开发指导书V1.0

版本	时间	修改人	说明
1.0.0	2018-05-30	柯尊超	--

一、框架介绍

框架是以spring和cxf为基础，集成了mybaties，将服务配置信息集中化，充分利用Spring的特性，按照MVC的模型进行开发。与之前的开发模式相比，更换了持久层为`Mybaties`，不再推荐使用之前的`workspace`，但是还是兼容之前的workspace的用法。

种子文件地址:https://github.com/zizhengzhuan/cf-service-seed

二、基础环境

- Git: 种子文件托管空间（git介绍）
- Maven：包依赖管理（maven介绍）
- JAVA：java编译，版本统一为 jdk1.8
- tomcat: 程序运行环境，版本为 8.x
- IDE：统一为`IntelliJ IDEA`,不反对使用MyEclipse，但是不提供支持。

三、工程代码

1.项目主体结构

服务包和业务实现包以子项目形式存在。

- sp-demo: 单独的服务包
- demo-object: 服务对应的业务实现
- pom.xml: 基础依赖，具体服务包会继承这个pom的依赖，不要重复引用

2.子模块的结构

2.1 命名规范

- 服务包以`sp-`开头，如`sp-demo`；
- 服务实现包以`-object`结尾，如`demo-object`；
- 服务包的基础报名前缀为`com.zzht.service.xxxx`；
- 业务实现包的报名前缀为`com.zzht.component.xxxx`；

示例代码：

```
//#1.服务包
pakacge com.zzht.service.demo.common;
//......
//....

// #2.实现包
pakacge com.zzht.service.demo.webservice
//....
//....
```

2.2 sp服务包代码结构

```
## main
- common            公共部分
    - ServiceCore   实现IService，，用来secore发布服务
- contanst          常量等基本定义
- exception         异常或者返回的消息定义
- webservice                服务接口
    -restful        服务接口restful
    -soap           服务接口soap
## resource
- Service.xml           服务发布依赖的bean
- Service-REST-http.xml  REST服务地址绑定
- Service-SOAP-http.xml  SOAP服务地址绑定
```

2.3 object实现包结构

```
## main:com.zzht.component.xxx
- api           jdk级别的API调用，包含逻辑实现层
- common        公共部分抽离
- dao           数据访问层
- entity        实体类
- exception      自定义异常
## resource
- com/zzht/component/xxx/dao                mybaties数据访问配置文件目录
    - XxxxMapper.xml    mapper配置
- spring/sprig-mabaties.xml mybaties与spring集成配置
```

2.4 maven打包使用

```
mvn clean install:将打包的jar添加到仓库（本地）
mvn clean package: 打包
```

3.应用的配置文件

1. 全局配置文件

应用的配置文件统一放在`cityserver/conf`里面，`application.yml`，也允许自己项目增加额外配置的文件，非必要情况下不推荐。

YAML规范（**尤其注意缩进和冒号之后的空格**）

YAML校验

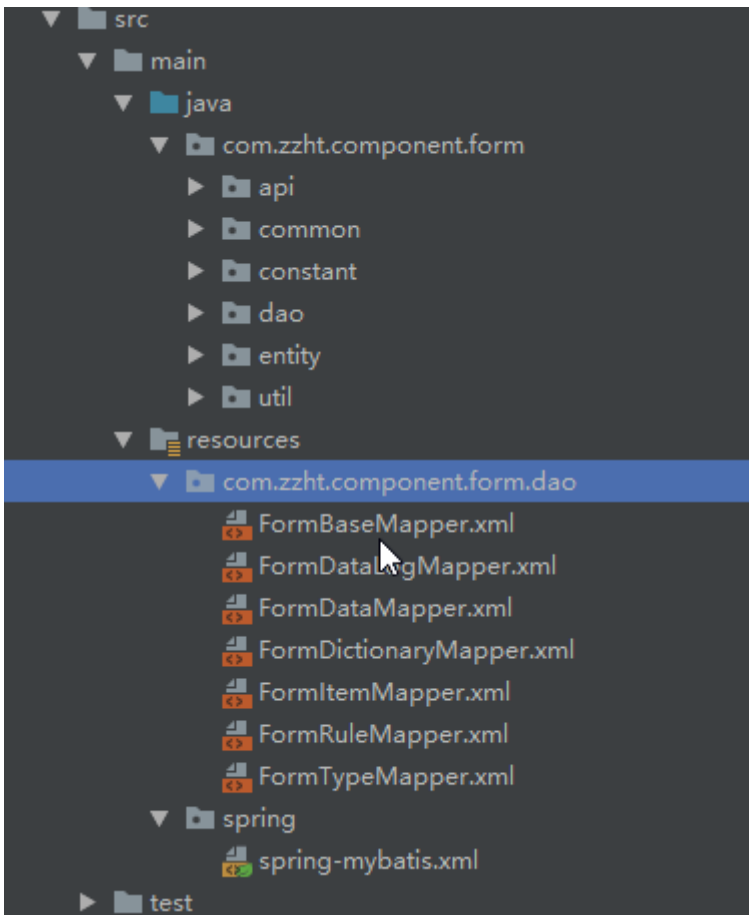
文件`application.yml`

```
## 数据库配置
###数据库类型：mysql、mssql、oracle


### driverClassName:
```

```
### mysql:com.mysql.jdbc.Driver
### mssql:com.microsoft.sqlserver.jdbc.SQLServerDriver
### oracle:oracle.jdbc.driver.OracleDriver

## 基础数据库配置
db: &base1
  type: mssql
  driver: com.microsoft.sqlserver.jdbc.SQLServerDriver
  url: jdbc:sqlserver://192.168.8.183:1433;DatabaseName=cole_flower
  username: zzht
  password: zzht
  initialSize: 0
  maxActive: 200
  maxIdle: 20
  minIdle: 1
  maxWait: 6000

##具体的项目配置文件可以直接引用基本数据库配置
oms: *base1
form:
  <<: *base2
  url: 192.168.8.183:1433:ecity
```

2.与spring集成

所有与spring的集成配置文件建议放在spring下面，classpath:spring/*.xml都会默认加入到context中。如果您不想你的配置文件集成xml自动加入context，放在除spring之外的位置。



例如：spring/spring-mybatis.xml

注意：bean的id是应用内唯一的，各项目需要添加前缀，把xxx替换掉。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-4.0.xsd">

    <!-- 配置数据源 -->
    <!-- 配置数据源 -->
    <bean id="xxxDataSource" destroy-method="close"
          class="org.apache.commons.dbcp2.BasicDataSource"
          p:driverClassName="${form.driver}"
          p:url="${form.url}"
          p:username="${form.username}"
          p:password="${form.password}"
          p:initialSize="${form.initialSize}"
          p:maxTotal="${form.maxActive}"
          p:maxIdle="${form.maxIdle}"
          p:minIdle="${form.minIdle}"
    />

    <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->
    <bean id="xxxSqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"
          p:dataSource-ref="xxxDataSource"
          p:mapperLocations="classpath:com/zzht/component/form/dao/*.xml"/>

    <!-- DAO接口所在包名，Spring会自动查找其下的类 --><!--只扫描带@Repository标签类-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"
          p:annotationClass="org.springframework.stereotype.Repository"
          p:basePackage="com.zzht.component.form.dao"
          p:sqlSessionFactoryBeanName="xxxSqlSessionFactory"/>

    <!-- 事务管理 -->
    <bean id="xxxTransactionManager"
          class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
          p:dataSource-ref="xxxDataSource"/>

    <tx:annotation-driven transaction-manager="xxxTransactionManager"/>


</beans>
```

3.JAVA获取配置

- 获取所有配置(Properties)：`AppConfig.Init(homePath).getApplicationProperties()`

- 按照前缀获取：`getApplicationProperties("oms")`
- `AppConfig`依赖`cole-flower-common-1.0.0-RELEASE.jar`

4.workspace使用配置文件

方式一：使用JAVA代码根据应用根地址获取

```
    Properties props = AppConfig.Init(homePath).getApplicationProperties();
        String dbType = props.getProperty("oms.type");
        String dbUrl = props.getProperty("oms.url");
        String name = props.getProperty("oms.username");
        String psw = props.getProperty("oms.password");
        String minIdle =props.getProperty("oms.minIdle");
        String maxIdle =props.getProperty("oms.maxIdle");
        int minnum = minIdle==null?1:Integer.valueOf(minIdle);
        int maxnum = maxIdle==null?1:Integer.valueOf(maxIdle);

        if(dbType.equalsIgnoreCase("mssql")) {
            dbType = "sql";
        }
        //.....其他的和之前没有区别
//==========分割线:按照前缀获取==========
//参数: (String prefix)
Properties props = AppConfig.Init(homePath).getApplicationProperties("oms");
String dbType = props.getProperty("type");
// ohthers.....
```

方式二：注解从全局的配置bean(`yamlProperties`)获取

```
package com.zzht.component.form.common;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

/**
 * 获取form的配置
 * @author : kunhour
 * @version :1.0
 * @since : 2018/5/30 9:24
*/
@Component
public class Configuration {

    @Value("#{yamlProperties['form.type']}")
    private String dbType;

    @Value("#{yamlProperties['form.wsUrl']}")
    private String dbUrl;

    @Value("#{yamlProperties['form.username']}")
    private String dbUsername;

    @Value("#{yamlProperties['form.password']}")
```

```
    private String dbPassword;

    public String getDbType() {
        return dbType;
    }

    public String getDbUrl() {
        return dbUrl;
    }

    public String getDbUsername() {
        return dbUsername;
    }

    public String getDbPassword() {
        return dbPassword;
    }

    public void setDbType(String dbType) {
        this.dbType = dbType;
    }

    public void setDbUrl(String dbUrl) {
        this.dbUrl = dbUrl;
    }
    public void setDbUsername(String dbUsername) {
        this.dbUsername = dbUsername;
    }

    public void setDbPassword(String dbPassword) {
        this.dbPassword = dbPassword;
    }
}
```

4.Mybatis-generator插件快速生成代码

1.在object模块的pom.xml中添加插件：

```
<plugins>
      <plugin>
        <!--Mybatis-generator插件,用于自动生成Mapper和POJO-->
        <groupId>org.mybatis.generator</groupId>
        <artifactId>mybatis-generator-maven-plugin</artifactId>
        <version>1.3.2</version>
        <configuration>
          <!--配置文件的位置-->
          <configurationFile>src/main/resources/mybatis-generator-config.xml</configurationFile>
          <verbose>true</verbose>
          <overwrite>true</overwrite>
        </configuration>
        <executions>

          <execution>
```

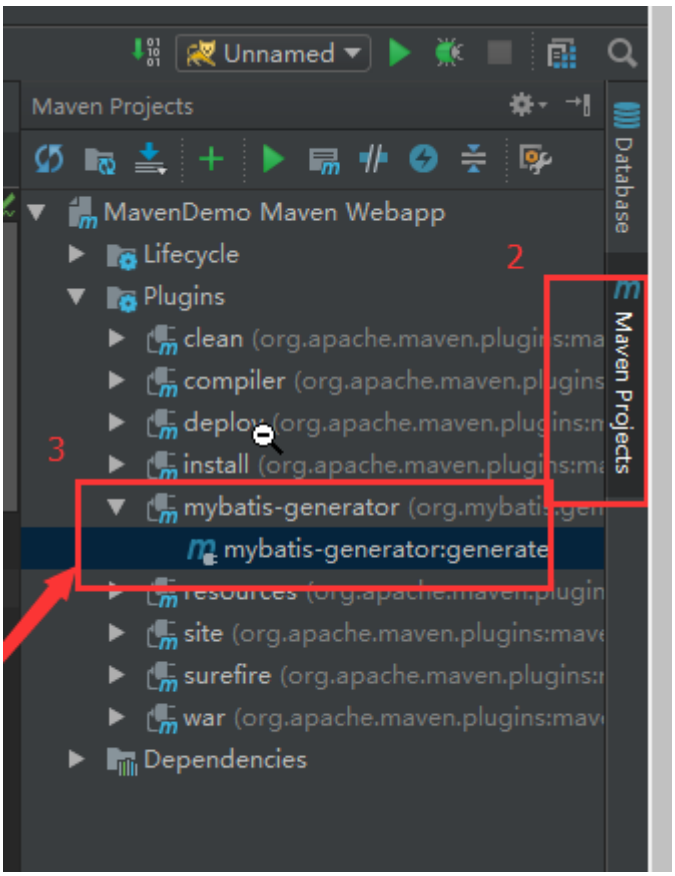
```
            <id>Generate MyBatis Artifacts</id>
            <goals>
              <goal>generate</goal>
            </goals>
          </execution>
        </executions>
        <dependencies>
          <dependency>
            <groupId>org.mybatis.generator</groupId>
            <artifactId>mybatis-generator-core</artifactId>
            <version>1.3.2</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
```

2.配置`mybatis-generator-config.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
  "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
   <!--mysql 连接数据库jar 这里选择自己本地位置-->
   <classPathEntry location="D:/mysql-connector-java-5.1.20-bin.jar" />
   <context id="testTables" targetRuntime="MyBatis3">
      <commentGenerator>
         <!-- 是否去除自动生成的注释 true：是 ： false:否 -->
         <property name="suppressAllComments" value="true" />
      </commentGenerator>
      <!--数据库连接的信息：驱动类、连接地址、用户名、密码 -->
      <jdbcConnection driverClass="com.mysql.jdbc.Driver"
         connectionURL="jdbc:mysql://localhost:3306/ecps" userId="root"
         password="root">
      </jdbcConnection>
      <!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC DECIMAL 和
         NUMERIC 类型解析为java.math.BigDecimal -->
      <javaTypeResolver>
         <property name="forceBigDecimals" value="false" />
      </javaTypeResolver>

      <!-- targetProject:生成PO类的位置 -->
      <javaModelGenerator targetPackage="com.ecps.seckill.pojo"
         targetProject="src/main/java">
         <!-- enableSubPackages:是否让schema作为包的后缀 -->
         <property name="enableSubPackages" value="false" />
         <!-- 从数据库返回的值被清理前后的空格 -->
         <property name="trimStrings" value="true" />
      </javaModelGenerator>
        <!-- targetProject:mapper映射文件生成的位置

           如果maven工程只是单独的一个工程，targetProject="src/main/java"
```

```
            若果maven工程是分模块的工程，targetProject="所属模块的名称"，例如：
            targetProject="ecps-manager-mapper"，下同-->
        <sqlMapGenerator targetPackage="com.ecps.seckill.mapper"
            targetProject="src/main/java">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </sqlMapGenerator>
        <!-- targetPackage：mapper接口生成的位置 -->
        <javaClientGenerator type="XMLMAPPER"
            targetPackage="com.ecps.seckill.mapper"
            targetProject="src/main/java">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </javaClientGenerator>
        <!-- 指定数据库表 -->
        <table schema="" tableName="seckill"></table>
        <table schema="" tableName="success_killed"></table>
    </context>
</generatorConfiguration>
```

3.在`idea`的右侧栏点击Maven,选中添加的Mybatis-generator插件并运



四、开发规范

开发规约

citysever的开发规范经过评估，决定采用阿里巴巴的`p3c规约`。《阿里巴巴Java开发手册》涵盖编程规约、单元测试规约、异常日志规约、MySQL规约、工程规约、安全规约等，使用这个规范希望能够帮助开发团队在Java开发上更高效、容错、有协作性，提高代码质量，降低项目维护成本。

p3c地址：https://github.com/alibaba/p3c

p3c完整PDF文档：在线地址

插件安装

参照p3c插件安装方法：https://github.com/alibaba/p3c/tree/master/idea-plugin

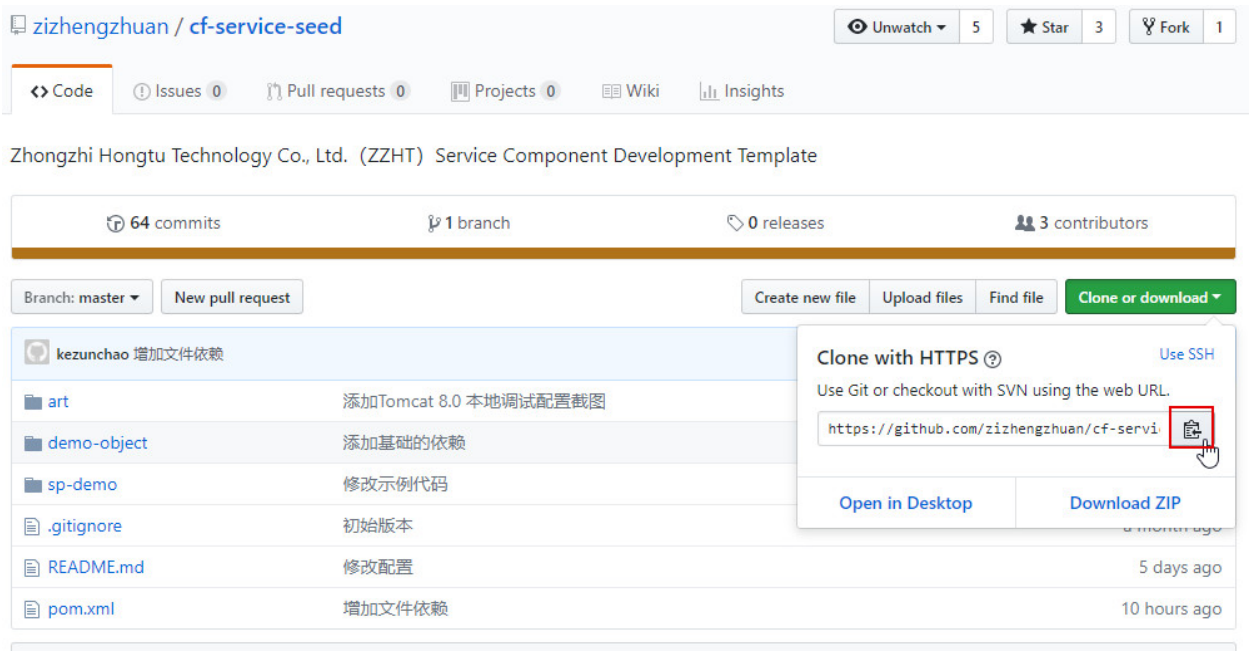
五、环境搭建与配置

> 务必保证基础环境中的提到的环境都安装完成并做好了相关的配置。

Step1:github克隆种子项目

种子项目地址：https://github.com/zizhengzhuan/cf-service-seed

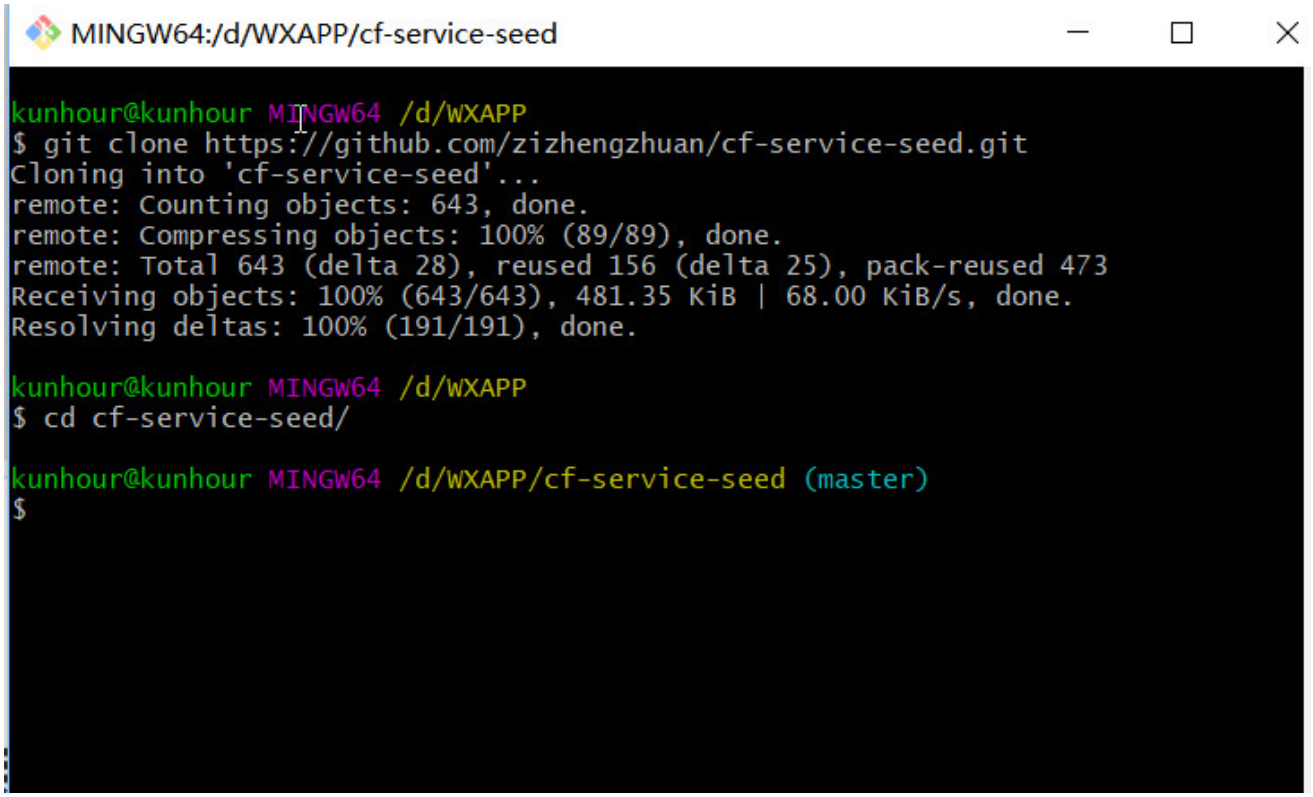
1. 复制项目地址



2. 打开Git的Bash

3. clone项目到本地

```
kunhour@kunhour MINGW64 /d/WXAPP
$ git clone https://github.com/zizhengzhuan/cf-service-seed.git
Cloning into 'cf-service-seed'...
remote: Counting objects: 643, done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 643 (delta 28), reused 156 (delta 25), pack-reused 473
Receiving objects: 100% (643/643), 481.35 KiB | 68.00 KiB/s, done.
Resolving deltas: 100% (191/191), done.

kunhour@kunhour MINGW64 /d/WXAPP
$ cd cf-service-seed/
```



Step2:cityserver基础包

CityServer地址：https://github.com/zizhengzhuan/CityServer

将cityserver的包解压到tomcat的webapps下，作为一个应用发布。cityserver的结构说明：

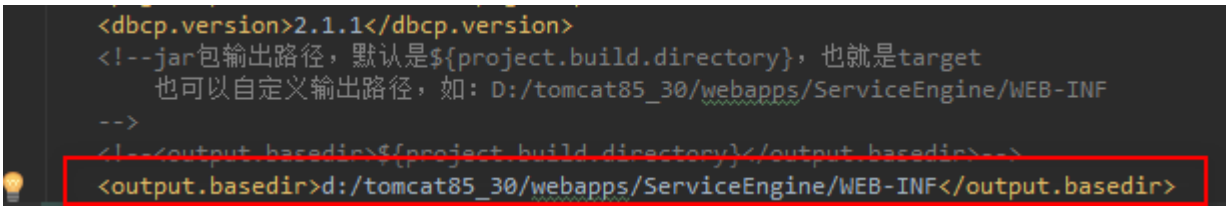
```
- conf              --配置文件
 - application.yml   --项目全局配置文件
- services          --服务包的目录
 - sp-xxx.jar
 - .....
- lib               --依赖包地址
 - xxxxx.jar
- WEB-INF           --应用的入口
  - beans.xml
  - web.xml
```

Step3:配置私有maven仓库

参照《maven私服使用-1.0.2.pdf》

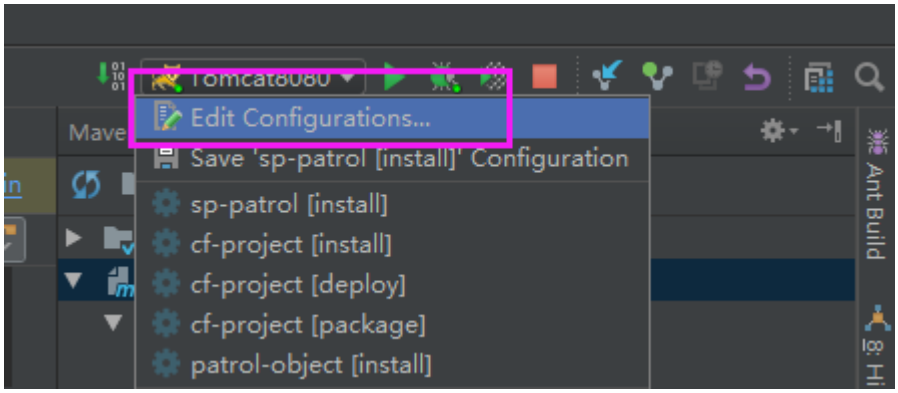
Step4:maven打包路径

修改pom.xml中`output.dir`路径.

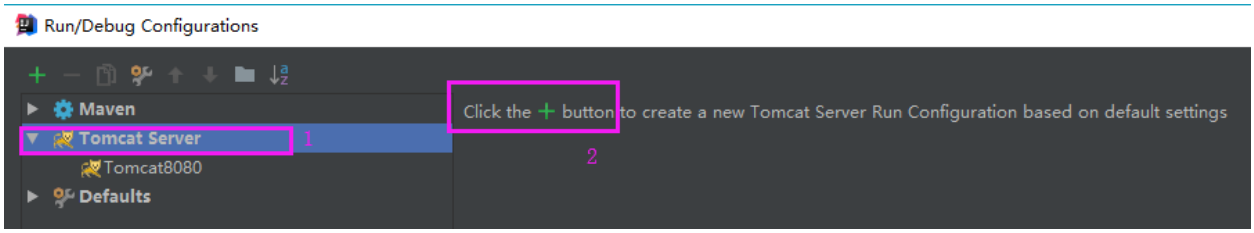


Step5:IDEA下代码调试

1. IDEA调试工具栏中编辑调试配置，选择Edit Configurations

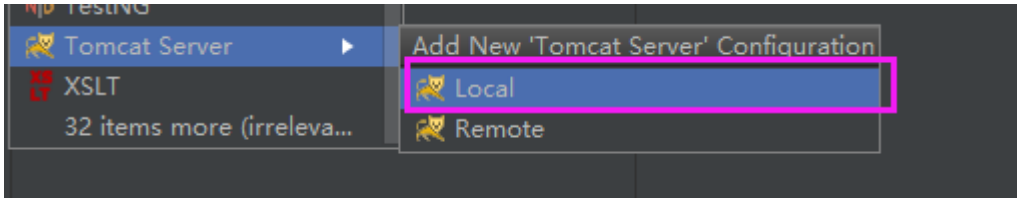


2. 选择Tomcat Server，点击绿色 + 号

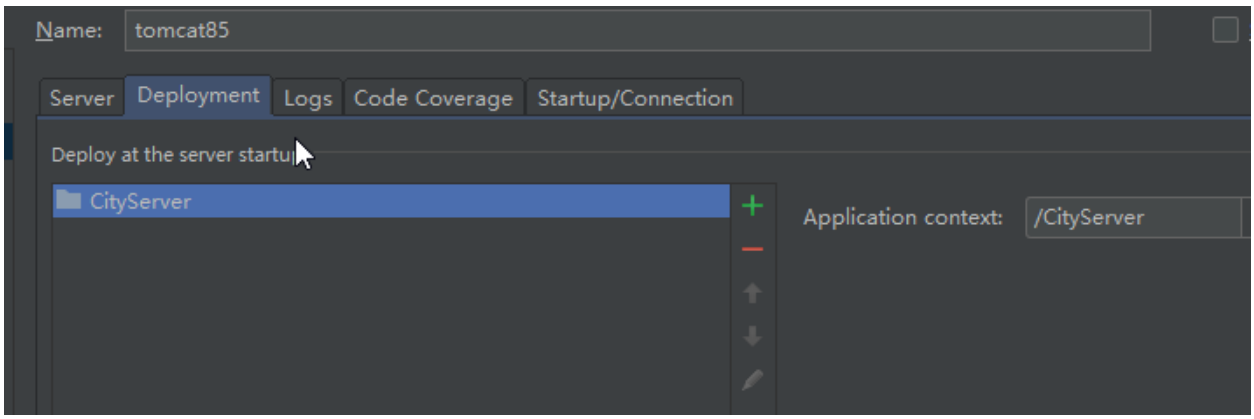


O

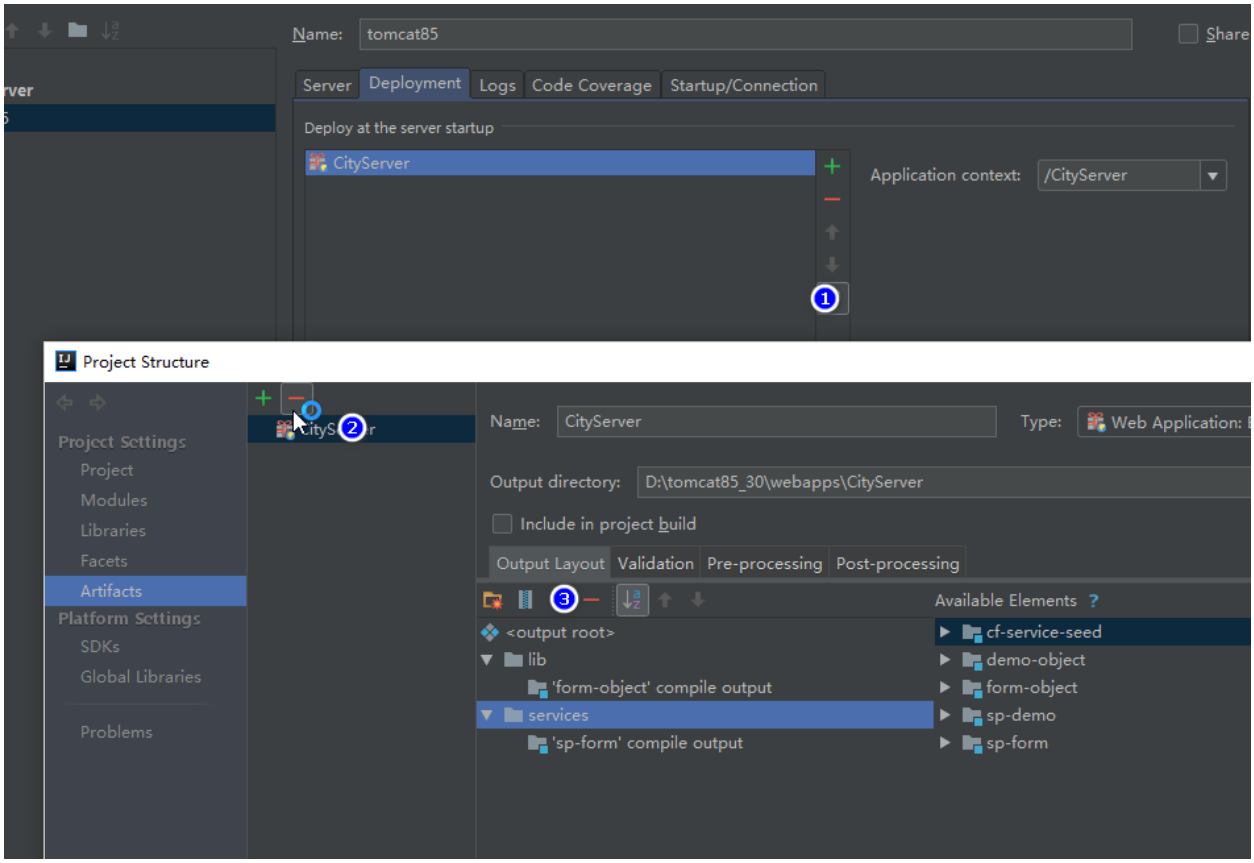
3. Tomcat IDEA下有 local 和 remote 两种模式，选择local



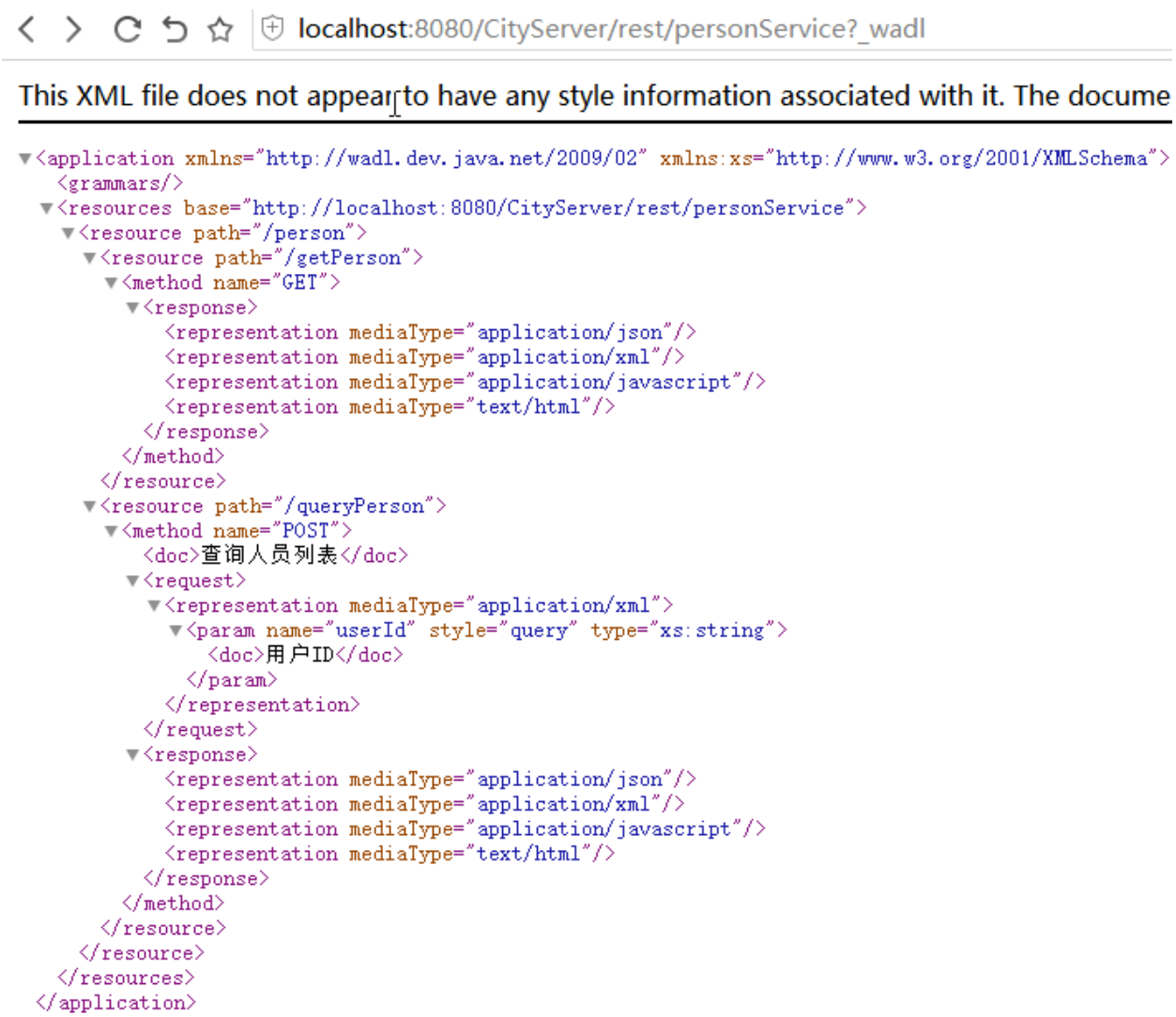
4. 填写Application Context，目前约定使用cityserver



5. 添加编译关联


Name: tomcat85
Share
Server
Deployment
Logs
Code Coverage
Startup/Connection
Deploy at the server startup
CityServer
Application context: /CityServer
1
Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
2
Name: CityServer
Type: Web Application:
Output directory: D:\tomcat85_30\webapps\CityServer
Include in project build
Output Layout
Validation
Pre-processing
Post-processing
3
Available Elements
<output root>
lib
'form-object' compile output
services
'sp-form' compile output
cf-service-seed
demo-object
form-object
sp-demo
sp-form

6. 验证



附录

附录一 Git介绍

1. **分布式管理更可靠**

git是分布式的，svn是集中式的。什么是集中式？以git举例来说，有一个远程服务器来托管代码，同时本地机器也是一个服务器。优点就是当远程服务器出现问题时，可以将本地服务器推送到远程，这样远程服务器不会丢失任何东西。所以git提交代码分为两部分，先要执行commit命令，将代码提交到本地服务器，然后通过push命令将本地服务器代码推送到远程服务器。

2. **更简单的分支操作**

更简单的分支操作。分支的作用是方便多个团队协同合作。默认分支为master分支。我们一般讲master分支做为稳定分支，在各自分支开发完成，测试通过，将代码合并到master分支，在master分支出版本。git合并分支很简单，运行一下pull命令即可，pull命令等同于fetch+merge。 当有新的任务或者临时修改BUG可以快速的切换代码。

关于git的操作教程：git教程

附录二 Maven介绍

现象:在实际开发或者学习中你可能遇到过下面的这些问题

> - 同样的代码，为什么在别人那里可以正常编译和运行，拷贝到我本地之后就报错了呢?
> - 在使用其他技术的时候需要导入一些jar包，有可能你导入的这些jar包又依赖于另一个技术的jar包，你还需要导入这些jar包。
> - 随着项目中使用技术的增多，项目中的jar包也越来越多，这样就会可能会存在一些jar包的冗余。
> - 你自己编写了一款jar包，在公司内部有多个项目使用了这块jar包，倘若某天你发现该jar包存在bug，修正后你需要把这个jar包更新到所有相关的项目中。

1.什么是Maven

Maven是Apache旗下一款开源自动化的项目管理工具，它使用java语言编写，因此Maven是一款跨平台的项目管理工具。 Maven主要功能：

- 项目构建
  在实际开发中，不仅仅是写完代码项目就算完成了，后面还有一些诸如：编译，打包，部署等工作要做，这些工作都可以使用maven来完成。
- 依赖管理
  说的简单一点就是对jar包的管理，开发者不用再手动的下载所需要的jar包，而是将想要的jar包通过配置一个叫做pom.xml的文件中，之后maven会自动的下载相关的jar包。

2.Maven的安装

1. 下载
   你可以通过maven的官网下载：http://maven.apache.org/
   注意：在安装前请确保机器上已经安装了jdk，并且jdk的版本最好是7以上的。
2. 解压
   将maven解压，解压的目录中最好不要含有空格、中文或者其他特殊符号。
   解压后目录如下：
   bin：maven的命令
   boot：含有一个类加载器，通常情况下不使用
   conf：maven的配置文件
   lib：maven的jar包，这里是maven运行时需要的jar包，并非用户在项目中的jar包
3. 配置maven环境变量
   添加一个环境变量：
   变量名：MAVEN_HOME
   变量值：填写你的maven的解压目录，我本地的是：D:\apache-maven-3.5.2
   之后在path中添加;%MAVEN_HOME%\bin
   注意前面使用";"与其他值隔开。
4. 验证是否配置成功
   在cmd中输入mvn -v
   如果显示出当前mvn的版本号，则说明maven的安装成功

3.maven配置

安装完成后分为`默认配置`和`用户配置` **默认配置** 在maven的安装目录的conf文件夹下有一个settings.xml文件，打开后，可以看到有一项：

```
<!-- localRepository
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
```

该配置是默认注释掉的，其意思是默认情况下，maven仓库的目录地址是在你的${user.home}/.m2/repository文件中，我的地址是：C:\Users\Administrator.m2\repository。${user.home}表示的是你本地电脑的用户名。 你可以在下面自己写一个localRepository标签来为其指定一个目录。 maven仓库目录，就是maven将你项目中所用到的jar包下载的目录地址。

用户配置　　可能在你的Windows操作系统中有多个用户，你可以为每个用户设定一个该用户自己的maven仓库地址，即在该用户的的.m2文件夹下复制一份settings.xml文件，然后在文件中指定其仓库地址。

　　如果设置了用户配置，则默认配置会失效。