# logback日志系统

(<https://logback.qos.ch/manual> )

### Conditional processing of configuration files

Developers often need to juggle between several logback configuration files targeting different environments such as development, testing and production. These configuration files have substantial parts in common differing only in a few places. To avoid duplication, logback supports conditional processing of configuration files with the help of <if>, <then> and <else> elements so that a single configuration file can adequately target several environments. Note that conditional processing requires the [Janino library](https://logback.qos.ch/setup.html#janino).

The general format for conditional statements is shown below.

   <!-- if-then form -->  
   <if condition="some conditional expression">  
    <then>  
      ...  
    </then>  
  </if>  
    
  <!-- if-then-else form -->  
  <if condition="some conditional expression">  
    <then>  
      ...  
    </then>  
    <else>  
      ...  
    </else>      
  </if>

The condition is a Java expression in which only context properties or system properties are accessible. For a key passed as argument, the property() or its shorter equivalent p() methods return the String value of the property. For example, to access the value of a property with key "k", you would write property("k") or equivalently p("k"). If the property with key "k" is undefined, the property method will return the empty string and not null. This avoids the need to check for null values.

The isDefined() method can be used to check whether a property is defined. For example, to check whether the property "k" is defined you would write isDefined("k") Similarly, if you need to check whether a property is null, the isNull() method is provided. Example: isNull("k").

<configuration debug="true">  
  
  **<if condition='property("HOSTNAME").contains("torino")'>**  
    **<then>**  
      <appender name="CON" class="ch.qos.logback.core.ConsoleAppender">  
        <encoder>  
          <pattern>%d %-5level %logger{35} - %msg %n</pattern>  
        </encoder>  
      </appender>  
      <root>  
        <appender-ref ref="CON" />  
      </root>  
    **</then>**  
  **</if>**  
  
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">  
    <file>${randomOutputDir}/conditional.log</file>  
    <encoder>  
      <pattern>%d %-5level %logger{35} - %msg %n</pattern>  
   </encoder>  
  </appender>  
  
  <root level="ERROR">  
     <appender-ref ref="FILE" />  
  </root>  
</configuration>

Conditional processing is supported anywhere within the <configuration> element. Nested if-then-else statements are also supported. However, XML syntax is awfully cumbersome and is ill suited as the foundation of a general purpose programming language. Consequently, too many conditionals will quickly render your configuration files incomprehensible to subsequent readers, including yourself.

# 2.git生产／测试

# 3. RESTful API 设计指南

<http://www.ruanyifeng.com/blog/2014/05/restful_api.html>

spring实战第11章

# 4.[MySQL之alter语句用法总结](http://www.cnblogs.com/aspnethot/articles/1397130.html)

1：删除列

ALTER TABLE 【表名字】 DROP 【列名称】

2：增加列

ALTER TABLE 【表名字】 ADD 【列名称】 INT NOT NULL  COMMENT '注释说明'

3：修改列的类型信息

ALTER TABLE 【表名字】 CHANGE 【列名称】【新列名称（这里可以用和原来列同名即可）】 BIGINT NOT NULL  COMMENT '注释说明'

4：重命名列

ALTER TABLE 【表名字】 CHANGE 【列名称】【新列名称】 BIGINT NOT NULL  COMMENT '注释说明'

5：重命名表

ALTER TABLE 【表名字】 RENAME 【表新名字】

6：删除表中主键

Alter TABLE 【表名字】 drop primary key

7：添加主键

ALTER TABLE sj\_resource\_charges ADD CONSTRAINT PK\_SJ\_RESOURCE\_CHARGES PRIMARY KEY (resid,resfromid)

8：添加索引

ALTER TABLE sj\_resource\_charges add index INDEX\_NAME (name);

9: 添加唯一限制条件索引

ALTER TABLE sj\_resource\_charges add unique emp\_name2(cardnumber);

10: 删除索引

alter table tablename drop index emp\_name;

# 5. [Java注释@interface的用法【转】](http://blog.csdn.net/liuwenbo0920/article/details/7290586)

# 6.git for windows

# [7.@Appconf](mailto:7.@Appconf)为什么用@Component修饰

# 8.部署错误

/cn/caijiajia/confplus/client/AppConfManager.class]: Invocation of init method failed; nested exception is java.io.FileNotFoundException: /appConfig.cache (No such file or directory)

2017-06-17 16:00:56.621 [APP\_NAME\_IS\_UNDEFINED|SERVER\_IP\_IS\_UNDEFINED|SERVER\_PORT\_IS\_UNDEFINED] [Thread-9] [c.c.c.c.service.AppConfigService] [ERROR] [153] - Dump app config error.

java.io.FileNotFoundException: /appConfig.cache (Permission denied)

at java.io.FileOutputStream.open0(Native Method) ~[na:1.8.0\_74]

at java.io.FileOutputStream.open(FileOutputStream.java:270) ~[na:1.8.0\_74]

at java.io.FileOutputStream.<init>(FileOutputStream.java:213) ~[na:1.8.0\_74]

at java.io.FileOutputStream.<init>(FileOutputStream.java:101) ~[na:1.8.0\_74]

at java.io.FileWriter.<init>(FileWriter.java:63) ~[na:1.8.0\_74]

at cn.caijiajia.confplus.client.dumper.ConfigDumper.dumpAppConfig(ConfigDumper.java:48) ~[confplus-client-1.0.0-SNAPSHOT.jar:na]

at cn.caijiajia.confplus.client.service.AppConfigService.refreshConfig(AppConfigService.java:151) ~[confplus-client-1.0.0-SNAPSHOT.jar:na]

at cn.caijiajia.confplus.client.AppConfManager$RefreshConfigThread.run(AppConfManager.java:100) [confplus-client-1.0.0-SNAPSHOT.jar:na]

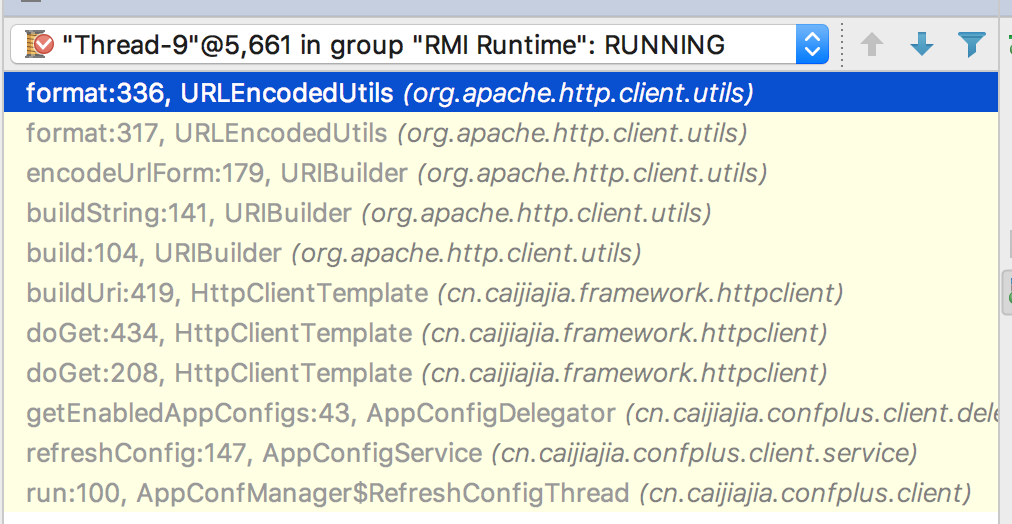
2017-06-18 14:59:08.916 [APP\_NAME\_IS\_UNDEFINED|SERVER\_IP\_IS\_UNDEFINED|SERVER\_PORT\_IS\_UNDEFINED] [RMI TCP Connection(2)-127.0.0.1] [c.c.c.c.service.AppConfigService] [ERROR] [54] - **Load config from confplus error**.

2017-06-18 14:59:08.948 [APP\_NAME\_IS\_UNDEFINED|SERVER\_IP\_IS\_UNDEFINED|SERVER\_PORT\_IS\_UNDEFINED] [Curator-ConnectionStateManager-0] [c.c.s.client.JobExecutorRegister] [ERROR] [26] - **Register Job error.**

# 9.traininghuanbei OK是怎么打出来的

/home 对应的controller会返回 context path + “ok”

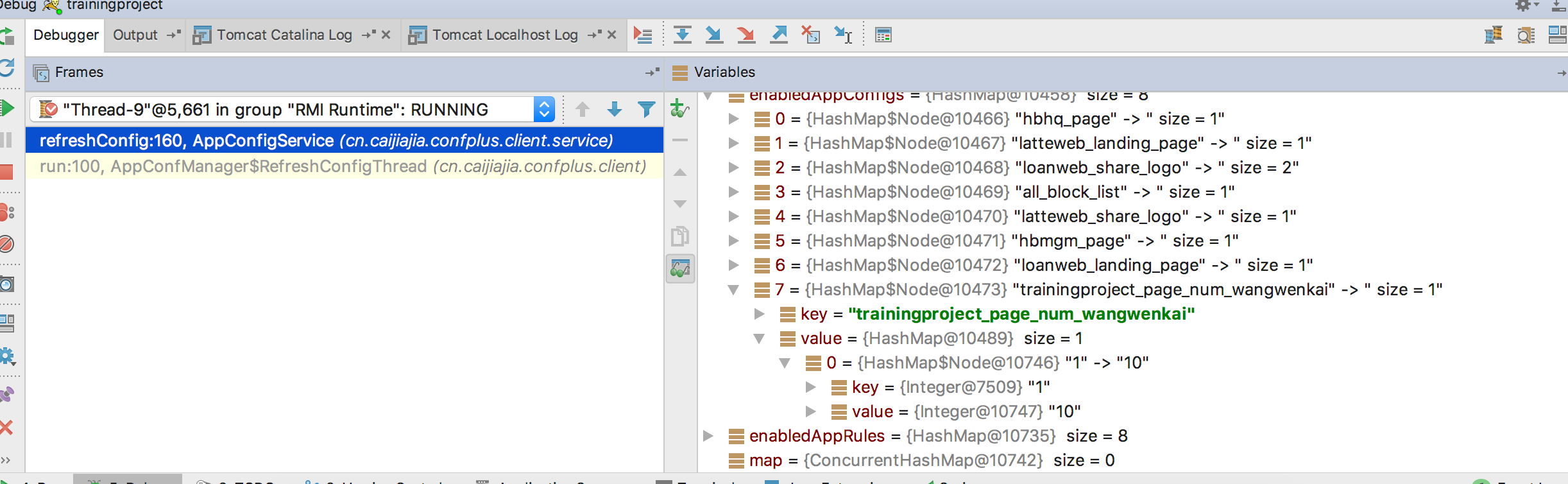
# 10.为什么项目的artifactid,display-name和param-name的名字要和配置中心的一样



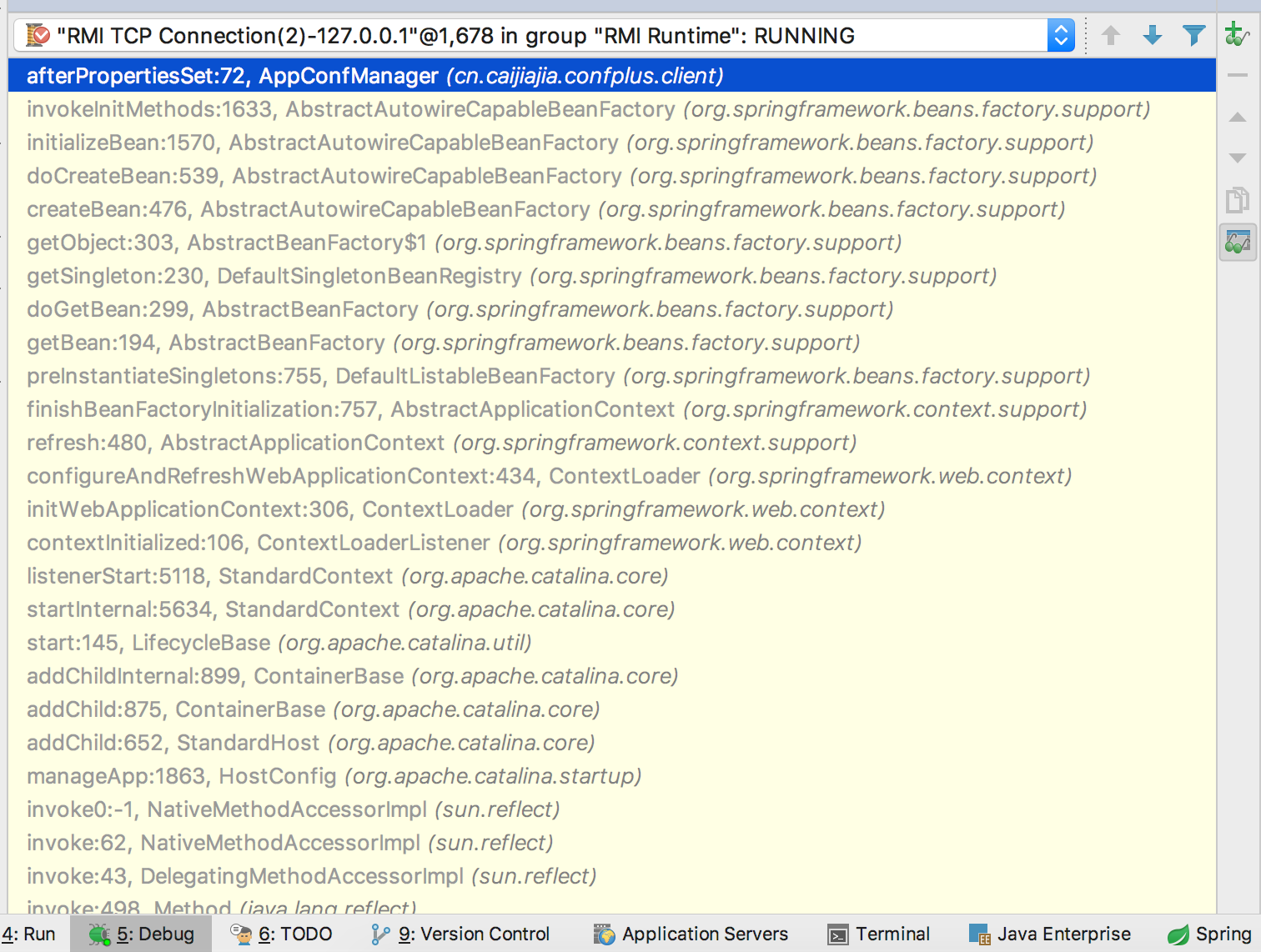
<http://confplus.apps.sit.caijj.net/confplus/enabledappconfigs?application=traininghuanbei&r_i=d8c02d9e-a30a-43c7-b18c-e9838930ef69.W.1&r_e=traininghuanbei>

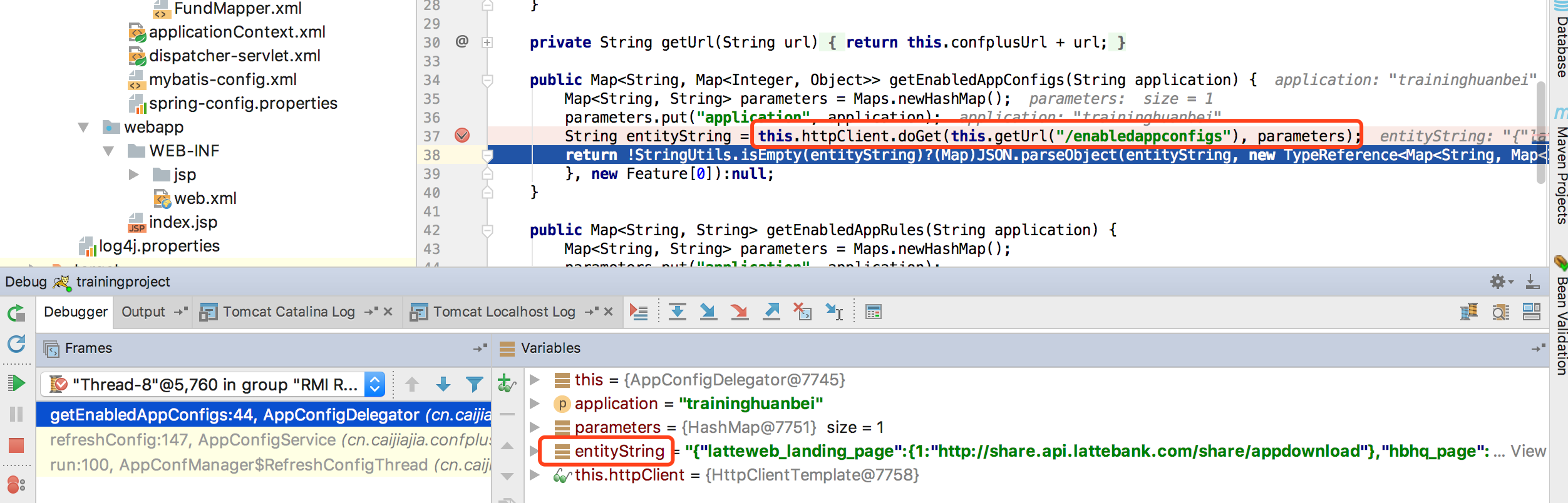
参数的含义是什么？

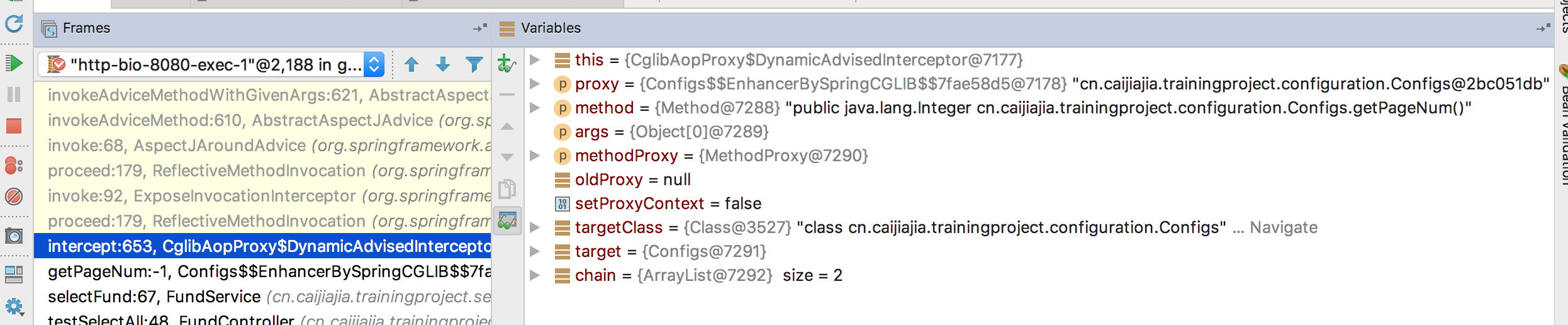
\u0080



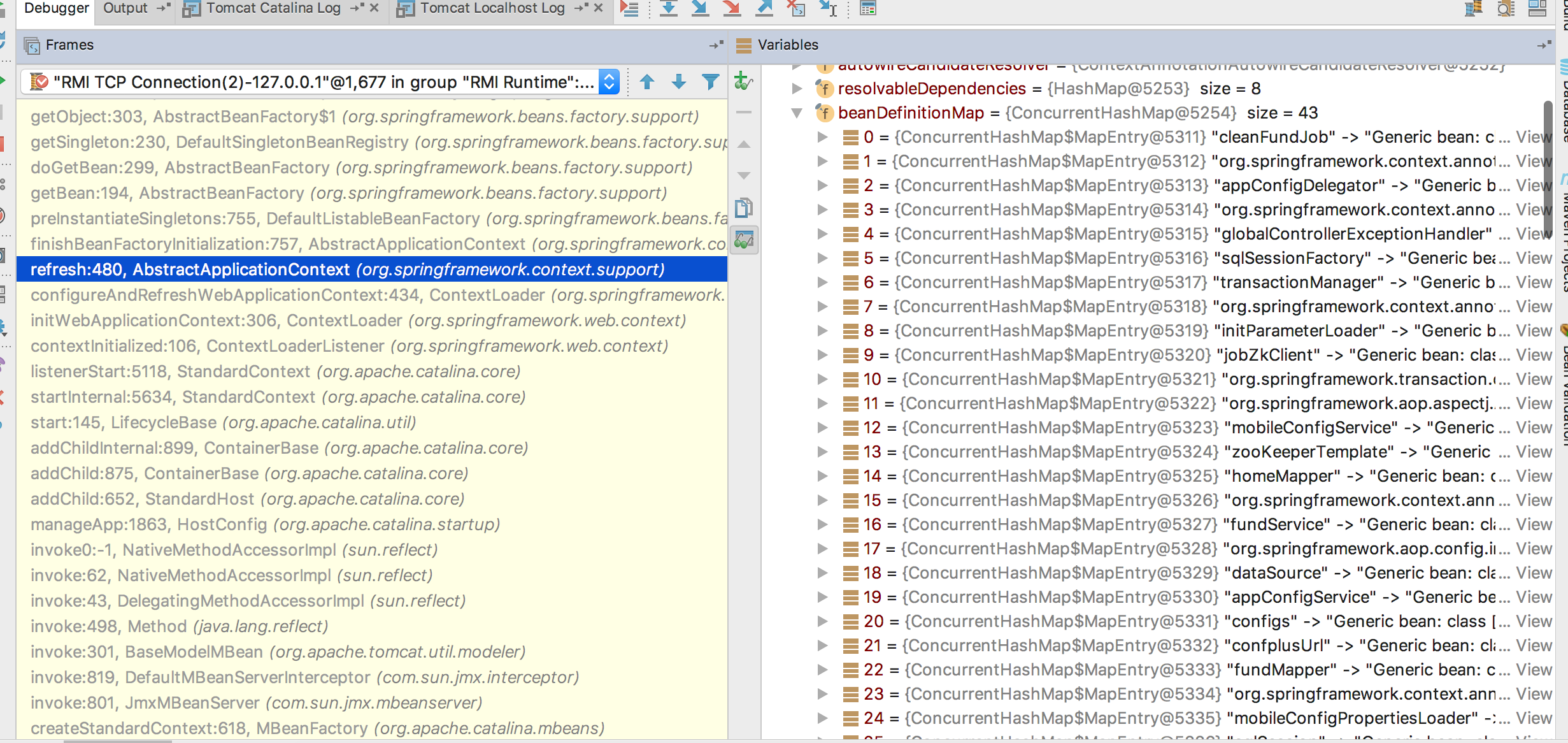
从配置中心获取配置的变量原理：因为AppConfManager注解为Component，在spring容器初始化的时候会实例化该bean，该bean实现了InitializingBean接口的afterPropertiesSet()方法，初始化时会执行该方法（sqlSessionFactory、DaoSupport、MapperFactoryBean(所有的mapper)、ListFactoryBean、DataSourceTransactionManager、也实现了该方法）







bean实例化跟踪

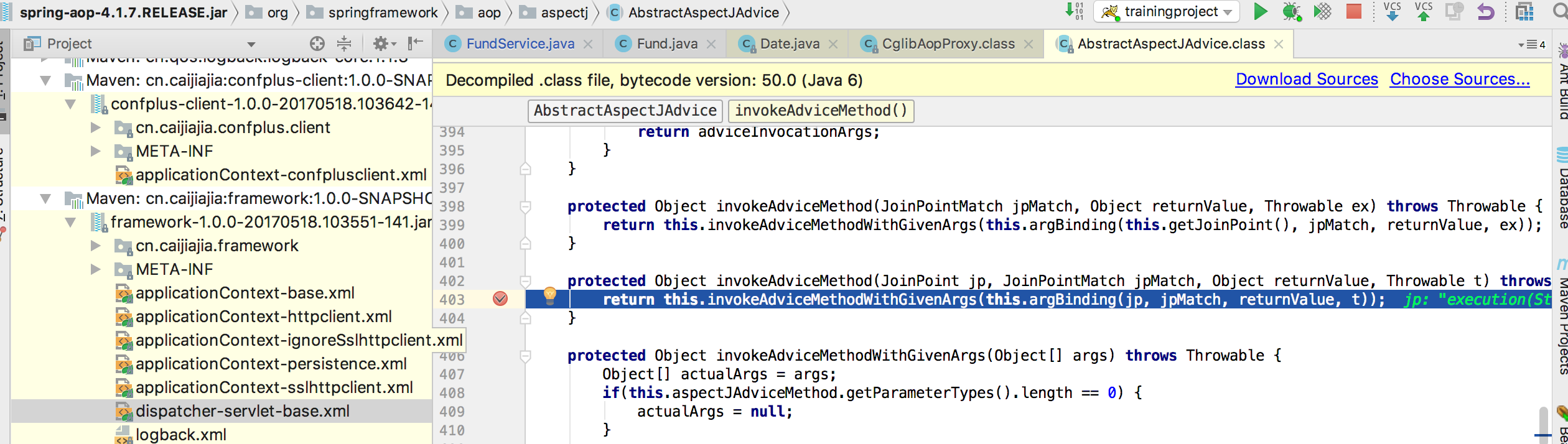


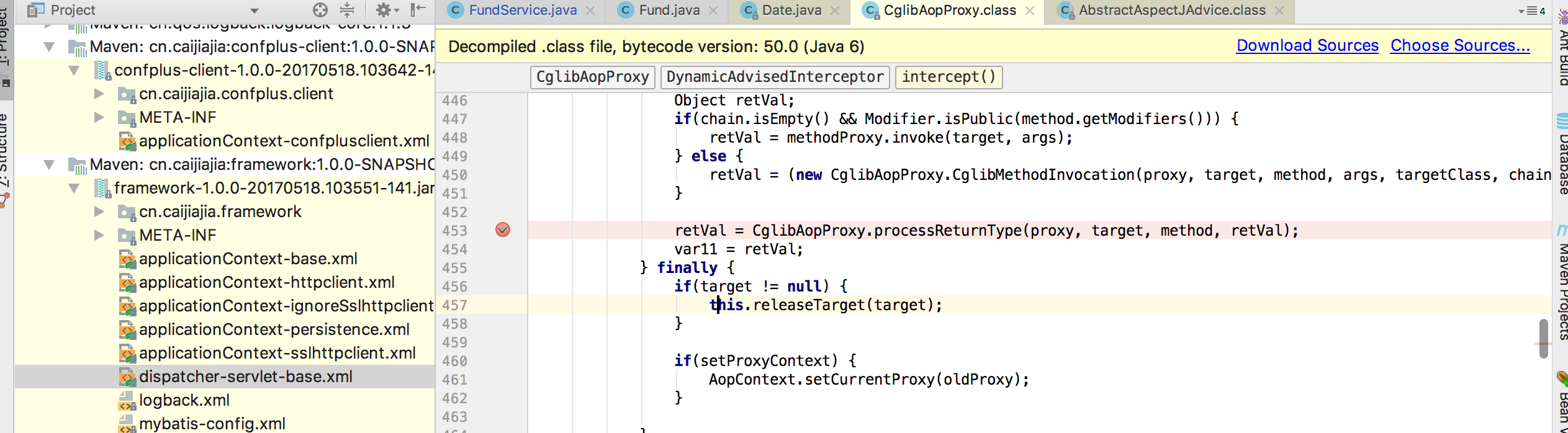
# 11.[Spring容器初始化过程](http://www.cnblogs.com/luyanliang/p/5567164.html)

autowired



# 12.时间戳转日期





# 13．Pagehelper只能用于分页查询，不能用于分页删除

# 14.cjjException