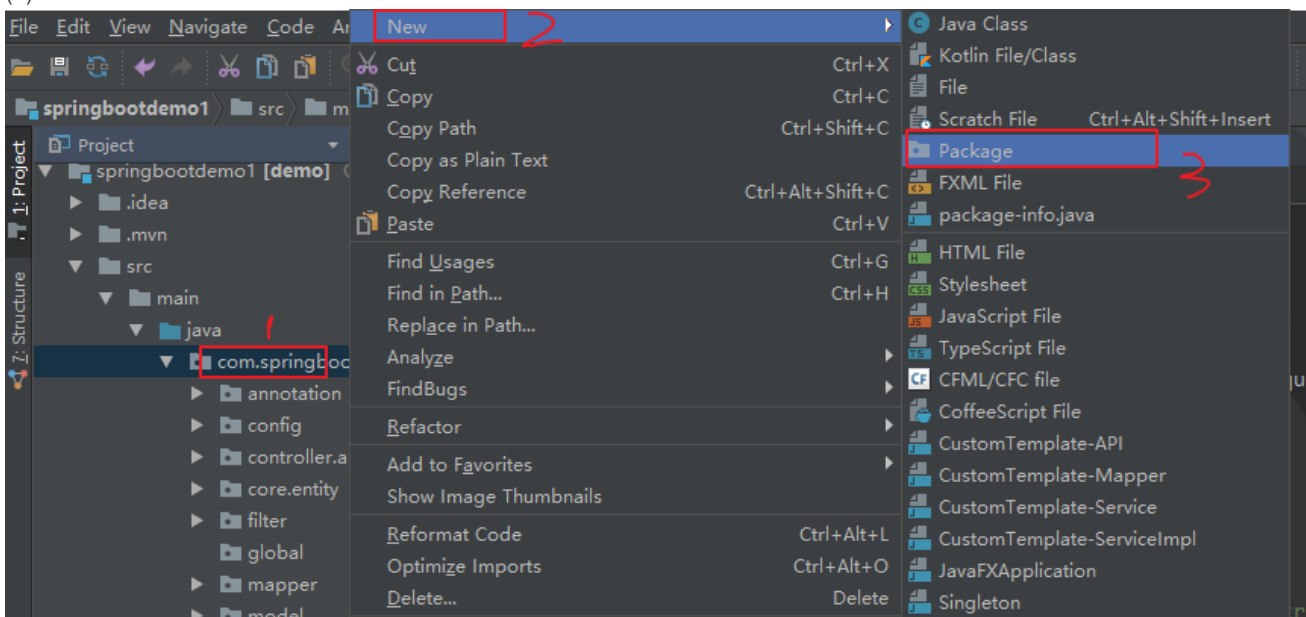
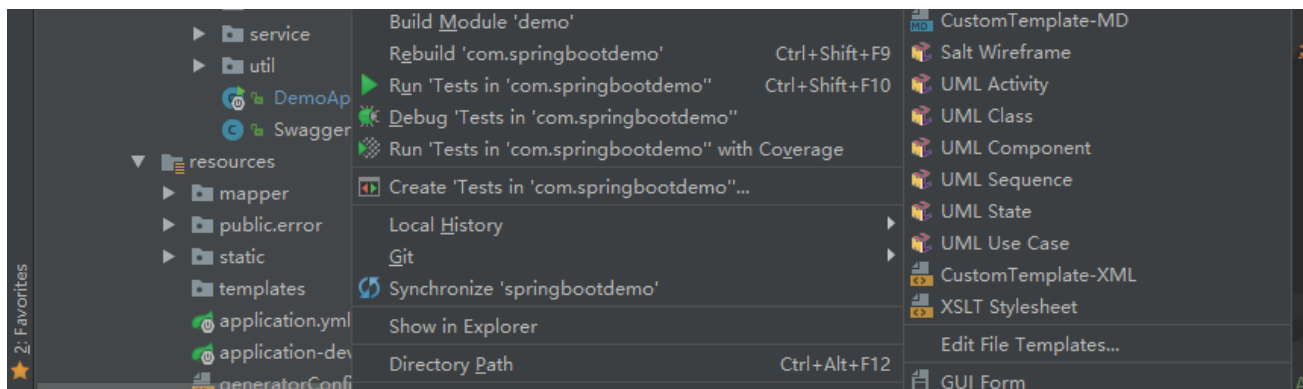


## springboot后台接口开发指导

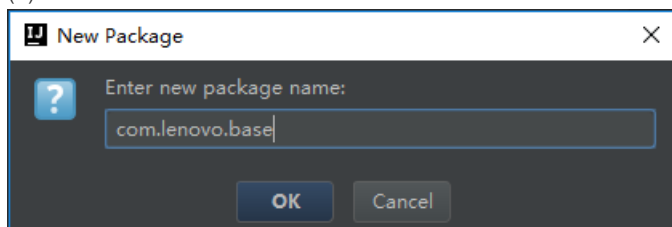
## 1.分层（以包package形式体现）





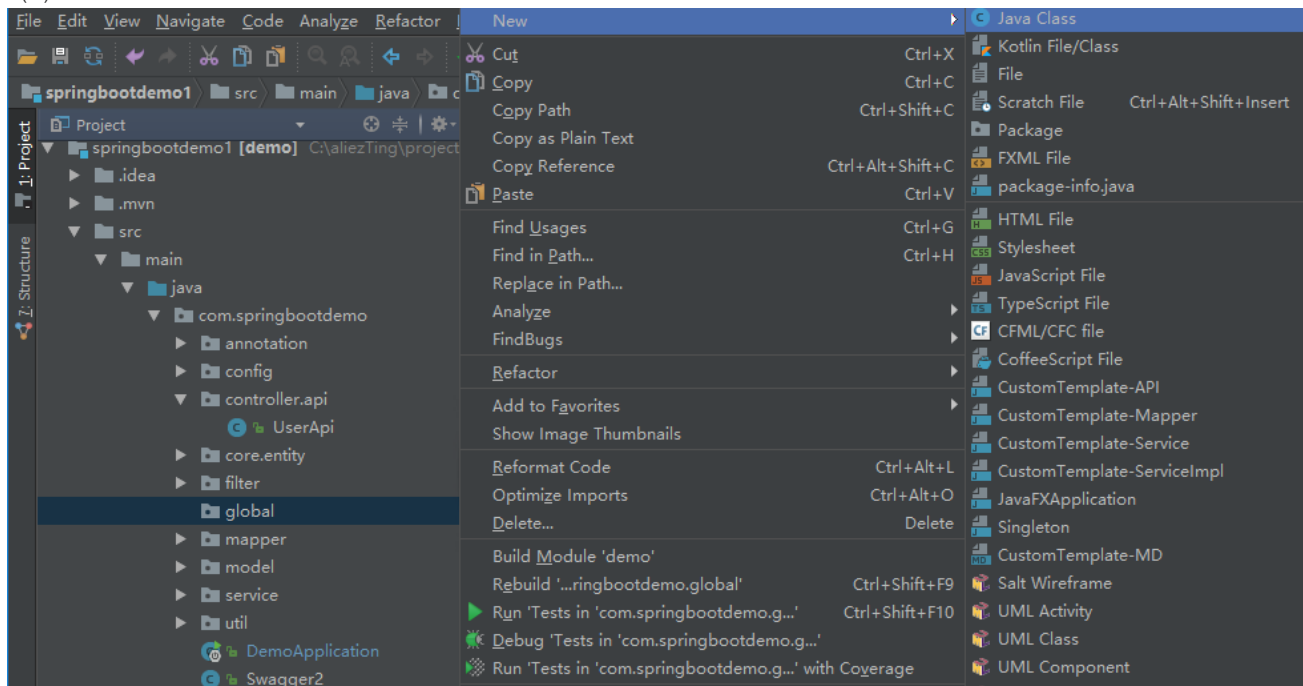
右键项目或者父包名称 (1) --> new新建(2) --> 点击package(3)

(2)

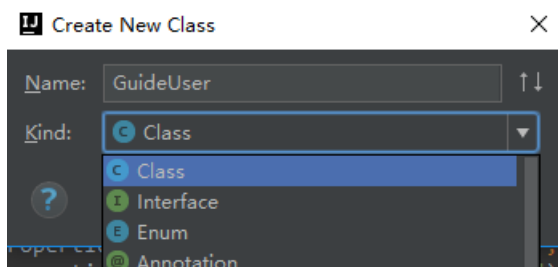


键入包名 (包名全小写中间可以用.隔开, 如果是公司域名以倒过来写, 最后是功能描述)

(3)



建类 (class) 或者接口(interface)、枚举 (enum)



类名遵循骆驼命名法

特殊：实现类的建立，如UserService类有以下方法

```
public interface UserService {

    /**
     * 获取用户列表。
     */
}
```

```

/**
 * @return 列表
 */
List<User> getUsers(int offset, int limit, Map<String, Object> condition);

```

则实现类UserServiceImpl如下：

```

26 public class UserServiceImpl implements UserService {
Class 'ServiceImpl' must either be declared abstract or implement abstract method 'getUsers(int, int, Map<String, Object>)' in 'UserService'
public class UserServiceImpl implements UserService {
// private final UserMapper mapper;
// public UserServiceImpl(UserMapper mapper) {
//     this.mapper = mapper;
// }

```

! Select Methods to Implement

- Implement methods
- Make 'ServiceImpl' abstract
- Create Test
- Create subclass
- Unimplement Interface

com.springbootdemo.service.UserService

getUsers(offset:int, limit:int, condition:Map<String, Object>)

☒ Copy JavaDoc

☒ Insert @Override

OK Cancel

```

/**
 * 获取用户列表。
 * @return 列表
 */
@Override
public List<User> getUsers(int offset, int limit, Map<String, Object> condition) {
    //你的实现逻辑代码
    return null;
}

```

(4) 以下注解的意思可理解为组件@Component(都在org.springframework.stereotype包里)

类级别注解

controller层

·@RestController

·@RequestMapping("api/users")

service具体实现类

·@service

mapper接口（映射）

·@Repository

方法级别注解restfulApi风格

·@GetMapping查

·@PutMapping改

·@PostMapping增

·@DeleteMapping删

方法参数注解

·@PathVariable 适合url中的变量，如 {id}

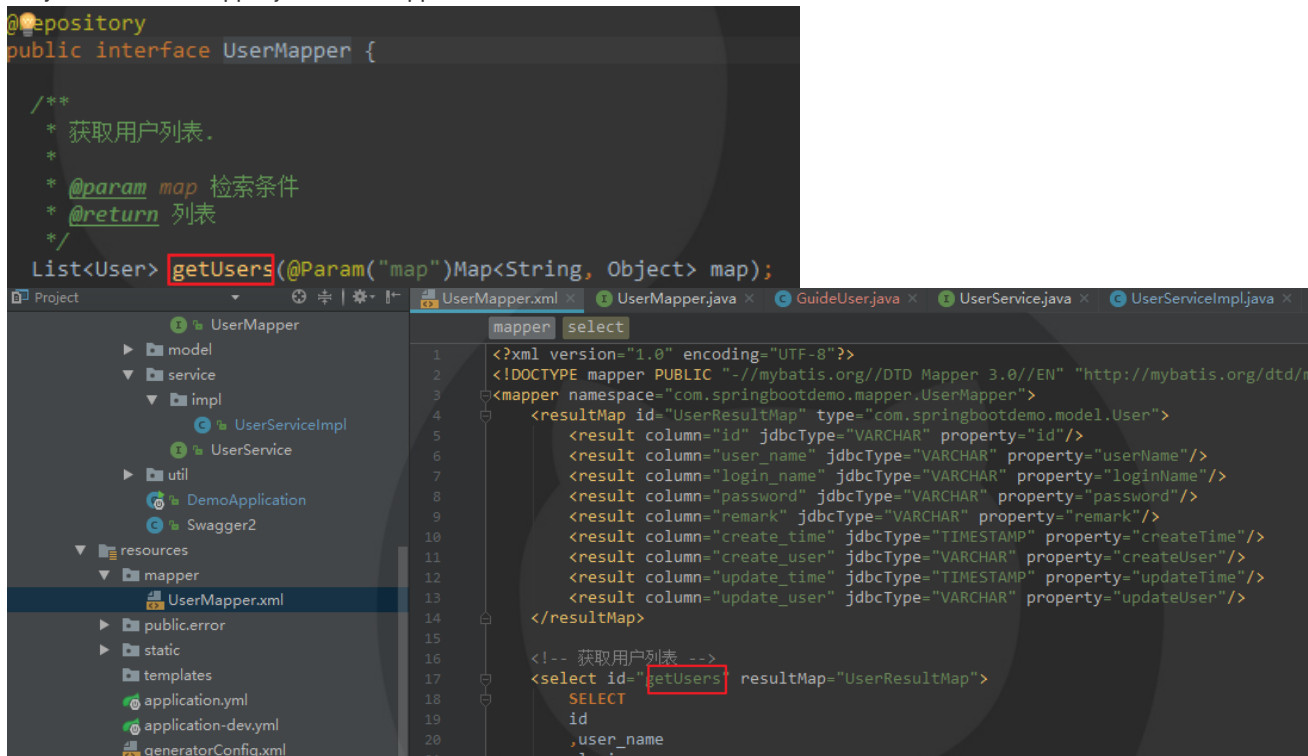
·@RequestBody 适合post方法，获取前端传入的方法体的参数

·@RequestParam 适合get方法，获取头部的参数

附：org.apache.ibatis.annotations.Param;

·@Param mybatis映射参数

## 5.mybatis映射 XxxMapper.java和XxxMapper.xml



java方法名同xml中id名，且必须唯一不重复。

## 2.具体数据流转

从调用者角度分析（自上而下）：

- (1) 一个request请求到后台
- (2) controller拦截找到对应url+type+参数列表
- (3) 进入api特定方法
- (4) 调用service的具体实现类
- (5) 逻辑处理并调用mapper层
- (6) 根据映射找到xml中的sql并进入数据库查询或操作返回结果
- (7) return到response
- (8) 结束

从后台开发者角度分析（自底向上）：

- (1) 编写关键性sql含参数到xml并写出映射mapper
- (2) 业务层和实现类（关键业务逻辑处理，包含mapper层调用）
- (3) 遵循restfulApi规范控制controller层方法入参 返回值（设计小星需求扩展原则）

(3) 遵循RESTful API规范规范URL命名方法/方法参数、返回码（以HTTP标准为准，扩展原则）

(4) api必须唯一不能重复，api命名规范：不能有.和\_，可用-替代

具体可参考<https://www.jianshu.com/p/8b769356ee67>

其他待补充！