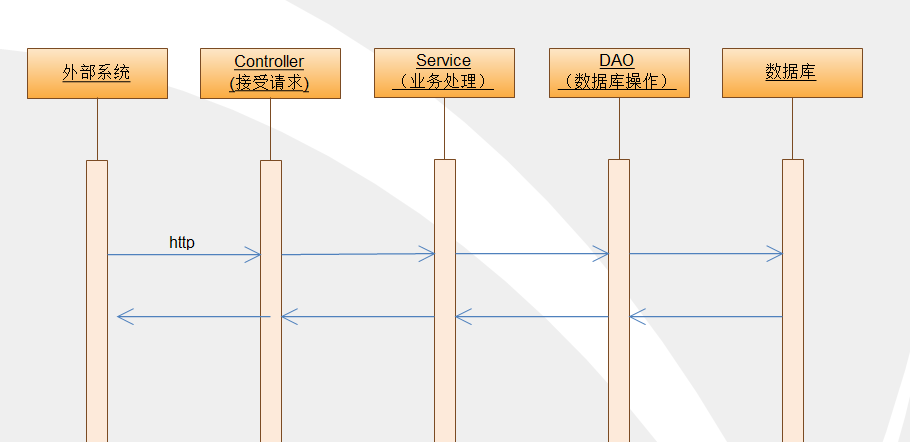
**平台开发手册**

本框架是以SpringMvc+Hibernater为基础搭建的，使用该框架需掌握一定的SpringMvc知识以及Hibernater知识。本系统主要是为外部操作提供操作接口。

1. **模块介绍**

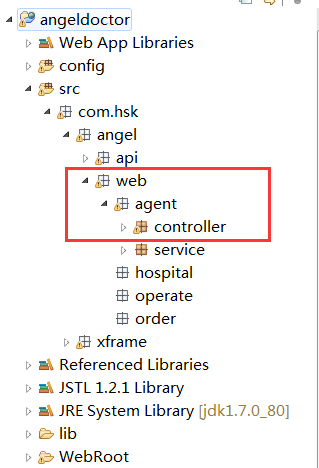


* 外部系统：调用本系统接口。
* Controller：接收外部请求，并将请求结果回传给外部系统。
* Service：对请求进行逻辑处理。
* DAO：对数据库进行增、删、改、查等操作。
* 数据库：指mysql、oracle等关系型数据库。

本系统主要编写：Controller、Service、DAO层。

1. Controller层

Controller层代码主要放在以下目录中：



其中每个实体类会对应一个Controller类

* Controller的命名规则为：实体类名称+Controller如：AgentController；Agent是实体类的名称。
* 每个Controller都必须继承DSTController类。
* Controller类通过@Controller注解注入spring中
* Controller类通过@Autowired注解引入service层接口
* Controller类通过WebUtils.*sendObject*(response, JsonUtil.*getjsonstr*(srm))方法将结果以json格式返回。

以下为Controller层示例：

//controller层注入需要添加的注解

@Controller

**public** **class** AgentController **extends** DSTController {

//引入并自动注入service层接口

@Autowired

**private** IAgentService agentService;//

//创建接口访问地址

@RequestMapping(value = "/testOne.htm")

/\*\*

\*

\* **@param** request

\* **@param** response

\* **@param** user 传入的参数

\* **@throws** HSKException

\*/

**public** **void** testOne(HttpServletRequest request ,HttpServletResponse response,@ModelAttribute Agent agent) **throws** HSKException{

//调用service接口处理逻辑

SysRetrunMessage srm = agentService.saveOrUpdateObject(agent);

//将处理后的数据已json格式返回给系统

WebUtils.*sendObject*(response, JsonUtil.*getjsonstr*(srm));

}

//接口访问地址

@RequestMapping(value = "/admin.htm", method = RequestMethod.***GET***)

/\*\*

\* 页面跳转

\* **@param** request

\* **@return**

\*/

**public** ModelAndView toLogin(HttpServletRequest request){

ModelAndView modelAndView = **new** ModelAndView();

**if**(request.getSession().getAttribute(SystemContext.***sessionUser***) != **null**)

modelAndView.setViewName("/main/main");//跳转的页面地址

**else**

modelAndView.setViewName("/main/login");//跳转的页面地址

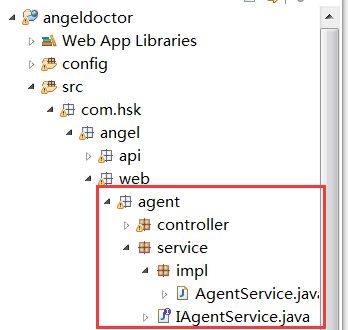
**return** modelAndView;

}

}

1. Service层

Service层代码主要放在以下目录中：



每个service会有一个接口类以及一个接口实现类。接口类中定义有哪些功能，实现类主要是实现接口中定义的功能。

* Service接口的命名规则为：I+实体类名称+Service如：IAgentService；Agent是实体类的名称。
* Service接口实现类的命名规则为: 实体类名称+Service如：AgentService；Agent是实体类的名称。
* 每个接口实现类都继承DSTService类，以及implements相应的接口类。
* Sevice类通过@Service注解注入spring中
* Sevice类通过@Autowired注解引入dao层接口
* 当需要对数据库数据进行增删改时方法名称需要以：add\*、save\*、update\*、del\*、delete\*、do\*、Create\*、create\*、new\*；否则将会报事务错误。

每个 service 中有以下固定的方法：

* public SysRetrunMessage findFormObject(Integer 主键) throws HSKException;//根据主键ID查询实体类，并将其放入SysRetrunMessage中
* public Agent findObject(Integer mdiId) throws HSKException; //根据主键ID查询实体类
* public SysRetrunMessage deleteObject(Integer主键) throws HSKException;//根据主键ID删除实体类
* public SysRetrunMessage deleteAllObject(String 主键组合) throws HSKException; //根据主键ID组合批量删除实体类
* public SysRetrunMessage saveOrUpdateObject(实体类) throws HSKException;//保存并修改实体类信息
* public PagerModel getPagerModelObject (实体类) throws HSKException;//根据实体类查询分页信息
* public List<实体类> getObjectList(实体类) throws HSKException; //根据实体类查询列表信息

以下为接口类列子：

**public** **interface** IAgentService {

/\*\*

\* 查找Agent对象记录，用于弹出添加、修改窗口时候初始化数据

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage findFormObject(Integer mdiId) **throws** HSKException;

/\*\*

\* 查找Agent对象记录，用于弹出添加、修改窗口时候初始化数据

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@return** Agent man\_drug\_info表记录

\* **@throws** HSKException

\*/

**public** Agent findObject(Integer mdiId) **throws** HSKException;

/\*\*

\* 根据man\_drug\_info表主键删除Agent对象记录

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage deleteObject(Integer mdiId) **throws** HSKException;

/\*\*

\* 根据man\_drug\_info表主键删除多条Agent对象记录

\* **@param** mdiIds Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage deleteAllObject(String mdiIds) **throws** HSKException;

/\*\*

\* 新增或修改man\_drug\_info表记录 ,如果man\_drug\_info表主键Agent.mdiId为空就是添加，如果非空就是修改

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage saveOrUpdateObject( Agent att\_Agent) **throws** HSKException;

/\*\*

\*根据Agent对象作为对(man\_drug\_info表进行查询，获取分页对象

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** PagerModel 分页对象

\* **@throws** HSKException

\*/

**public** PagerModel getPagerModelObject (Agent att\_Agent) **throws** HSKException;

/\*\*

\* 获取Agent对象List

\* **@param** att\_Agent

\* **@return**

\* **@throws** HSKException

\*/

**public** List<Agent> getObjectList(Agent att\_Agent) **throws** HSKException;

}

以下为接口实现类列子：

@Service

**public** **class** AgentService **extends** DSTService **implements** IAgentService{

/\*\*

\*业务处理dao类 agentDao

\*/

@Autowired

**protected** IAgentDao agentDao;

/\*\*

\* 根据man\_drug\_info表主键删除Agent对象记录

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage findFormObject(Integer mdiId) **throws** HSKException{

SysRetrunMessage srm=**new** SysRetrunMessage(1l);

**try**{

Agent att\_Agent= agentDao.findAgentById( mdiId) ;

srm.setObj(att\_Agent);

} **catch** (HSKDBException e) {

e.printStackTrace();

srm.setCode(0l);

srm.setMeg(e.getMessage());

}

**return** srm ;

}

/\*\*

\* 根据man\_drug\_info表主键删除Agent对象记录

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@throws** HSKException

\*/

**public** Agent findObject(Integer mdiId) **throws** HSKException{

Agent att\_Agent=**new** Agent();

**try**{

att\_Agent= agentDao.findAgentById( mdiId) ;

} **catch** (HSKDBException e) {

e.printStackTrace();

**throw** **new** HSKException(e);

}

**return** att\_Agent;

}

/\*\*

\* 根据man\_drug\_info表主键删除Agent对象记录

\* **@param** mdiId Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage deleteObject(Integer mdiId) **throws** HSKException{

SysRetrunMessage srm=**new** SysRetrunMessage(1l);

**try**{

agentDao.deleteAgentById(mdiId);

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** HSKException(e);

}

**return** srm;

}

/\*\*

\* 根据man\_drug\_info表主键删除多条Agent对象记录

\* **@param** MdiIds Integer类型(man\_drug\_info表主键)

\* **@return** SysRetrunMessage 对象记录

\* **@throws** HSKException

\*/

**public** SysRetrunMessage deleteAllObject(String mdiIds) **throws** HSKException{

SysRetrunMessage srm = **new** SysRetrunMessage(1l);

**try** {

String[] ids = mdiIds.split(",");

**for** (String did : ids) {

**if** (did != **null** && !"".equals(did.trim())) {

agentDao.deleteAgentById(Integer

.*valueOf*(did));

}

}

} **catch** (NumberFormatException e) {

e.printStackTrace();

HSKException hse = **new** HSKException(e);

hse.setDisposeType("01");

**throw** hse;

} **catch** (HSKDBException e) {

e.printStackTrace();

**throw** **new** HSKException(e);

}

**return** srm;

}

/\*\*

\* 新增或修改man\_drug\_info表记录 ,如果man\_drug\_info表主键Agent.MdiId为空就是添加，如果非空就是修改

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** Agent 修改后的Agent对象记录

\* **@throws** HSKDBException

\*/

**public** SysRetrunMessage saveOrUpdateObject( Agent att\_Agent) **throws** HSKException{

SysRetrunMessage srm = **new** SysRetrunMessage(1l);

**try**{

agentDao.saveOrUpdateAgent(att\_Agent);

srm.setObj(att\_Agent);

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** HSKException(e);

}

**return** srm;

}

/\*\*

\*根据Agent对象作为对(man\_drug\_info表进行查询，获取分页对象

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** PagerModel 分页对象

\* **@throws** HSKException

\*/

**public** PagerModel getPagerModelObject (Agent att\_Agent) **throws** HSKException{

PagerModel pm=**new** PagerModel(**new** ArrayList<Agent>());

**try**{

pm=agentDao.getPagerModelByAgent(att\_Agent);

} **catch** (Exception e) {

e.printStackTrace();

}

**return** pm;

}

@Override

**public** List<Agent> getObjectList(Agent att\_Agent) **throws** HSKException {

List<Agent> reList = **new** ArrayList<Agent>();

**try**{

reList = agentDao.getListByAgent(att\_Agent);

} **catch** (Exception e) {

e.printStackTrace();

}

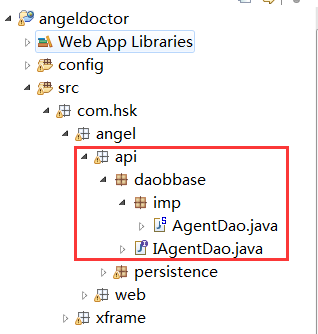
**return** reList;

}

}

1. DAO层

DAO层代码主要放在以下目录中：



每个DAO会有一个接口类以及一个接口实现类。接口类中定义有哪些功能，实现类主要是实现接口中定义的功能。

* DAO接口的命名规则为：I+实体类名称+Dao如：IAgentDao；Agent是实体类的名称。
* Service接口实现类的命名规则为: 实体类名称+Dao如：AgentDao；Agent是实体类的名称。
* 每个接口实现类都继承SupperDao类，以及implements相应的接口类。
* dao类通过@Component注解注入spring中

每个 dao 中有以下固定的方法（Agent为实体类名称）：

* public Agent findAgentById(Integer 主键ID) throws HSKDBException;//根据主键ID查询实体类
* public void deleteAgentById(Integer主键ID) throws HSKDBException;//根据主键删除实体类
* public Agent updateAgentById(Integer 主键ID,Agent att\_Agent) throws HSKDBException;//根据ID修改实体类
* public Integer saveAgent(Agent att\_Agent) throws HSKDBException;//新建实体类返回主键ID
* public Agent saveOrUpdateAgent( Agent att\_Agent) throws HSKDBException;//保存并修改实体类
* public PagerModel getPagerModelByAgent (Agent att\_Agent) throws HSKDBException;//查询实体类分页信息
* public List<Agent> getListByAgent ( Agent att\_Agent) throws HSKDBException;//查询实体类列表信息

dao层中常用的数据库操作方法如下：

* this.getOne(实体类)//获取一个实体类
* this.deleteObject(实体类)//删除实体类
* this.updateObject(实体类)//修改实体类
* this.newObject(实体类)//新建实体类
* this.getHibernateTemplate().saveOrUpdate(实体类)//保存并修改实体类
* this.getHibernateTemplate().find(Hql)//查询hql并返回实体类list
* this.getHibernateDao().findByPage(Hql)//查询回去了并返回分页信息
* this.getJdbcDao().execute(sql);//执行sql语句
* this.getJdbcDao().query(sql);//查询sql语句并返回list<Map>

以下为接口类例子

/\*\*

\* agent表数据库层面操作接口类

\* **@author** 作者:admin

\* **@version** 版本信息:v1.0 创建时间: 2018-06-22 16:59:49

\*/

**public** **interface** IAgentDao{

/\*\*

\* 根据agent表主键查找Agent对象记录

\* **@param** MdiId Integer类型(agent表主键)

\* **@return** Agent agent表记录

\* **@throws** HSKDBException

\*/

**public** Agent findAgentById(Integer MdiId) **throws** HSKDBException;

/\*\*

\* 根据agent表主键删除Agent对象记录

\* **@param** MdiId Integer类型(agent表主键)

\* **@throws** HSKDBException

\*/

**public** **void** deleteAgentById(Integer MdiId) **throws** HSKDBException;

/\*\*

\* 根据agent表主键修改Agent对象记录

\* **@param** MdiId Integer类型(agent表主键)

\* **@param** att\_Agent Agent类型(agent表记录)

\* **@return** Agent 修改后的Agent对象记录

\* **@throws** HSKDBException

\*/

**public** Agent updateAgentById(Integer MdiId,Agent att\_Agent) **throws** HSKDBException;

/\*\*

\* 新增agent表 记录

\* **@param** att\_Agent Agent类型(agent表记录)

\* **@return** agent表主键值

\* **@throws** HSKDBException

\*/

**public** Integer saveAgent(Agent att\_Agent) **throws** HSKDBException;

/\*\*

\* 新增或修改agent表记录 ,如果agent表主键Agent.MdiId为空就是添加，如果非空就是修改

\* **@param** att\_Agent Agent类型(agent表记录)

\* **@return** Agent 修改后的Agent对象记录

\* **@throws** HSKDBException

\*/

**public** Agent saveOrUpdateAgent( Agent att\_Agent) **throws** HSKDBException;

/\*\*

\*根据Agent对象作为对(agent表进行查询，获取分页对象

\* **@param** att\_Agent Agent类型(agent表记录)

\* **@return** PagerModel 分页对象

\* **@throws** HSKDBException

\*/

**public** PagerModel getPagerModelByAgent (Agent att\_Agent) **throws** HSKDBException;

/\*\*

\*根据Agent对象作为对(agent表进行查询，获取分页对象

\* **@param** att\_Agent Agent类型(agent表记录)

\* **@return** List<Agent> 列表对象

\* **@throws** HSKDBException

\*/

**public** List<Agent> getListByAgent ( Agent att\_Agent) **throws** HSKDBException;

}

以下为接口实现类例子

/\*\*

\* man\_drug\_info表数据库层面操作实现类

\* **@author** 作者:admin

\* **@version** 版本信息:v1.0 创建时间: 2018-06-22 16:59:49

\*/

@Component

**public** **class** AgentDao **extends** SupperDao **implements** IAgentDao {

/\*\*

\* 根据man\_drug\_info表主键查找Agent对象记录

\*

\* **@param** MdiId Integer类型(man\_drug\_info表主键)

\* **@return** Agent man\_drug\_info表记录

\* **@throws** HSKDBException

\*/

**public** Agent findAgentById(Integer MdiId) **throws** HSKDBException{

Agent att\_Agent=**new** Agent();

**if**(MdiId!=**null**){

att\_Agent.setMdiId(MdiId);

Object obj= **this**.getOne(att\_Agent);

**if**(obj!=**null**){

att\_Agent=(Agent) obj;

}

}

**return** att\_Agent;

}

/\*\*

\* 根据man\_drug\_info表主键删除Agent对象记录

\* **@param** MdiId Integer类型(man\_drug\_info表主键)

\* **@throws** HSKDBException

\*/

**public** **void** deleteAgentById(Integer MdiId) **throws** HSKDBException{

Agent att\_Agent=**new** Agent();

**if**(MdiId!=**null**){

att\_Agent.setMdiId(MdiId);

Object obj= **this**.getOne(att\_Agent);

**if**(obj!=**null**){

**this**.deleteObject(obj);

}

}

}

/\*\*

\* 根据man\_drug\_info表主键修改Agent对象记录

\* **@param** MdiId Integer类型(man\_drug\_info表主键)

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** Agent 修改后的Agent对象记录

\* **@throws** HSKDBException

\*/

**public** Agent updateAgentById(Integer MdiId,Agent att\_Agent) **throws** HSKDBException{

**if**(MdiId!=**null**){

att\_Agent.setMdiId(MdiId);

Object obj= **this**.getOne(att\_Agent);

**if**(obj!=**null**){

**this**.updateObject(obj);

}

}

**return** att\_Agent;

}

/\*\*

\* 新增man\_drug\_info表 记录

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@throws** HSKDBException

\*/

**public** Integer saveAgent(Agent att\_Agent) **throws** HSKDBException{

**return** **this**.newObject(att\_Agent);

}

/\*\*

\* 新增或修改man\_drug\_info表记录 ,如果man\_drug\_info表主键Agent.MdiId为空就是添加，如果非空就是修改

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@throws** HSKDBException

\*/

**public** Agent saveOrUpdateAgent(Agent att\_Agent) **throws** HSKDBException{

**this**.getHibernateTemplate().saveOrUpdate(att\_Agent);

**return** att\_Agent;

}

/\*\*

\*根据Agent对象作为对(man\_drug\_info表进行查询，获取列表对象

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** List<Agent> 列表对象

\* **@throws** HSKDBException

\*/

@SuppressWarnings("unchecked")

**public** List<Agent> getListByAgent(Agent att\_Agent) **throws** HSKDBException{

String Hql=**this**.getHql( att\_Agent);

List<Agent> list=**this**.getHibernateTemplate().find(Hql);

**return** list;

}

/\*\*

\*根据Agent对象作为对(man\_drug\_info表进行查询，获取分页对象

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** List<Agent> 分页对象

\* **@throws** HSKDBException

\*/

**public** PagerModel getPagerModelByAgent(Agent att\_Agent)

**throws** HSKDBException {

String Hql=**this**.getHql(att\_Agent);

**return** **this**.getHibernateDao().findByPage(Hql);

}

/\*\*

\* 根据Agent对象获取Hql字符串

\* **@param** att\_Agent Agent类型(man\_drug\_info表记录)

\* **@return** hql字符串

\*/

**private** String getHql(Agent att\_Agent){}

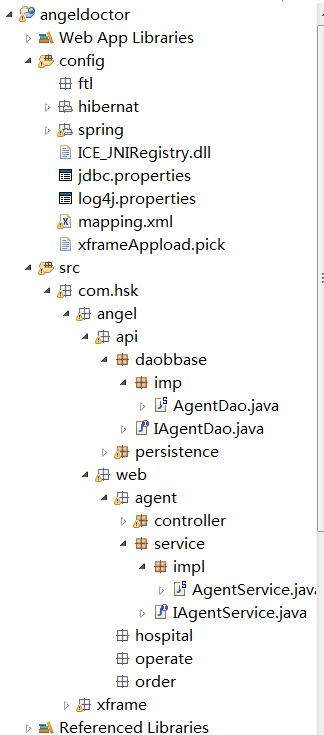
**}**

1. 常用类介绍

* SysRetrunMessage：向其他系统返回规范数据类该类有以下属性：code:0代表失败、1代表成功；obj:返回的对象和列表；meg：返回的消息。
* PagerModel：分页实体类。total：总数；rows:数据列表；items：数据列表。
* CreateCodeUtil：编号自动生成类。
* ExportExcel：导出excel工具类：

1. 项目结构介绍。

项目名称



Service层代码

Controller层代码

实体类代码与数据库表对应

Dao层代码

数据库连接地址