- 【会议安排】：
  - **主题**：【Java 代码格式规范-内部-小范围讨论】
  - **时间**：16:30 - 17:30（预期10~30 mins，快速同步、讨论）
  - **地点**：3F-新加坡会议室，7人
  - **人员**：不超过 7 人 （群中成员，选择性参加，一般不建议参加）

- 【内容要点】：
  - 代码格式规范，Java Code Style： https://wiki.mobike.com/pages/viewpage.action?pageId=32567024

小结

- **摩拜的 Code Style**：代码格式化
  - **统一**的、**良好**的、**通用**的；
  - **定制版**的 *Google Java Code Style*

- **摩拜的 Check Style**：代码校验
  - 跟 Code Style 完全对应
  - **定制版**的 *Google Check Style*

- Checkstyle 检查什么?
  - Import
  - 空白、修饰符
  - 命名约定、代码、类设计
  - 混合检查（包括一些有用的比如非必须的System.out和printstackTrace）

Java 代码规范：Code Style 和 Check Style

guoning@mobike.com
2018/08/02

目录

为什么

- **背景**：
  - 不同的小组\同学，采用**不同的**代码**格式**规范
  - 导致每次 **format 代码**，都有大量的变化
  - **review 代码**时，引入很多**干扰**项

- **目标**：
  - **统一**代码**格式**规范
  - 保证 format 代码时，**不**会引入格式上的**干扰**
  - 提升小组**协作效率**、代码 review 效率

为什么

```
package com.mobike.abacus.order;



import javax.annotation.Resource;
-
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@@ -18,19 +20,14 @@ import
org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.session.data.redis.config.ConfigureRedisAction;
import org.springframework.session.data.redis.config.annotation.web.http.EnableRedisHttpSe
ssion;

-import com.mobike.abacus.order.common.CustomLogoutHandler;
-import com.mobike.abacus.order.filter.WhiteListUtils;
-
-import lombok.extern.slf4j.Slf4j;
-
@SpringBootApplication
@EnableDiscoveryClient
@EnableOAuth2Sso
@Slf4j
@EnableRedisHttpSession(redisNamespace = "abacus_order")
@EnableAutoConfiguration
@EnableFeignClients(basePackages = { "com.mobike.client.payment", "com.mobike.client.user"
, "com.mobike.client.member", "com.mobike.client.sms",
-       "com.mobike.client.creditsystem", "com.mobike.push.client" })

public class AbacusOrderApplication extends WebSecurityConfigurerAdapter {

    public static void main(String[] args) {
@@ -46,11 +43,17 @@ public class AbacusOrderApplication extends
WebSecurityConfigurerAdapter {
        // 1. 基本配置
        http
                // a. 登出控制
-               .logout().logoutSuccessUrl("/").logoutRequestMatcher(new AntPathRequestMat
cher("/logout")).addLogoutHandler(customLogoutHandler).and()
```

```
package com.mobike.abacus.order;

+import com.mobike.abacus.order.common.CustomLogoutHandler;
+import com.mobike.abacus.order.filter.WhiteListUtils;
import javax.annotation.Resource;
+import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@@ -18,19 +20,14 @@ import
org.springframework.security.web.util.matcher.AntPathRequestMatcher;
import org.springframework.session.data.redis.config.ConfigureRedisAction;
import org.springframework.session.data.redis.config.annotation.web.http.EnableRedisHttpSe
ssion;

-import com.mobike.abacus.order.common.CustomLogoutHandler;
-import com.mobike.abacus.order.filter.WhiteListUtils;
-
-import lombok.extern.slf4j.Slf4j;
-
@SpringBootApplication
@EnableDiscoveryClient
@EnableOAuth2Sso
@Slf4j
@EnableRedisHttpSession(redisNamespace = "abacus_order")
@EnableAutoConfiguration
-@EnableFeignClients(basePackages = { "com.mobike.client.payment", "com.mobike.client.user"
, "com.mobike.client.member", "com.mobike.client.sms",
-       "com.mobike.client.creditsystem", "com.mobike.push.client" })
+@EnableFeignClients(basePackages = {"com.mobike.client.payment", "com.mobike.client.user",
 "com.mobike.client.member",
+       "com.mobike.client.sms", "com.mobike.client.creditsystem", "com.mobike.push.clien
t"})
public class AbacusOrderApplication extends WebSecurityConfigurerAdapter {

    public static void main(String[] args) {
@@ -46,11 +43,17 @@ public class AbacusOrderApplication extends
WebSecurityConfigurerAdapter {
        // 1. 基本配置
        http
                // a. 登出控制
-               .logout().logoutSuccessUrl("/").logoutRequestMatcher(new AntPathRequestMat
cher("/logout")).addLogoutHandler(customLogoutHandler).and()
+               .logout().logoutSuccessUrl("/")
+
+               .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
```

位置变化

格式变化

目录

怎么做：明确规范

- **统一**的、**良好**的、**通用**的，Java 代码规范：
  - **定制版**的 *Google Java Code Style*

- 选用 Google Java Code Style，具体原因：
  - 业界**使用广泛**，基本是**通用标准**
  - 自动化**校验工具完善**，有完善的 checkstyle 配置文档

-

怎么做：明确规范

- 为满足「代码美感」，借鉴其他公司的定制 & 公司内部建议，进行一些**摩拜的定制**：
  - GoogleStyle-Mobike 的 Java 代码规范：
    - IntelliJ IDEA： intellij-java-google-style.xml （在线查看）
    - 摩拜定制的工程： https://mobike.io/guoning/google-styleguide
  -

修改 `intellij-java-google-style.xml` 中的内容:

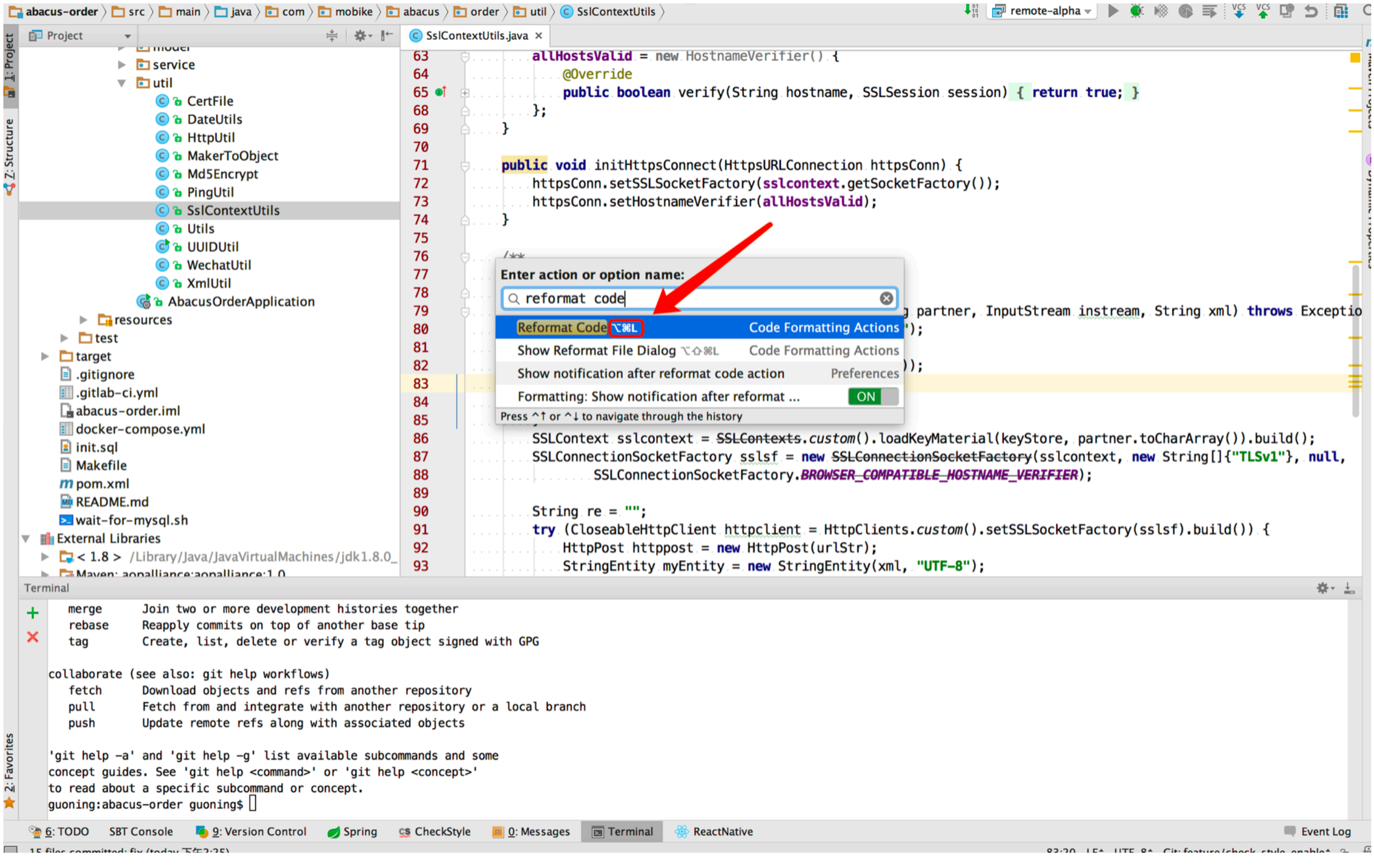
项	原始取值	定制之后	定制时间	备注
`INDENT_SIZE`	2	4	20180731	行缩进
`TAB_SIZE`	2	4	20180731	TAB 缩进
`CONTINUATION_INDENT_SIZE`	2 or 4	8	20180731	换行缩进
`RIGHT_MARGIN`	100	120	20180731	单行的长度
`JD_PRESERVE_LINE_FEEDS`	无	true	20180731	注释中, 保留手动的换行
`KEEP_LINE_BREAKS`	无	true	20180731	Java 代码中, 保留手动的换行

目录

怎么做：使用规范-代码格式化

- 配置了 code style 后，在 Mac 下，IntelliJ IDEA进行代码格式化：
  - 快捷键：Shift + Command + L
  - 界面操作：参考截图

怎么做：使用规范-代码格式化



怎么做：使用规范-代码格式校验

- 代码**自动化检查**，意义：
  - **节省人力**：机器能做的事情，交给机器，特别是枯燥的事情，让机器去做
  - **避免遗漏**：机器自动执行，全范围扫描

- 根据初步调研，决定采用通用的自动化**代码检查**工具： Checkstyle。

- 备注：蚂蚁金服、快手，团队开发过程中，都在使用 Checkstyle，进行自动化代码检查。

怎么做：使用规范-代码格式校验

- Checkstyle 会在代码开发过程中，检查代码规范，一般检查的内容包括：
  - Javadoc注释
  - 命名约定
  - 标题
  - Import
  - 大小写
  - 空白
  - 修饰符
  - 代码
  - 类设计
  - 混合检查（包活一些有用的比如非必须的System.out和printstackTrace）

怎么做：使用规范-代码格式校验

- 基于「摩拜定制的 Google Code Style」，需要对 原始 Google Code Style 的 Checkstyle 摩拜的定制：
  - 摩拜定制的 Checkstyle 配置： 在线查看
  - 摩拜定制的 Checkstyle：https://mobike.io/guoning/checkstyle

摩拜定制 Java Code Style

为了增加代码的「美感」, 摩拜内部, 进行了 java code style 的部分定制;

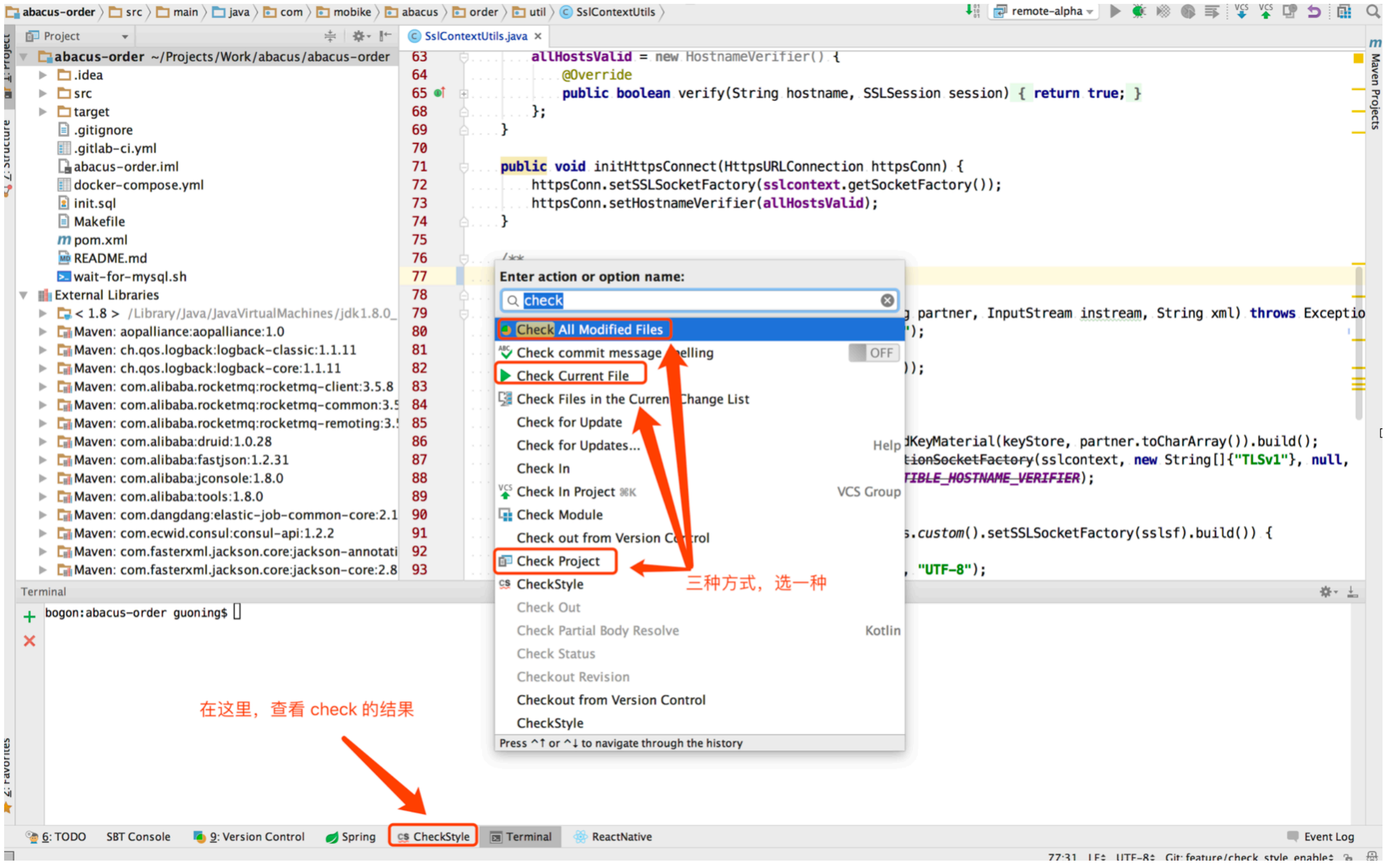
因此, 需要同步定制 checkstyle。

具体定制的内容:

项	原始取值	定制之后	定制时间	备注
`Indentation.basicOffset`	2	4	20180731	行缩进
`Indentation.caseIndent`	2	4	20180731	行缩进
`Indentation.throwsIndent`	4	8	20180731	行缩进
`Indentation.lineWrappingIndentation`	4	8	20180731	行缩进
`Indentation.arrayInitIndent`	2	4	20180731	行缩进
`LineLength`	100	120	20180731	行长度
`JavadocParagraph`	启用	删除	20180731	JavaDoc 的校验
`SummaryJavadoc`	启用	删除	20180731	JavaDoc 的校验
`VariableDeclarationUsageDistance.allowedDistance`	3	10	20180801	变量的声明和使用
`LineLength.max`	120	180	20180801	单行长度校验

更多细节, 参考: Google's Java Style Checkstyle Coverage

怎么做：使用规范-代码格式校验



目录

参考资料

- 代码格式规范：Java Code Style

- https://github.com/google/styleguide

- https://github.com/checkstyle/checkstyle

小结

- **摩拜的 Code Style**：代码格式化
  - **统一**的、**良好**的、**通用**的；
  - **定制版**的 *Google Java Code Style*

- **摩拜的 Check Style**：代码校验
  - 跟 Code Style 完全对应
  - **定制版**的 *Google Check Style*

- Checkstyle 检查什么?
  - Import
  - 空白、修饰符
  - 命名约定、代码、类设计
  - 混合检查（包括一些有用的比如非必须的System.out和printstackTrace）