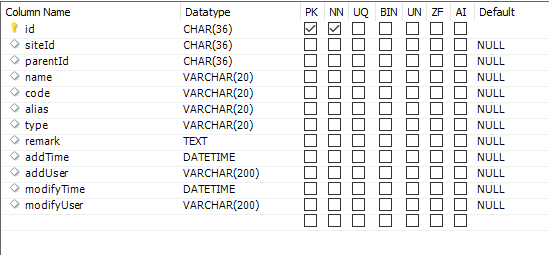
**1、表结构：**



**2、表对应的实体类：**

**public** **class** CmsColumn **implements** Serializable {

**private** **static** **final** **long** *serialVersionUID* = 1L;

**private** String id;

**private** String siteId;

**private** String parentId;

**private** String name;

**private** String code;

**private** String alias;

**private** String type;

**private** String remark;

**private** Date addTime;

**private** String addUser;

**private** Date modifyTime;

**private** String modifyUser;

/\*\*

\* 下面写字段对应的getter和setter方法

\*/

}

//实体类字段名与表字段名对应，首字母小写。实体类字段名与maper配置文件引用字段要保持大小写一致。

**3、mapper的接口类**

**public** **interface** CmsColumnMapper {

List<CmsColumn> selectAll(Map<String, Object> params, RowBounds rowBounds);

**void** insert(CmsColumn bean);

**void** update(CmsColumn bean);

**void** delete(CmsColumn bean);

}

//接口类定义了增删改查4个方法，方法名与配置文件的id保持一致，方法参数类型与配置文件的parameterType一致。

**4.mapper的配置文件与mapper接口类同名：CmsColumnMapper.xml**

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

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"cc.isystem.cms.persistence.mybatis.mapper.CmsColumnMapper"*>

<select id=*"selectAll"* parameterType=*"map"* resultType=*"CmsColumn"*>

SELECT \* FROM cms\_column

<where>

<if test=*"id != null"*>

AND id = #{id}

</if>

<if test=*"siteId != null"*>

AND siteId = #{siteId}

</if>

<if test=*"parentId != null"*>

AND parentId = #{parentId}

</if>

<if test=*"code != null"*>

AND code = #{code}

</if>

<if test=*"alias != null"*>

AND alias = #{alias}

</if>

<if test=*"type != null"*>

AND type = #{type}

</if>

</where>

<choose>

<when test=*"desc != null"*>

ORDER BY ${desc} DESC

</when>

<when test=*"asc != null"*>

ORDER BY ${asc} ASC

</when>

<otherwise>

ORDER BY code ASC

</otherwise>

</choose>

</select>

<insert id=*"insert"* parameterType=*"CmsColumn"*>

INSERT INTO cms\_column (id,siteId,parentId,name,code,alias,type,remark,addTime,addUser,modifyTime,modifyUser)

VALUES (#{id},#{siteId},#{parentId},#{name},#{code},#{alias},#{type},#{remark},#{addTime},#{addUser},#{modifyTime},#{modifyUser})

</insert>

<update id=*"update"* parameterType=*"CmsColumn"*>

UPDATE cms\_column SET

siteId=#{siteId},parentId=#{parentId},name=#{name},code=#{code},alias=#{alias},type=#{type},remark=#{remark},addTime=#{addTime},addUser=#{addUser},modifyTime=#{modifyTime},modifyUser=#{modifyUser}

WHERE id=#{id}

</update>

<delete id=*"delete"* parameterType=*"CmsColumn"*>

DELETE FROM cms\_column WHERE id=#{id}

</delete>

</mapper>

**5、DAO类**

@Repository

**public** **class** CmsColumnDao **extends** AbstractDao<CmsColumn> {

@Autowired

**private** CmsColumnMapper cmsColumnMapper;

**public** Pagination<CmsColumn> find(Map<String, Object> params,

**int** pageIndex, **int** pageSize) {

**return** **new** Pagination<CmsColumn>(cmsColumnMapper.selectAll(params,

**new** RowBounds(pageIndex, pageSize)));

}

**public** **void** create(CmsColumn bean) {

cmsColumnMapper.insert(bean);

}

**public** **void** update(CmsColumn bean) {

cmsColumnMapper.update(bean);

}

**public** **void** delete(CmsColumn bean) {

cmsColumnMapper.delete(bean);

}

}

//DAO类继承AbstractDao 类，并对mapper接口的方法进行封装

**6、Server类**

@Service

**public** **class** CmsColumnService **extends** AbstractService<CmsColumnDao, CmsColumn> {

@Autowired

**private** CmsColumnDao cmsColumnDao;

@Override

**protected** CmsColumnDao getMyDao() {

**return** cmsColumnDao;

}

}

//Server类继承AbstractService，基础方法已在AbstractService实现，server类中主要添加业务方法。