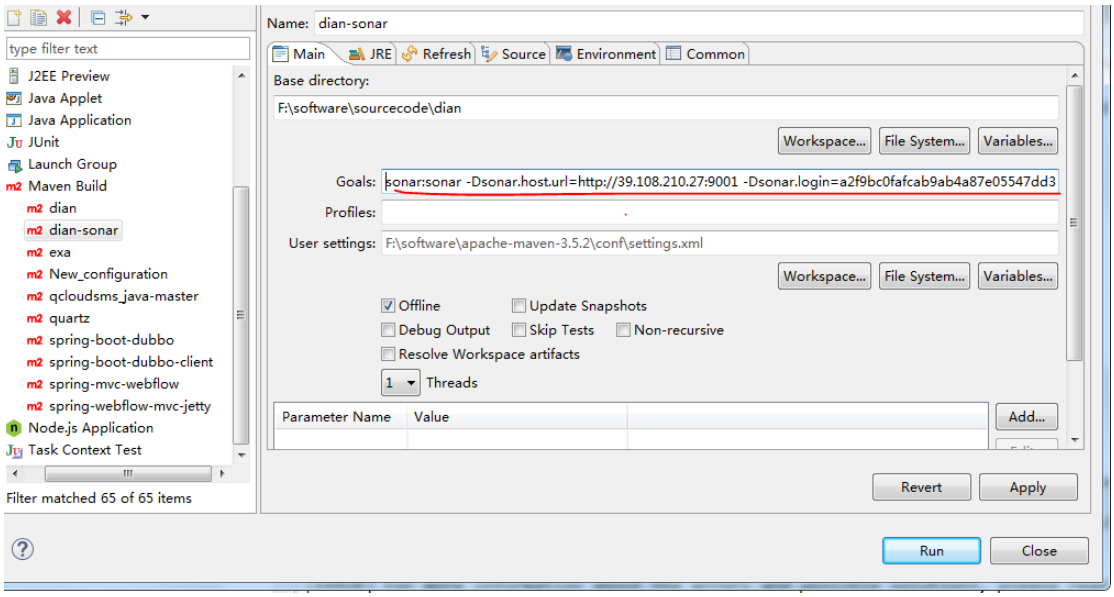



mvn sonar:sonar \

-Dsonar.host.url=http://39.108.210.27:9001 \

-Dsonar.login=a2f9bc0fafcab9ab4a87e05547dd38ce4b84f11c

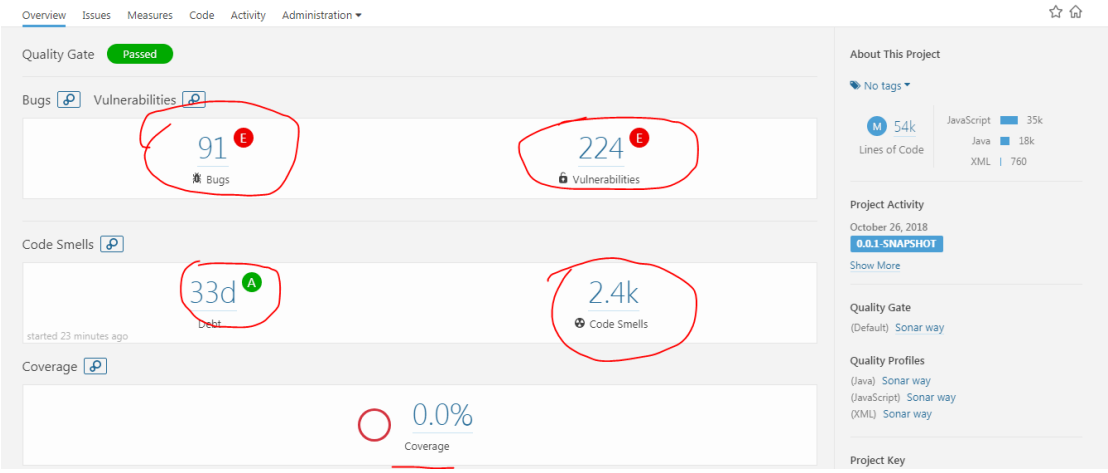
直接在ECLIPSE上对我的项目进行质量扫描：



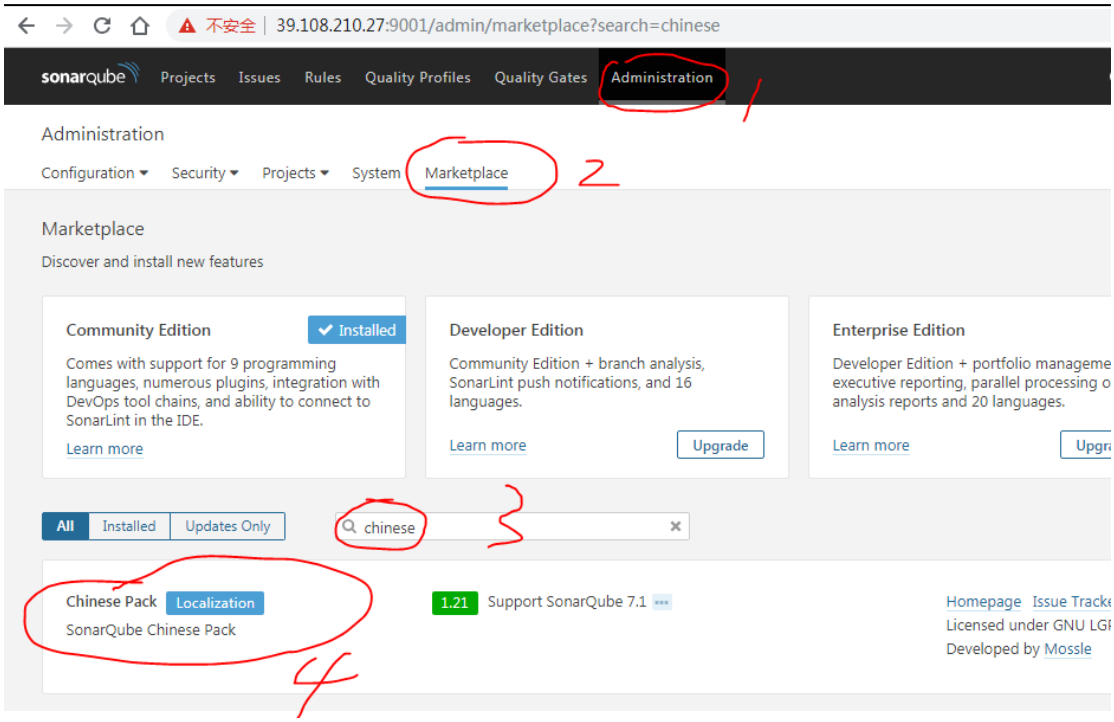

运行完后，直接形成报告。

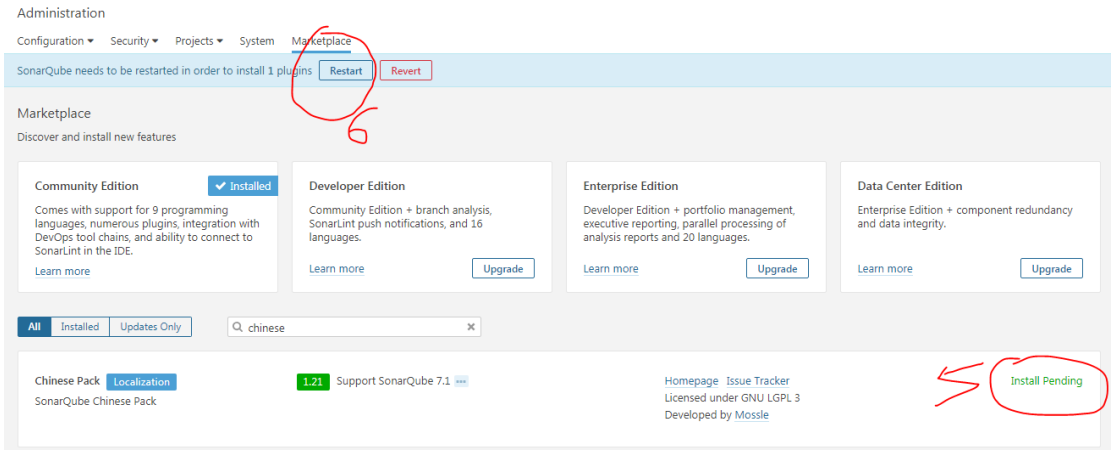
```
WARNING] Invalid character encountered in file F:/software/sourcecode/dian/src/main/webapp/js/jquery.fancybox.min.js at line
INFO] Analysis report generated in 4350ms, dir size=6 MB
INFO] Analysis reports compressed in 9291ms, zip size=2 MB
INFO] Analysis report uploaded in 27093ms
INFO] ANALYSIS SUCCESSFUL, you can browse http://39.108.210.27:9001/dashboard/index/com.hanniu.dian:dian
INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO] More about the report processing at http://39.108.210.27:9001/api/ce/task?id=AWau4EJ_Hqs0ObEGziSv
INFO] Task total time: 5:23.818 s
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time: 06:15 min
INFO] Finished at: 2018-10-26T13:37:27+08:00
INFO] Final Memory: 18M/247M
INFO] ------------------------------------------------------------------------
```

报告如下：

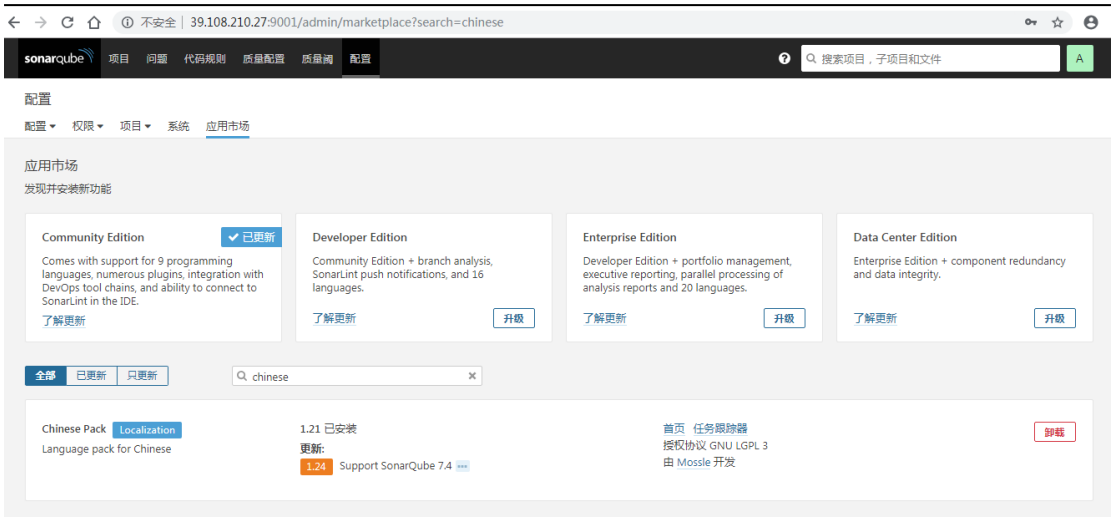



# 3 sonarqube 汉化安装


不安全 | 39.108.210.27:9001/admin/marketplace?search=chinese
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
1
Administration
Configuration
Security
Projects
System
Marketplace
2
Marketplace
Discover and install new features
Community Edition
Installed
Comes with support for 9 programming languages, numerous plugins, integration with DevOps tool chains, and ability to connect to SonarLint in the IDE.
Learn more
Developer Edition
Community Edition + branch analysis, SonarLint push notifications, and 16 languages.
Learn more
Upgrade
Enterprise Edition
Developer Edition + portfolio manageme
executive reporting, parallel processing o
analysis reports and 20 languages.
Learn more
All
Installed
Updates Only
chinese
3
Chinese Pack
Localization
SonarQube Chinese Pack
4
1.21
Support SonarQube 7.1
Homepage
Issue Tracke
Licensed under GNU LGI
Developed by Mossle


Administration
Configuration
Security
Projects
System
Marketplace
SonarQube needs to be restarted in order to install 1 plugins
Restart
Revert
6
Marketplace
Discover and install new features
Community Edition
Installed
Comes with support for 9 programming languages, numerous plugins, integration with DevOps tool chains, and ability to connect to SonarLint in the IDE.
Learn more
Developer Edition
Community Edition + branch analysis, SonarLint push notifications, and 16 languages.
Learn more
Upgrade
Enterprise Edition
Developer Edition + portfolio management, executive reporting, parallel processing of analysis reports and 20 languages.
Learn more
Upgrade
Data Center Edition
Enterprise Edition + component redundancy and data integrity.
Learn more
Upgrade
All
Installed
Updates Only
chinese
Chinese Pack
Localization
SonarQube Chinese Pack
1.21
Support SonarQube 7.1
Homepage
Issue Tracker
Licensed under GNU LGPL 3
Developed by Mossle
5
Install Pending

重启后登陆（）汉化完成。




# 4 findBugs 插件

前面章节介绍了SONAR 服务器的安装， 但实际上将本地的项目提交到远程SONAR服务器上将耗费比较长的时间。sonarqube 的 findbugs 插件让本地的静态扫描成为了可能。可以通过多种途径利用findbugs 插件实现代码静态扫描。下面主要介绍eclipse findbugs 插件和maven 插件两种方式。

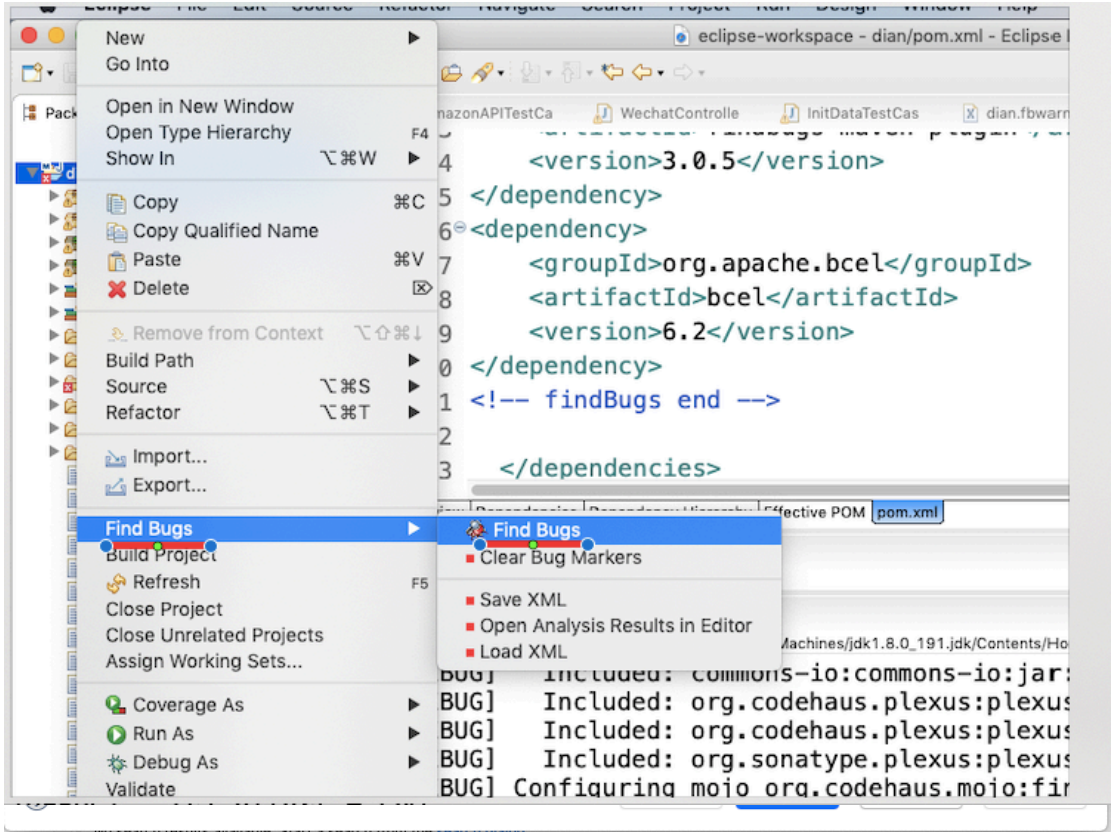
## 4.1 Eclipse findbugs 插件

### 4.1.1 Install

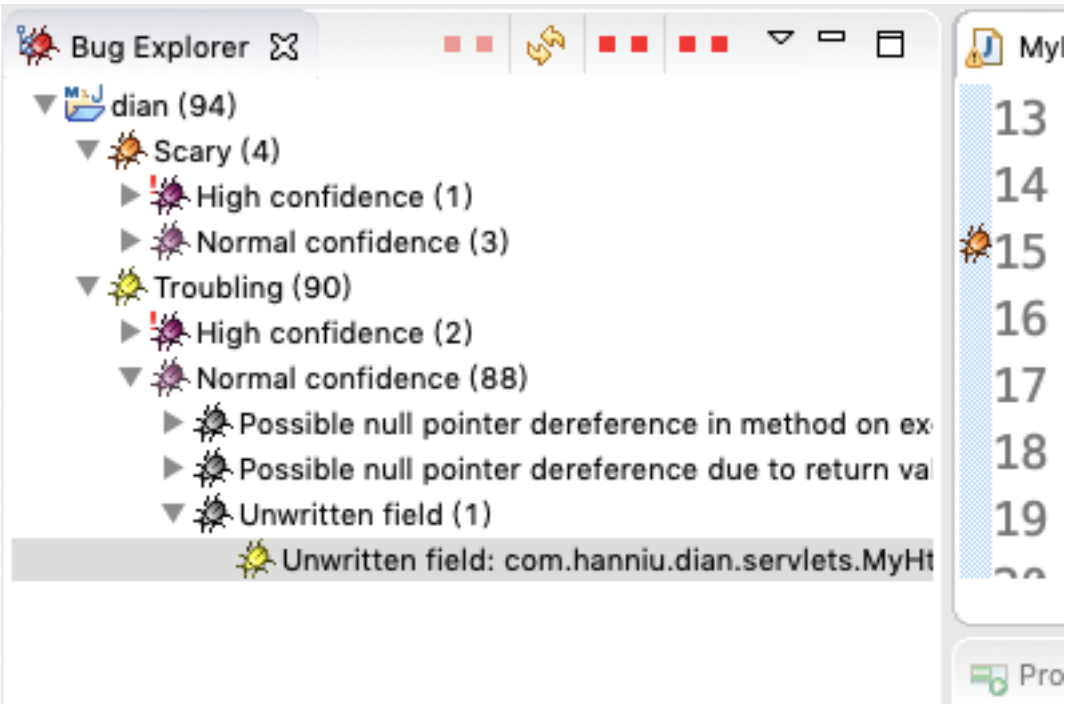
插安装URL： http://findbugs.cs.umd.edu/eclipse

### 4.1.2 执行findbugs 静态扫描

鼠标右键/Find Bugs/Find Bugs 即可。 如下图：



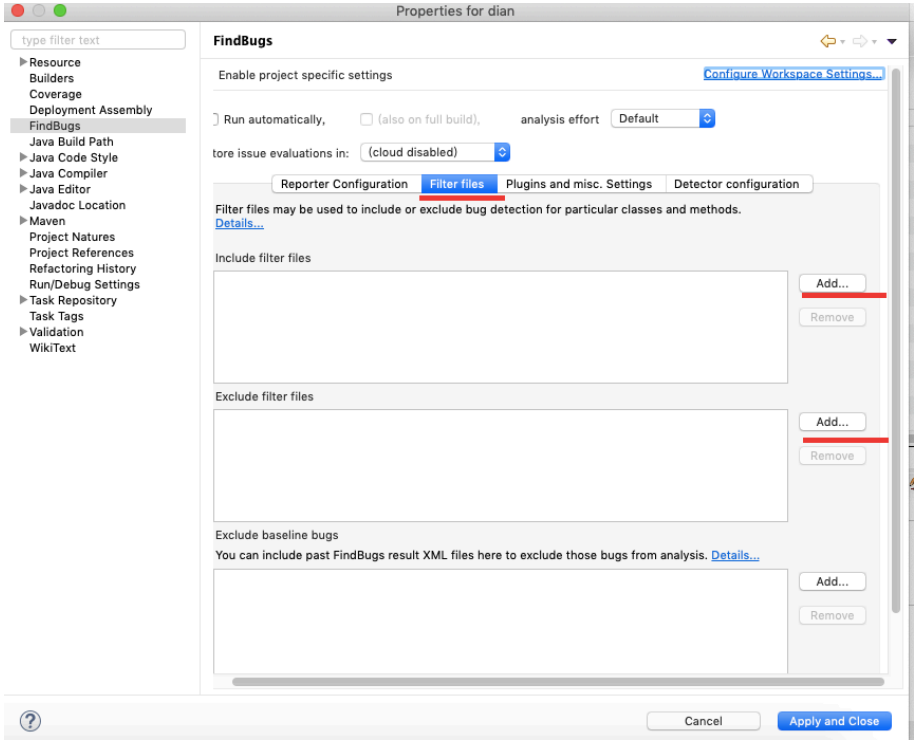

扫描后，打开Findbugs 视图：



可以点开任意一个BUG 条目前往查看。

## 4.1.3 新增过滤规则

如下图， 可以点击 “Exclude Filter Files ” 按钮 导入多个过滤规则文件。



## 4.2 Maven FindBugs 插件

第一步：需要在 pom.xml 中加入maven FindBugs 插件的依赖。 并将过滤的规则文件配置好。插件内容如下：

```
<plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>findbugs-maven-plugin</artifactId>
            <version>3.0.4</version>
            <configuration>
                <!-- 设置分析工作的等级，可以为Min、Default和Max -->
                <effort>Low</effort>
                <!-- Low、Medium和High (Low最严格) -->
                <threshold>Medium</threshold>
                <failOnError>true</failOnError>
                <includeTests>true</includeTests>
                <!--findbugs需要忽略的错误的配置文件-->
                <excludeFilterFile>/Users/source_code/dian/src/main/
resources/findBugs-filter.xml</excludeFilterFile>
                <findbugsXmlOutput>true</findbugsXmlOutput>
                <findbugsXmlOutputDirectory>target/site</
findbugsXmlOutputDirectory>
            </configuration>
            <executions>
                <execution>
                    <id>run-findbugs</id>
                    <phase>compile</phase>
                    <goals>




                        <goal>check</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
```

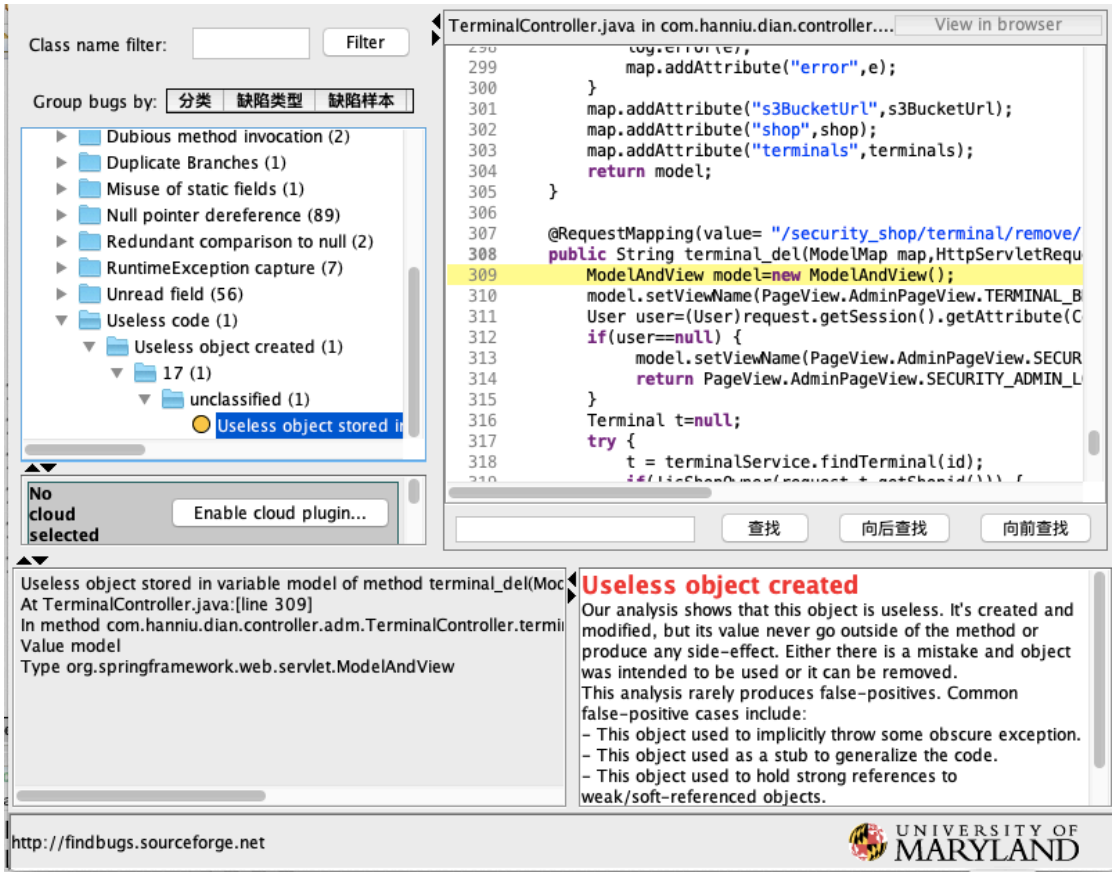
第二步： 定义好需要过滤规则的文件。

如下图： 所有的 main方法不需要扫描，所有的xml 文件， 所有以Impl.java, 包 com.hanniu.dian.online.test 包中的文件都需要过滤。规则参照下面：

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <FindBugsFilter>
3    <Match>
4        <Method name="main"/>
5   </Match>
6  <Match>
7    <Source name="~.*\.xml"/>
8  </Match>
9  <Match>
10   <Source name="~.*Impl\.java"/>
11 </Match>
12 <Match>
13 <Package name="com\.hanniu\.online\.test"/>
14 </Match>
15 </FindBugsFilter>
```

第三步： 执行 mvn findbugs:gui 命令：

运行命令后，会弹出一个FindBug GUI 界面：



## 4.3.2 过滤标签介绍

- <Bug> 设置警告的类型 (CORRECTNESS, MT_CORRECTNESS, BAD_PRACTICICE, PERFORMANCE, STYLE)
- <Confidence>设置过滤等级(1 : high-confidence warnings; 2 :normal-confidence warnings; 3 :low-confidence warnings)
- <Package>设置过滤包名
- <Class>设置过滤的类名
- <Source>设置过滤的源文件名
- <Method>设置过滤的方法名
- 过滤组合操作符：
  <Or>或
  <And>与
  <Not>非

## 4.3.3 Bug 类别解释

- Correctness bug

Probable bug - an apparent coding mistake resulting in code that was probably not what the developer intended. We strive for a low false positive rate.

- Bad Practice

Violations of recommended and essential coding practice. Examples include hash code and equals problems, cloneable idiom, dropped exceptions, serializable problems, and misuse of finalize. We strive to make this analysis accurate, although some groups may not care about some of the bad practices.

- Dodgy

Code that is confusing, anomalous, or written in a way that leads itself to errors. Examples include dead local stores, switch fall through, unconfirmed casts, and redundant null check of value known to be null. More false positives accepted. In previous versions of FindBugs, this category was known as Style.

# 5自定义规则

https://docs.sonarqube.org/latest/user-guide/built-in-rule-tags/

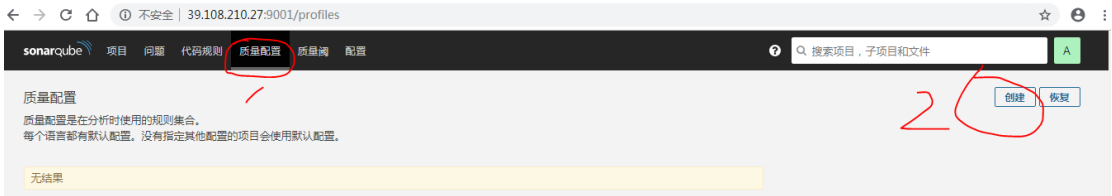
## 5.1 安装CheckStyle插件

官网：http://checkstyle.sourceforge.net/

## 5.2 配置自定义的CheckStyle代码规则

### 5.2.1使用CheckStyle代码规则配置文件

注意：这种方法只有新建一个质量配置时才能用，质量配置创建好后，就不能利用配置文件来配置代码规则了

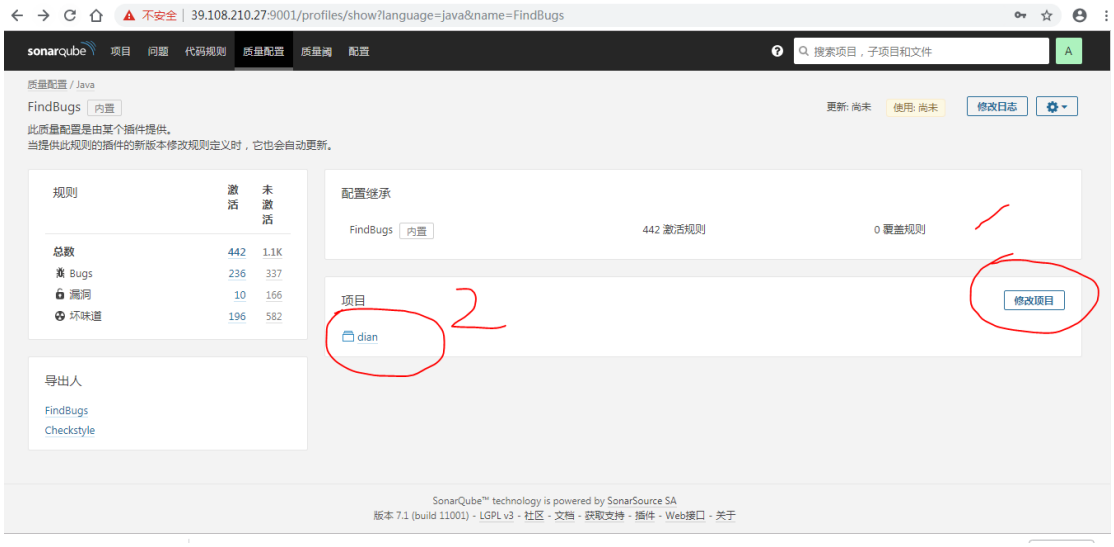


```
<module name="Checker">

    <property name="charset" value="UTF-8" />
    <property name="severity" value="warning" />
    <property name="fileExtensions" value="java, properties, xml" />


    <!-- 检查文件的长度（行） default max=2000 -->
    <module name="FileLength">
        <property name="max" value="2000" />
    </module>

    <module name="TreeWalker">

        <!-- ====================命名规范==================== -->

        <!-- 包名的检查（只允许小写字母） -->
        <module name="PackageName">
            <property name="format" value="^[a-z]+(\.[a-z][a-z0-9]*)*$" />
        </module>
```

上述方法不推荐。

## 5.2.2 启用SonarQube中CheckStyle相关代码规

让项目dian使用findBugs规则：




CheckStyle: 官网：

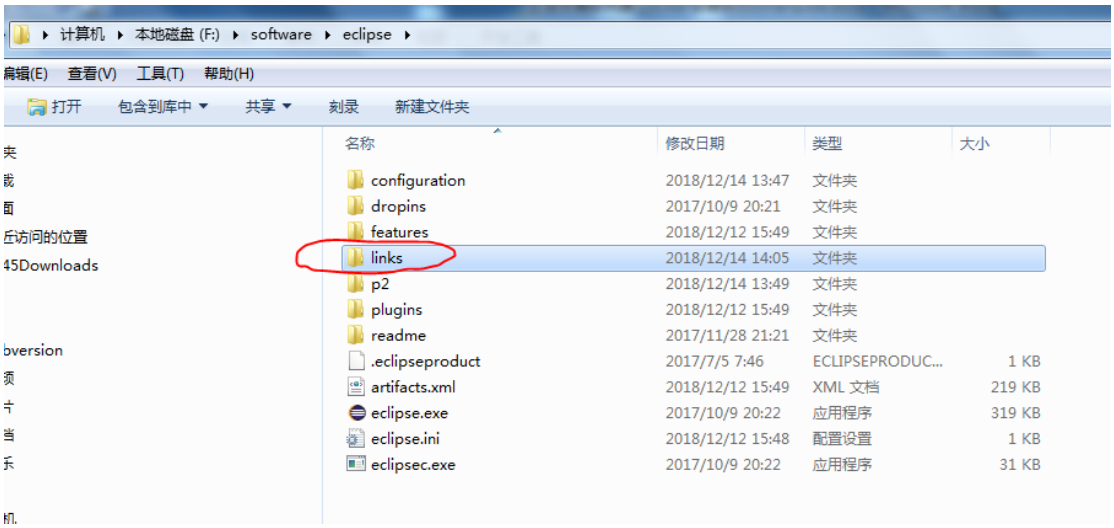
http://checkstyle.sourceforge.net/

配置checkstyle插件进行代码检查

下载插件：

https://sourceforge.net/projects/eclipse-cs/files/Eclipse%20Checkstyle%20Plug-in/

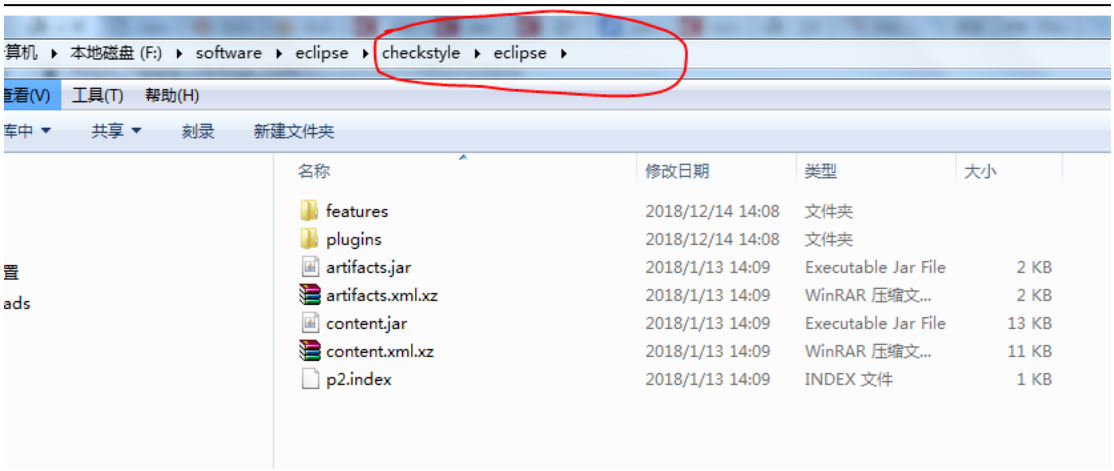
新建links



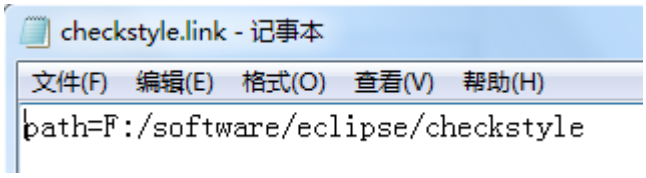

将下载的解压插件（含 ）拷贝至F:/softwares/eclipse/checkstye/

F:/softwares/eclipse/checkstye/eclipse/plugins

F:/softwares/eclipse/checkstye/eclipse/features




新建checkstyle.link 文件，内容如下：




# 6 如何调整远程sonar 服务器的扫描规则

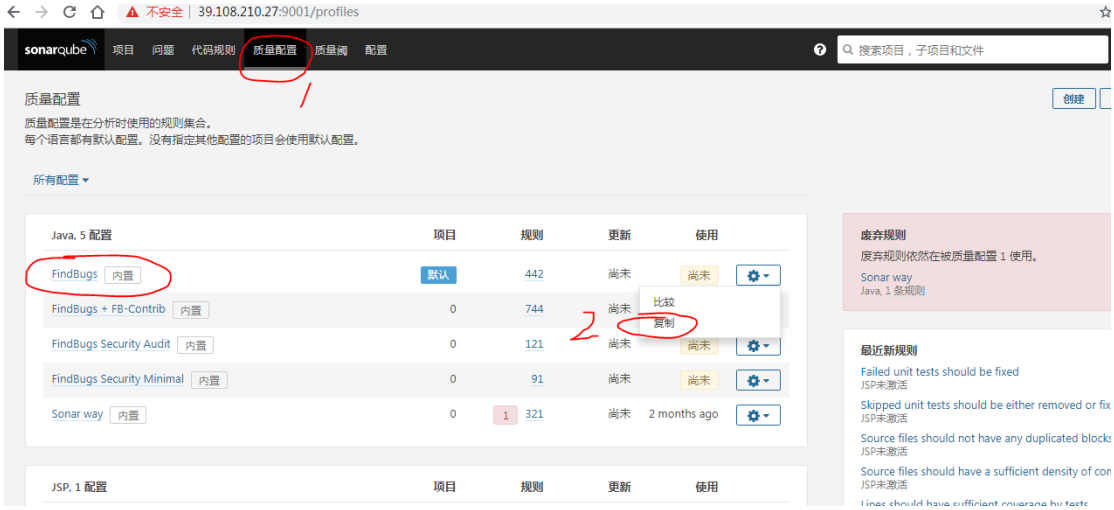
## 6.1 自定义

需要按照sonarqube API代码实现。（知识量大，下一版本介绍）

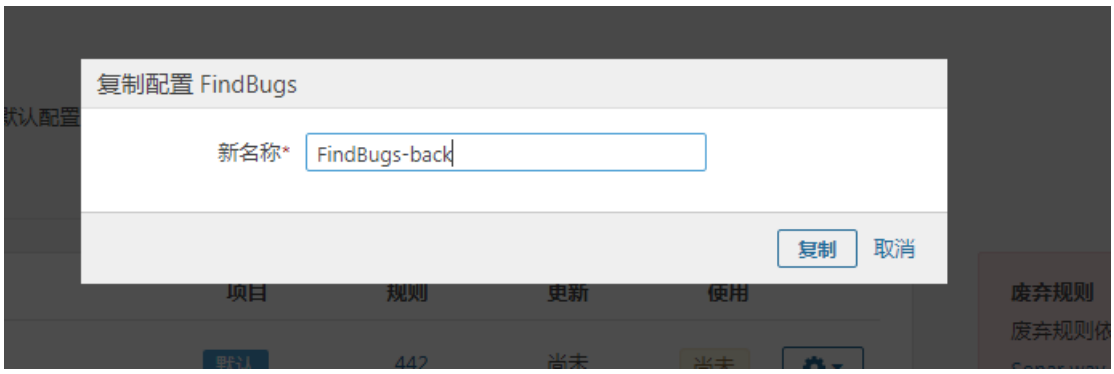
## 6.2 服务器端修改模板

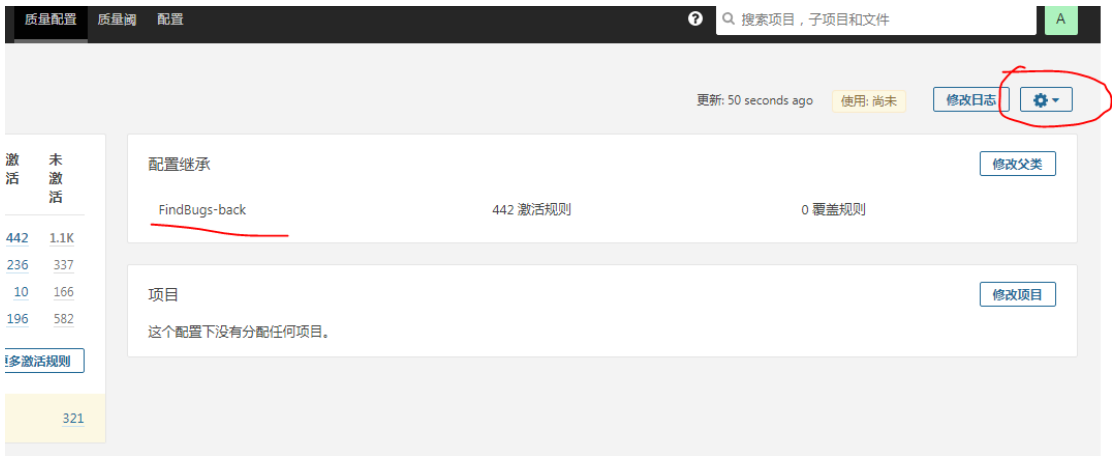
➢ 第一步： 导出模板xml文件

管理员账号登陆 sonarqube, 点击"质量配置"菜单，显示所有插件配置，如下图： 我们觉得findBugs 插件规则太多， 可以点击最右边的  键， 选择"复制"。




取名为：findBugs-back




点击"备份"，如下图。


点击"备份"后， 可以下载 findBugs 的xml格式的模板文件。文件格式如下：

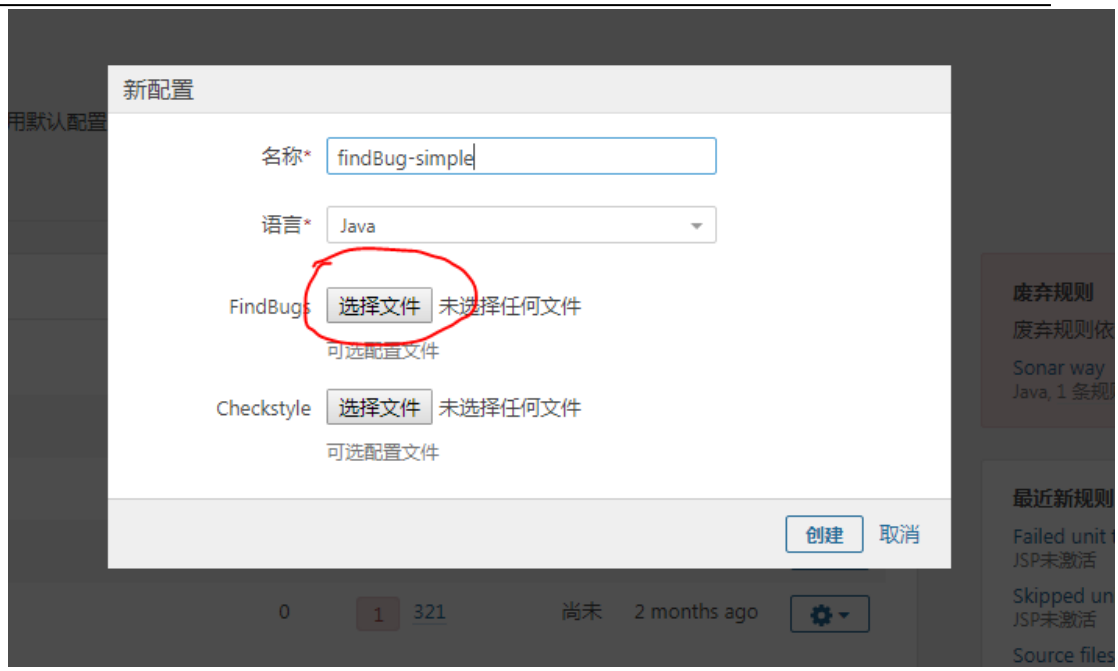
```
▼<profile>
  <name>FindBugs-new</name>
  <language>java</language>
 ▼<rules>
  ▼<rule>
     <repositoryKey>findbugs</repositoryKey>
     <key>AM_CREATES_EMPTY_JAR_FILE_ENTRY</key>
     <priority>MAJOR</priority>
     <parameters/>
   </rule>
  ▼<rule>
     <repositoryKey>findbugs</repositoryKey>
     <key>AM_CREATES_EMPTY_ZIP_FILE_ENTRY</key>
     <priority>MAJOR</priority>
     <parameters/>
   </rule>
  ▼<rule>
     <repositoryKey>findbugs</repositoryKey>
     <key>AT_OPERATION_SEQUENCE_ON_CONCURRENT_ABSTRACTION</key>
     <priority>MAJOR</priority>
     <parameters/>
   </rule>
  ▼<rule>
     <repositoryKey>findbugs</repositoryKey>
     <key>BAC_BAD_APPLET_CONSTRUCTOR</key>
     <priority>MAJOR</priority>
     <parameters/>
   </rule>
  ▼<rule>
     <repositoryKey>findbugs</repositoryKey>
     <key>BC_BAD_CAST_TO_ABSTRACT_COLLECTION</key>
     <priority>INFO</priority>
     <parameters/>
   </rule>
```


➢ 第二步：修改模板文件：

findBugs 自带400多种规则，实际用到的不多，可以根据实际情况删减部分规则，直接修改XML模板文件并保存。

➢ 第三步：新建新的模板文件

点击"新建"，取名为：findBug-simple




点击"选择文件"，将前面修改的模板文件上传：

如下图所示：新的规则已经生效。

# 7 sonar rules 整理

参考:

https://rules.sonarsource.com/java/RSPEC-4434

## 7.1 LDAP 初始化不允许反系列化

JNDI supports the deserialization of objects from LDAP directories, which is fundamentally insecure and can lead to remote code execution.

This rule raises an issue when an LDAP search query is executed with SearchControls configured to allow deserialization.

**Noncompliant Code Example**

```
DirContext ctx = new InitialDirContext();
// ...
ctx.search(query, filter,
        new SearchControls(scope, countLimit, timeLimit, attributes,
            true, // Noncompliant; allows deserialization
            deref));
```

**Compliant Solution**

```
DirContext ctx = new InitialDirContext();
// ...
ctx.search(query, filter,
        new SearchControls(scope, countLimit, timeLimit, attributes,
            false,
            deref));
```

## 7.2 Cryptographic keys should not be too short

https://rules.sonarsource.com/java/RSPEC-4426

When generating cryptographic keys (or key pairs), it is important to use a key length that provides enough entropy against brute-force attacks. For the Blowfish algorithm the key should be at least 128 bits long, while for the RSA algorithm it should be at least 2048 bits long.

This rule raises an issue when a Blowfish key generator or RSA key-pair generator is initialized with too small a length parameter.

**Noncompliant Code Example**

```
KeyGenerator keyGen = KeyGenerator.getInstance("Blowfish");
keyGen.init(64); // Noncompliant

KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
keyPairGen.initialize(512); // Noncompliant
```

**Compliant Solution**

```
KeyGenerator keyGen = KeyGenerator.getInstance("Blowfish");
keyGen.init(128);

KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("RSA");
keyPairGen.initialize(2048);
```

## 7.3 SQL注入

```
public boolean authenticate(javax.servlet.http.HttpServletRequest request, java.sql.Connection connection) throws SQLException {
  String user = request.getParameter("user");
  String pass = request.getParameter("pass");

  String query = "SELECT * FROM users WHERE user = '" + user + "' AND pass = '" + pass + "'"; // Unsafe

  // If the special value "foo' OR 1=1 --" is passed as either the user or pass, authentication is bypassed
  // Indeed, if it is passed as a user, the query becomes:
  // SELECT * FROM users WHERE user = 'foo' OR 1=1 --' AND pass = '...'
  // As '--' is the comment till end of line syntax in SQL, this is equivalent to:
  // SELECT * FROM users WHERE user = 'foo' OR 1=1
  // which is equivalent to:
  // SELECT * FROM users WHERE 1=1
  // which is equivalent to:
  // SELECT * FROM users

  java.sql.Statement statement = connection.createStatement();
  java.sql.ResultSet resultSet = statement.executeQuery(query); // Noncompliant
  return resultSet.next();
}
```

正确写法:

```
public boolean authenticate(javax.servlet.http.HttpServletRequest request, java.sql.Connection connection) throws SQLException {
  String user = request.getParameter("user");
  String pass = request.getParameter("pass");

  String query = "SELECT * FROM users WHERE user = ? AND pass = ?"; // Safe

  java.sql.PreparedStatement statement = connection.prepareStatement(query);
  statement.setString(1, user); // Will be properly escaped
  statement.setString(2, pass);
  java.sql.ResultSet resultSet = statement.executeQuery();
  return resultSet.next();
}
```

## 7.4 不再安全的加密方式

According to the US National Institute of Standards and Technology (NIST), the Data Encryption Standard (DES) is no longer considered secure:

**Noncompliant Code Example**

```
Cipher c = Cipher.getInstance("DESede/ECB/PKCS5Padding");
```

**Compliant Solution**

```
Cipher c = Cipher.getInstance("AES/GCM/NoPadding");
```

# 7.5操作系统命令需要校验

**Noncompliant Code Example**

```
public void run(javax.servlet.http.HttpServletRequest request) throws IOException {
  String binary = request.getParameter("binary");

  // If the value "/sbin/shutdown" is passed as binary and the web server is running as
  // then the machine running the web server will be shut down and become unavailable f

  Runtime.getRuntime().exec(binary); // Noncompliant
}
```

**Compliant Solution**

```
public void run(javax.servlet.http.HttpServletRequest request) throws IOException {
  String binary = request.getParameter("binary");

  // Restrict to binaries within the current working directory whose name only contains
  if (!binary.matches("[a-zA-Z]++")) {
    throw new IllegalArgumentException();
  }

  Runtime.getRuntime().exec(binary);
}
```

**Noncompliant Code Example**

```
public void run(javax.servlet.http.HttpServletRequest request) throws IOException {
  String binary = request.getParameter("binary");

  // If the value "/sbin/shutdown" is passed as binary and the web server is running as
  // then the machine running the web server will be shut down and become unavailable f

  Runtime.getRuntime().exec(binary); // Noncompliant
}
```

**Compliant Solution**

```
public void run(javax.servlet.http.HttpServletRequest request) throws IOException {
  String binary = request.getParameter("binary");

  // Restrict to binaries within the current working directory whose name only contains
  if (!binary.matches("[a-zA-Z]++")) {
    throw new IllegalArgumentException();
  }

  Runtime.getRuntime().exec(binary);
}
```

## 7.6 不要使用ThreadGroup

Even though thread groups are useful for keeping threads organized, programmers seldom benefit from their use because many of the methods of the `ThreadGroup` class (for example, `allowThreadSuspension()`, `resume()`, `stop()`, and `suspend()`) are deprecated. Furthermore, many nondeprecated methods are obsolete in that they offer little desirable functionality. Ironically, a few `ThreadGroup` methods are not even thread-safe [Bloch 2001].

使用线程池：

```
ThreadFactory threadFactory = Executors.defaultThreadFactory();
ThreadPoolExecutor executorPool = new ThreadPoolExecutor(3, 10, 5, TimeUnit.SECONDS,
                                 new ArrayBlockingQueue<Runnable>(2), threadFactory);

for (int i = 0; i < 10; i++) {
  executorPool.execute(new JobThread("Job: " + i));
}

System.out.println(executorPool.getActiveCount()); // Compliant
executorPool.shutdown();
```

持续更新中。。。