总结：

再次熟悉了下面8点：

一、多线程：线程池ThreadPoolExecutor，prototypeTask任务，Future， CountDownLatch主线程等带所有子线程任务执行结束。

private static ThreadPoolExecutor *pool* = new ThreadPoolExecutor(100,  
 200, 1000, TimeUnit.*MILLISECONDS*, new LinkedBlockingQueue<>(),  
 new DefaultThreadFactory("cod-pool"), new ThreadPoolExecutor.DiscardOldestPolicy());

没有用下面这种方式的线程池

ExecutorService executorService = Executors.newCachedThreadPool();

下面这种线程不可控，出了问题会影响到整个机器。

二、在代码中获取Spring容器ApplicationContext，其实有很多中方法，在这里尝试了下面2中方法：

1、ApplicationContext context = ContextLoader.getCurrentWebApplicationContext();

这种方法需要依赖spring-web jar包，因为是在非web模块，所以有点不得劲。

2、通过实现ApplicationContextAware接口的setApplicationContext(ApplicationContext applicationContext)方法来获取Spring的容器ApplicationContext。这样不需要依赖spring-web包，这次使用的就是这样方法。

三、gzip压缩数据，需要gzip.finish()结束数据写入ByteArrayOutputStream流，才能通过out.toByteArray()获取到流中的数据。

四、在程序中加载配置文件，这个程序是web程序中的一个依赖jar。也就是说在jar包中读取jar包中的配置文件。

第一种方法。

URL url = TestBase.class.getResource(name);  
String path = url.getPath();  
File file = new File(path);  
String conent = FileUtils.*readFileToString*(file, "UTF-8");  
Map<String, Object> dataMap = JSONObject.*parseObject*(conent, Map.class);

用getResouce("")在main方法中测试是木有问题的，但是部署到tomcat中始终都报错

jar:file:/D:/tools/apache-tomcat-8.5.31/webapps/ROOT/WEB-INF/lib/jolly-service-1.2.0-SNAPSHOT.jar!/cod/base\_setting.json

说下面的文件（就是要加载的文件）找不到，但是文件确实是有的。无解，没整清楚道理。

第二种方法。

InputStream inputStream = TestBase.class.getResourceAsStream(name);  
BufferedReader br = new BufferedReader(new InputStreamReader(inputStream));  
StringBuilder sb = new StringBuilder();  
String line = br.readLine();  
while (line != null){  
 sb.append(line);  
 line = br.readLine();  
}  
Map<String, Object> dataMap = JSONObject.*parseObject*(sb.toString(), Map.class);

用getResourcesAsStream("")在main放发中测试，还是部署到tomcat中都可以成功读取到文件内容。

注：而且是静态变量、静态块、静态方法，就是在类加载的时候执行的，而且是类变量。

对象实例化过程又有点模糊了，得找个时间温习一下。

五、自定义对象转换称map

*/\*\*  
 \* 获取利用反射获取类里面的值和名称  
 \*  
 \** ***@param*** *obj  
 \** ***@return*** *\** ***@throws*** *IllegalAccessException  
 \*/*public static Map<String, Object> objectToMap(Object obj)throws IllegalAccessException{  
 Map<String, Object> map = new HashMap<>();  
 Class<?> clazz = obj.getClass();  
 for (Field field : clazz.getDeclaredFields()) {  
 field.setAccessible(true);  
 String fieldName = field.getName();  
 Object value = field.get(obj);  
 map.put(fieldName, value);  
 }  
 return map;  
}

六、定义业务异常，在一个工功能里层逻辑抛出业务逻辑，在最外层逻辑进行捕获做相应的异常处理逻辑。

这个功能也可以通过返回值来处理，如果里层如果不满足逻辑要求，或者是出现异常，当即捕获异常进行处理，设置对应的返回值，或空对象，或空字符串，或空数组集合等等来处理。

这个档需要好好设计，才能让编码跟方便更快捷，逻辑更清楚。主要是要好好设计。

七、Hbase查询，这个没什么好说的，只要能正常脸上hbase，操作比较简单。

八、LocalDate格式话的时候，默认格式，预定义的格式都不满足要求，可以自定义格式，如下：

*/\*\*  
 \* localDate自定义FORMATTER  
 \*/*private static DateTimeFormatter *FORMATTER* = new DateTimeFormatterBuilder()  
 .appendValue(*YEAR*, 4, 10, SignStyle.*EXCEEDS\_PAD*)  
 .appendValue(*MONTH\_OF\_YEAR*, 2)  
 .appendValue(*DAY\_OF\_MONTH*, 2)  
 .toFormatter();

使用如下：

LocalDate date; date.format(FORMATTER); 就OK了

这次编码不足的地方：

1、代码中用到了很多魔法常量，这个是很不好的。以前编码重来都是很规范的，虽然现在公司这方面不重视（现在讲究速度），但是作为一个有情怀的优秀的coder还是的注意。

2、注释有但是不是特别详细，方法抽取，if else避免，都做的不是很好。