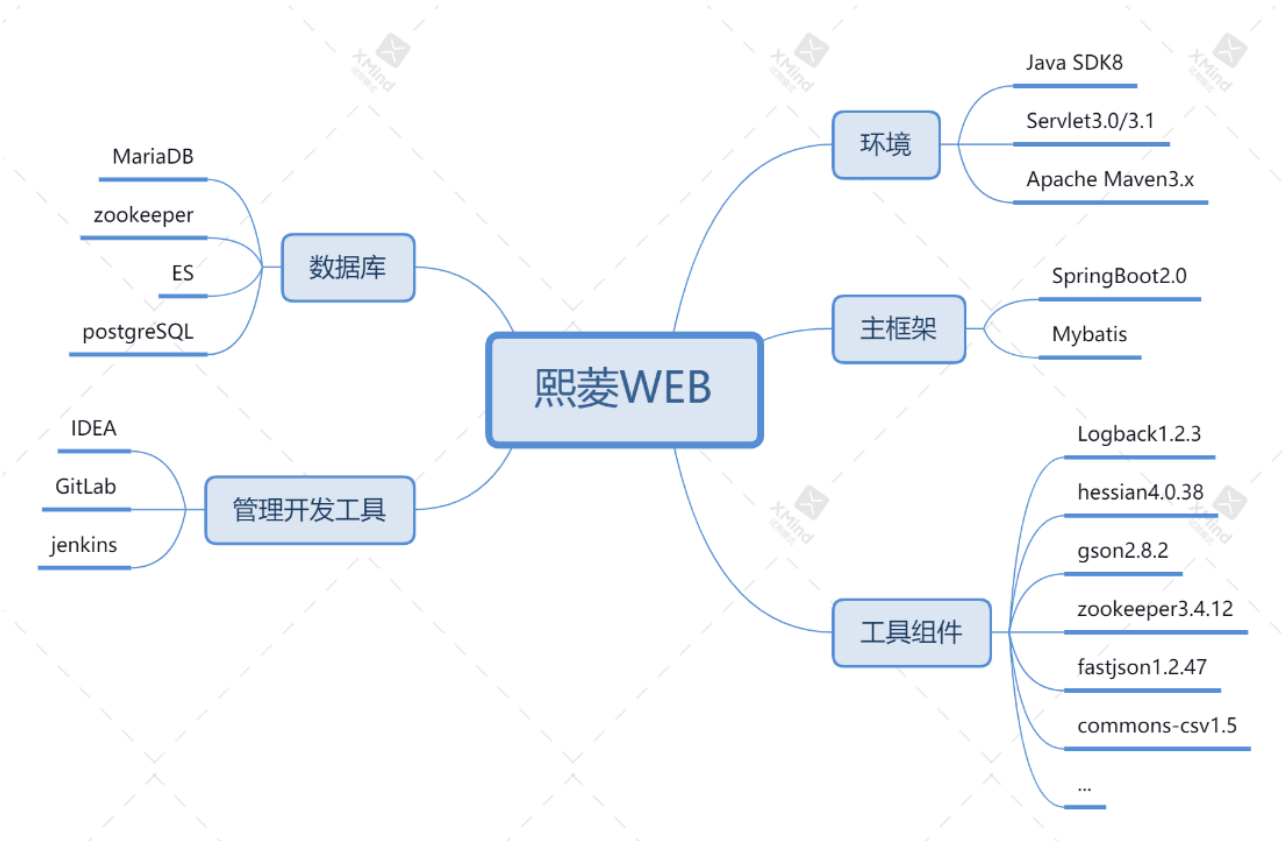
WEB开发文档

> 熙菱WEB后端开发采用SpringBoot为主要框架，数据库基于产品特性采用MariaDB\zookeeper\ES等相结合的方式存储数据。

一、主要技术



二、开发规范

2.1 java基本开发规范

java基本开发规范参考**《阿里巴巴java开发手册1.5》**，该手册已包含java开发中约定的规范。

2.2包的命名规范

设有公司统一的包名：com.sailing包

- 模块格式：com.sailing.分层.模块名.子模块.功能类
  - 分层：分层为controller/service/entity/dao/common/interceptor/config
  - 若模块下只有功能没有子模块，就不需要建立子模块
- 通用封装包或工具类：com.sailing.common.大类.小类.封装类，若无大小类可忽略。

2.3分层命名规范

Controller

操作	命名	备注
类名	以Controller结尾	DemoController
类上的映射路径	/模块/子模块	/demo
自动注入属性	与注入名相同，不允许注入Dao	@Autowired
方法上的映射路径	与方法名相同	@RequestMapping，POST
方法上的操作类型	@OperateType	用于记录用户操作类型、操作菜单
所有列表数据	all	
列表分页数据	list	
列表数据	listData	非分页查询
获取详情数据	get	
新增或编辑数据	save	
新增数据	add	
修改数据	update	
删除	delete	
停用	disable	
启用	enable	
停止	stop	
启动	start	
部署	deploy	
校验数据	check开头	
导入	import开头	
导出	export开头	
树结构数据	treeData	

示例：

```
/**
```

```
 * 模块/子模块路径/功能路径 无子模块可忽略 如：/demo/queryAll
 * 请求方法：POST
 * 功能路径与方法名一致：queryAll
 * 操作类型：@OperateType
 *          operateType：操作的类型
 *          menu：菜单
 */
@RestController
@RequestMapping(value = "/demo")
@CrossOrigin
@Slf4j
public class DemoController {

    @Autowired
    private IDemoService demoService;


    @RequestMapping(value = "/queryAll", method = RequestMethod.POST)
    @OperateType(operateType= OperateMenu.OPERATE_LIST,menu=OperateMenu.MENU_DEMO)
    public RespData<List<Demo>> queryAll() {
        RespData<List<Demo>> respData = new RespData<>();
        try {
            List<Demo> serviceManagers = demoService.queryAll();
            respData.setRespCode(RespCodeEnum.SUCCESS);
            respData.setData(serviceManagers);
        } catch (Exception e) {
            respData.setRespCode(RespCodeEnum.EXCEPTION);
            log.error(e.getMessage(), e);
        }
        return respData;
    }
}
```

Service

操作	命名	备注
接口	以I开头，Service结尾。	继承IBaseService，简单Service可不继承
实现类	以Service结尾	实现接口
查询所有数据	queryAll	
查询多条数据或分页数据	queryList	根据方法重载来确定分页或多条
查询一条数据	get	
新增或编辑	save	
删除	delete	

接口示例：

```java
/**
 * Description:
 * <p>
 * Update by Panyu on 2018/10/12 下午 03:10:54
 */

public interface IDemoService extends IBaseService<Demo> {

}
```

实现类示例：

```java
@Service
@Slf4j
public class DemoService implements IDemoService {

    @Override
    public List<Demo> queryAll() throws Exception {
        //查询所有
        return null;
    }

    @Override
    public PageHelper<Demo> queryList(PageHelper<Demo> pageHelper) throws Exception {
        //分页
        return null;
    }

    @Override
    public List<Demo> queryList(Demo demo) throws Exception {
        //查询多条
        return null;
    }

    @Override
    public Demo get(Demo demo) throws Exception {
        //查询一条
        return null
    }

    @Override
    public RespData<Boolean> save(Demo demo) throws Exception {
        //新增或编辑
        return null;
    }

    @Override
    public Boolean delete(Demo demo) throws Exception {
        //删除
        return true;
    }
```

```
}
```

Dao

Dao层类使用mybatis的注解方式进行数据的增删改操作，一个Dao类操作一个主表数据。

操作	命名	备注
接口类	以I开头(大写的i)，以Dao结尾	
查询一条数据	get	以get开头
插入数据	insert	以 insert 开头
更新数据	update	以 update 开头
删除	delete	以 delete 开头
查询多条数据	以query开头	

示例：

```
@Mapper
public interface ISwapConfigDao {
    @Select("select * from config_swap where serviceId=#{serviceId}")
    SwapConfig get(SwapConfig swapConfig);

    @Insert({ "insert into config_swap(serviceId, serviceName, inResourceCatalogName,
outResourceCatalogName,category) " +
            "values(#{serviceId}, #{serviceName}, #{inResourceCatalogName}, #
{outResourceCatalogName},#{category})" })
    boolean insert(SwapConfig swapConfig);

    @Update("update config_swap set serviceName=#{serviceName},inResourceCatalogName=#
{inResourceCatalogName},outResourceCatalogName=#{outResourceCatalogName},category=#
{category} " +
            "where serviceId=#{serviceId}")
    boolean update(SwapConfig swapConfig);

    @Delete("delete from config_swap where serviceId=#{serviceId}")
    boolean delete(SwapConfig swapConfig);
}
```

Entity

操作	命名	备注
类名	根据表名以驼峰命名法转换	继承BaseEntity，简单Entity可不继承
父类中已有的属性	父类中已有的属性无需定义，特殊情况除外	
非表属性的字段	如查询字段，必须写清楚注释说明用途。	
ZK存储注解	@Node	名称为zk存储节点
Set/Get	@Data	采用Data注解自动set/get

示例：

```
@Data
@Node(name="demo")
public class Demo extends BaseEntity {
    private String name;
    private String text;
}
```

2.4 用户操作记录规范

用户操作记录规范是用户请求接口时记录的相关请求操作的规范。对于增删改等影响数据变化的请求，需要在请求方法上添加**@OperateType**注解。

OperateType接收两个参数：

- operateType：操作类型，主要由OperateMenu提供基本的操作类型，比如：OperateMenu.OPERATE_LIST查询。若基本操作类型不满足，则通过继承OperateMenu来自定义操作类型，类型的命名规范为：OPERATE_操作名称=“操作名称”，例如：OPERATE_EXPORT = "导出";
- menu：功能菜单，各产品根据产品功能菜单通过继承OperateMenu来自定义产品的功能菜单，菜单的命名规范为：MENU_模块名_子模块_功能 =“模块名/子模块/功能”，无子模块可忽略。例如：MENU_DEMO = "演示/演示"

2.5 自定义产品配置参数

配置

公司采用的是SpringBoot框架，SpringBoot有application.properties或application.yml配置文件，需要自定义配置参数时在该配置文件中进行配置，自定义参数以sailing开头。例如：

```
sailing: #自定义的属性和值
  zookeeperIps: 127.0.0.1:2181
  zookeeperTimeout: 30000
```

使用

使用自定义配置参数时，在entity包下的ApplicationProperties类中定义参数名称。例如：

```java
/***
 * application配置文件自定义属性文件
 */
@Component
@Configuration
@ConfigurationProperties(prefix="sailing") //接收application.yml中的sailing下面的属性
public class ApplicationProperties {
    /**zookeeper集群地址*/
    private String zookeeperIps;
    /**zookeeper超时时间*/
    private Integer zookeeperTimeout;

    public String getZookeeperIps() {
        return zookeeperIps;
    }

    public void setZookeeperIps(String zookeeperIps) {
        this.zookeeperIps = zookeeperIps;
    }

    public Integer getZookeeperTimeout() {
        return zookeeperTimeout;
    }

    public void setZookeeperTimeout(Integer zookeeperTimeout) {
        this.zookeeperTimeout = zookeeperTimeout;
    }
}
```

使用时通过自动注入进行获取，例如：

```java
 @Autowired
 private ApplicationProperties applicationProperties;

 applicationProperties.getZookeeperIps()
```

三、Maven管理

公司采用Maven来管理引入jar，公司部署了自己的Maven私服，在开发中以公司maven私服为主，开发的二分库组件，则需要部署到私服库。

地址：http://172.20.52.101:8081/#browse/browse

setting.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <localRepository>D:\java\apache-maven-3.6.1\repository</localRepository>
  <pluginGroups>
  </pluginGroups>
  <proxies>
  </proxies>
  <servers>
    <server>
      <id>nexus</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
    <server>
      <id>hzxl-releases</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
    <server>
      <id>hzxl-snapshots</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
  </servers>
  <mirrors>
        <mirror>
            <id>hzxl</id>
            <name>Company Nexus</name>
            <url>http://172.20.52.101:8081/repository/maven-public/</url>
            <mirrorOf>central</mirrorOf>
        </mirror>
    </mirrors>
  <profiles>
        <profile>
        <id>nexus-dev</id>
        <repositories>
            <repository>
                <id>nexus</id>
                <url>http://172.20.52.101:8081/repository/maven-public/</url>
                <snapshots>
                    <enable>true</enable>
                    <updatePolicy>always</updatePolicy>
                </snapshots>
             </repository>
        </repositories>
      </profile>
    </profiles>
    <activeProfiles>
    <activeProfile>nexus-dev</activeProfile>
  </activeProfiles>
</settings>
```

四、代码管理

公司采用GitLab进行源代码管理。产品工程权限有相关负责人进行授权。

gitLab地址：http://172.20.52.100:8088/

五、任务与问题

公司采用禅道进行计划任务、产品Bug的分配

禅道地址：http://172.20.52.101/zentao

六、其他

6.1 语雀前端知识库

https://www.yuque.com/sailingfront/ipm