易为平台—权限系统开发指南

**易为开发团队**

**2010年8月**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **版本号** | | | **V1.0** | | | | **发布时间** | | **2010-12-6** | |
| **主要编写(修订)人员** | | | | | | | | | | |
| **序号** | **姓名** | | | **编写内容** | | | | **Email** | | **QQ** |
| 1 | 兰天 | | | 第1,2章 | | | | Ltian.cn@gmail.com | | 595534098 |
| 2 | 姜建卫 | | | 第3，4章 | | | |  | |  |
|  |  | | |  | | | |  | |  |
|  |  | | |  | | | |  | |  |
|  |  | | |  | | | |  | |  |
|  |  | | |  | | | |  | |  |
| **修订记录** | | | | | | | | | | |
| **序号** | | **修订日期** | | | **修订人** | **修订说明** | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |
|  | |  | | |  |  | | | | |

# 权限系统概述

权限系统，一般也称为访问控制(Access Control)系统，是信息系统的重要组成部分，尽管（美国）国家技术与标准委员会（NIST）制定了基于角色的访问控制模型（RBAC）标准,但是由于各种系统的访问控制需求各不相同，RBAC标准相当开放，RBAC没有规定，也无法规定访问控制系统的实现细节。因此，有各种不同方式基于RBAC标准的访问控制系统出现，用于各种不同领域的权限访问控制。

在国内，通用且好用的权限管理系统不多，一般都或多或少地存在以下问题：

1. 有的权限控制比较简陋，仅能控制菜单的访问性，无法深入控制数据访问和界面上具体控件的使能或者隐现。
2. 很多权限控制逻辑依赖于操作者所在的部门。
3. 数据访问权限控制逻辑与业务逻辑分不清，或者权限逻辑过多地侵入到了SQL语句层面，使得最终用户的管理员无法理解权限如何授予，同时无法管理同关系数据无关的数据访问权限。
4. 组织机构模型不够全面，仅仅树形的组织机构模型无法满足企业级权限系统需求。
5. 角色模型是列表状，而非层级结构，不符合现实中的角色模型，增加了工作流系统中业务流程定义的复杂程度。

本权限系统分析了以上问题，并结合开发者们多年企业应用开发的经验，开发了本权限系统。

正如上面我们所提到的，NIST RBAC标准也认为不存在能够全面解决所有权限问题的权限系统。读者一定要认识到：**本权限系统不能够全面解决所有复杂的数据访问问题，因为有些少数的数据访问逻辑看起来更像业务逻辑**，比如：入职不足两年的员工不能访问重要客户信息这样的问题。类似这样可以说是权限也可以说是业务逻辑的问题，我们当前建议将其看做业务逻辑来处理，在未来，或许可以将其当做权限来处理。

那么，本权限系统的优点在哪里呢？本权限系统的最大特点是：**在数据权限访问控制方面，找到了一条既适用于开发者，又适用于最终用户管理员的数据访问控制方法。**通过本权限系统，可以简单、灵活地控制目前我们所遇到的绝大多数的数据访问。

我们认为，通过控制数据库表的行与列的访问来控制数据的访问，很难被最终用户的系统管理员所掌握，因为系统管理员不可能了解每个表和字段的含义，同时也不知道这些表和字段被那些人机界面所引用。除此之外，该种粒度的控制方式必然深入侵入到数据库访问层面，很多类似的权限系统要求用户的SQL语句执行交由权限系统来执行，并将其命名为“数据资源引擎”。而理想情况下，数据的提取是根据业务需求设计的，强制的“数据资源引擎”会打乱或干扰业务逻辑的设计思路。

因此，在本权限系统中，我们将数据资源访问控制的重点放在数据资源访问的参数约束上。在大多数情况下，我们发现，对数据资源的访问控制都是通过控制访问数据的参数进行限制的，比如思考以下场景：

1. 物资管理中，限定某个操作者只能访问特定“**仓库**”中的**物资**。
2. 人事管理中，限定某个操作者只能访问特定“**部门**”中的**人员**。
3. 学籍管理中，限定某个操作者只能访问特定“**班级**”中的**学生**。
4. 财务管理中，限定某个操作者智能访问特定“**账套**”下的**账务**。

在上述场景中，我们发现，**物资、人员、学生、账务**是我们要最终控制的数据资源，但在实际操作中，却通过**仓库、部门、班级、账套**这些参数来对这些数据资源进行控制。在所有需要控制数据资源访问控制的业务中，通过这种方式进行数据资源访问控制的业务占95%以上。而且实际上，企业为了简化管理，也往往都是利用诸如**“仓库”、“部门”、“班级”、“账套”**等资源来控制其他资源，**通过这种方式控制权限也能够被系统管理人员所理解和接受**。在这种方式下，“部门”则是最为常见的一种控制参数，因此，有些权限管理系统直接将数据资源的访问控制与操作者所在部门挂钩也解决了相当多的问题。

本权限系统，主要是通过控制资源访问参数来实现对数据资源的管控，并将这些资源称为“受控资源”，同时本权限系统初步实现了RBAC标准中的层级角色，并且在组织架构模型上有所突破。通过本文，希望读者能了解本权限系统的思想和使用。

# 权限系统概念与模型

## 2.1组织机构

组织机构是信息系统的基础构成部分，可以不被当做权限系统的一部分，但无论组织机构管理是否属于权限，我们都应重视企业组织机构的模型。

通常情况下，组织机构是一个树状的模型。但是对于大型集团型起来说，从信息管理角度来看，认为组织结构是一棵单一的树无法满足应用的要求。比如：大型企业往往会向下级企业派驻财务分支机构，这样，对于财务系统来讲，由总部财务和分总司的财务部门构成了财务组织结构。但是从人事管理系统的角度来看，总部设在分公司的财务部门隶属于分公司的人事管理。因为管理角度的不同，部门的组织机构所承担的角色也不同，其上下级关系不同。

权限系统开发者从2005年开始认识到并思考了该问题，遗憾的是由于当时水平有限，并没有将组织结构的模型抽象的更完美，虽然我们在物理模型（数据库表层）上进行了支持，我们将在今后的版本中予以彻底解决，并保持对现有系统的兼容。

在当前的版本中，我们的组织机构有如下缺点：

1. ***没有将组织机构的视图作为单独的对象进行提炼。***
2. ***认为组织机构对象、人员对象仍可以扩展，即可能有不同的子类来实现组织机构和人员。***

因此，本版本的组织机构管理虽然有突破性的创新思想，但在实现该思想时由于时间和当时技术水平的关系，没能够将组织机构的API设计完美，**将简单的问题复杂化，并使得系统的配置变得较为复杂。**

权限系统开发者认为，如果针对中小企业或者开发单一的系统，开发者仍然可以将组织机构看作为一棵树。待将来本权限系统升级后对引用了组织机构的代码进行重构（实际上，需要重构的代码量并不大，对组织机构引用最多的是权限、人事等系统）才能更好支持多视图的组织机构模型。因此，本系统作者建议开发者了解了理想中的组织机构模型后，暂时不要深入研究本系统的组织机构模型，在使用中不要更改其系统配置。

因此，我们建议开发者可以借鉴组织机构模型的思想，但是在实际开发中保持组织机构现状不变，当前组织机构的现状如下：

1. **散列的组织机构。**SYS\_DEPT表中存储了所有的组织机构的基本，是组织机构信息的来源。该表各字段详见PDM文件。该表中不描述组织机构之间的上下级关系。

2.**只能很好地支持base和right两种组织机构视图**。SYS\_DEPT\_RELATION表持久化了不同视图下组织机构之间的上下级关系。VIEW\_TYPE字段存储的就是组织机构视图类型，视图类型是一个字符串。目前，系统只保证两种类型的组织机构视图可用:一个是**base**，表示基准组织机构视图，一个是**right**，表示权限组织机构视图。开发者目前不要更改这两个视图类型和配置。表示组织机构视图类型的字符串也出现在权限配置文件（WEB-INF/appconfig/com-xdf-right-config.xml）中的如下位置（红色加粗部分）：

<?xml version="1.0" encoding="GBK" ?>

<application packagename="com.xdfsoft.right" displayname="新东方权限系统">

<base-organize>

<store type="db" ref="com.xdfsoft" />

**<dept-view-type>base</dept-view-type>**

<emp-view-type>right</emp-view-type>

<cache-initializer>

com.xdfsoft.organize.base.daoimpl.OrganizeCacheDbInitializer

</cache-initializer>

<dept-dao-class>

com.xdfsoft.organize.base.daoimpl.BaseDepartmentDbDao

</dept-dao-class>

<emp-dao-class>

com.xdfsoft.organize.base.daoimpl.BaseEmployeeDbDao

</emp-dao-class>

</base-organize>

<app-organize>

<dept-impl-class>

com.xdfsoft.organize.base.BaseDepartment

</dept-impl-class>

**<dept-view-type>base</dept-view-type>**

<emp-view-type>right</emp-view-type>

<emp-impl-class></emp-impl-class>

<!-- 以下配置扩展部门机构或者人员的属性 -->

</app-organize>

……

……

</application>

3.**散列的人员**。在表SYS\_EMPLOYEE中存储了所有的人员（需注意的是：系统中的人员未必都有账号）。这是人员的根本来源，至于人员在哪个部门下，本表不进行描述，而是在表SYS\_DEPT\_EMP\_RELATION中存储。

4.**只能很好地支持base和right两种人员与组织机构的视图关系。本文认为人员与组织机构之间的关系也如同组织机构与组织机构之间的关系，是在某种视图下，人员才从属于某个部门。在表**SYS\_DEPT\_EMP\_RELATION中存储了人员与组织机构之间的关系（这种认识是否正确有待实践检验）。VIEW\_TYPE字段存储的就是人员与组织机构视图类型。视图类型是一个字符串。目前，系统只保证两种类型的组织机构视图可用：一个是**base**，表示在基准的人员与组织机构的关系视图；一个是**right**，表示权限中人员与组织机构的关系视图。表示人员与组织机构关系视图的字符串也出现在权限配置文件（WEB-INF/appconfig/com-xdf-right-config.xml）中的如下位置（红色加粗部分）：

<?xml version="1.0" encoding="GBK" ?>

<application packagename="com.xdfsoft.right" displayname="新东方权限系统">

<base-organize>

<store type="db" ref="com.xdfsoft" />

<dept-view-type>base</dept-view-type>

**<emp-view-type>right</emp-view-type>**

<cache-initializer>

com.xdfsoft.organize.base.daoimpl.OrganizeCacheDbInitializer

</cache-initializer>

<dept-dao-class>

com.xdfsoft.organize.base.daoimpl.BaseDepartmentDbDao

</dept-dao-class>

<emp-dao-class>

com.xdfsoft.organize.base.daoimpl.BaseEmployeeDbDao

</emp-dao-class>

</base-organize>

<app-organize>

<dept-impl-class>

com.xdfsoft.organize.base.BaseDepartment

</dept-impl-class>

<dept-view-type>base</dept-view-type>

**<emp-view-type>right</emp-view-type>**

<emp-impl-class></emp-impl-class>

<!-- 以下配置扩展部门机构或者人员的属性 -->

</app-organize>

……

……

</application>

有关组织机构和人员具体使用，详见组织机构API文档。

***注意：*API中部门类型**[***IDeptType***](file:///E:\workspaces\platform\OrganizeJava\doc\com\xdfsoft\organize\IDeptType.html)**或许有用，但是我们没有界面对其进行维护。而部门的区划ILocal可能在将来被去除，因为这个模型当前尚未成熟。**

## 2.2受控资源

在本文概述中阐述了受控资源的概念，即：主要是通过控制资源访问参数来实现对数据资源的管控，并将这些资源称为“受控资源”。

在易为平台理念中，凡是需要被人管理和控制访问的（数据）资源，都应是人所能识别的（在计算机系统中，人可以识别的数据，计算机一定可以识别，但是，不是所有计算机可以识别的数据都是人可以识别的）。

人可以识别的资源应具备以下几个基本属性：

1. ***具有唯一标识***
2. ***具有人可以识别的名字***
3. ***具有人可以识别的代码***

通过总结，我们发现，我们通常遇到的受控资源只有两种：“列表资源”和“树状资源”。

**列表资源**：是指每个资源之间没有上下级关系，因此，“列表资源”对象上的“操作”之间没有任何联系。例如：对不同“合同类型”下的合同进行“增、删、改、查”等“操作”是独立的。

**树状资源**：是指资源之间有上下级关系。同时，“树状资源”对象上的“操作”有特定的关系。也就是如果对子节点没有定义操作权限，那么“子节点”自动继承其直接上级父节点的操作权限，以此类推，直到找到根上操作权限。如果对根也没有操作权限，则意味着对该资源没有操作权限。例如：对不同“部门”下的人员档案的“增、删、改、查”的操作，如果对某部门节点定义了“查”的权限，那么即便其上级部门定义了“增、删、改、查”的操作，操作者在该部门下的权限也仍是“查”的权限，如果在该部门上没有定义权限，则其自动继承其上级部门所定义的“增、删、改、查”权限。

由此可见，系统中千差万别的资源可以归为两类，在授权时，权限系统需要做的就是能够将这些具体资源转换为抽象的两类资源，思路如下图所示：

上述思路明确后，我们就可以建立权限资源模型，主要接口、类和库表说明如下：

1. com.xdfsoft.resource.IResourceType用来描述每种类型的资源，包括：资源的ID，名称，描述，类型（树状或列表）以及该类资源的资源转换器（ResourceProvider），及其他参数。这个类是对资源描述和定义，是资源的“元数据”。其数据来源于库表：SYS\_RESOURCE，因此，对于开发者而言，凡是需要管控的资源均需注册到SYS\_RESOURCE表中（平台未提供可视化的界面，只能通过手工进行）。***注意：资源选择路径（SEL\_URL）是在JSF版下的产物，在权限管理UI转变为Flex后已经不再使用。***com.xdfsoft.resource.ResourceFactory类中提供了一些静态方法可以注册、反注册和获取资源类型。
2. com.xdfsoft.resource.IResourceObject是特定资源类型下的具体资源数据的基类接口，它定义了所有资源对像都具备的通用的属性，详见API文档。如果将“部门”看作资源类型（ResourceType），那么“生产部”、“销售部”、“办公室”就是资源对象（IResourceObject）实例。
3. com.xdfsoft.resource.IListResourceObject是com.xdfsoft.resource.IResourceObject子类，用来表示列表状资源对象都必须遵从的接口。
4. com.xdfsoft.resource.ITreeResourceObject是com.xdfsoft.resource.IResourceObject子类用来表示树状资源都必须遵从的接口。
5. com.xdfsoft.resource.IResourceProvider是两种类型的资源转换器的接口基类，该转换器，用来实现将业务系统中千差万别的资源转化为权限中抽象的资源。开发者需要在库表SYS\_RESOURCE的PROVIDER\_CLASS字段中注册具体的资源转换器实现类。
6. com.xdfsoft.resource. IListResourceProvider是所有列表资源转换器必须遵从的接口。
7. com.xdfsoft.resource. IListResourceProvider是所有树状资源转换器必须遵从的接口。
8. 业务系统中的数据资源一般都存在与数据库表中，为了让开发者不必开发更多的资源转换器，平台提供了缺省的资源转换器，可以从业务系统的数据库表之中直接将业务资源转换为权限资源。数据库表SYS\_DB\_RESOURCE描述了业务资源的来源表和来源字段。com.xdfsoft.resource.DefaultResProvider是缺省资源转换器的基类，com.xdfsoft.resource.DefaultListResProvider是缺省的列表资源转换器，com.xdfsoft.resource.DefaultTreeResProvider是缺省的树状资源转换器。

以上我们介绍了资源配置模型和总体思路，资源的注册和配置细节，详见第四章。

***注： com.xdfsoft.resource.ResourceFactory类提供了有关资源操作的各种公共方法。***

## 2.3权限核心概念

在易为平台中，权限系统主要由以下几个核心模型对象组成：

**账号**：账号是系统使用者在计算机系统中的表示。通常一个账号要有多个角色。账号所有角色权限的集合就是该账号的权限。在本版本中实现中，账号ID和人员ID是同一个，在以后的版本中将会改进。易为平台中，账号必须遵循com.xdfsoft.right.core.IAccount接口。为了支持批量创建账号，易为平台允许设置账号的初始密码，在权限配置文件中（比如：com-xdf-right-config.xml），以下配置片段允许管理员设定账号的初始密码。

<new-account-template>

<password>111111</password>

</new-account-template>

**角色**：角色是权限的集合，也就是说角色可以使用哪些功能，在这些功能上对何种数据资源进行何种操作。一个角色可以授给多个账号，那些账号被称为角色的“成员账号”。易为平台支持RBAC提出的“层级角色”概念，即：每个角色都可以有一个或者多个父角色，角色自动继承其所有父角色的所有权限。同理，一个角色也可以有多个子角色，角色的子角色被称之“成员角色”。

易为平台中，还把角色分为系统内建角色和用户自建角色两类。由于某些业务系统要求建立内建的角色，因此，系统内建角色不能被删除。用户自建角色的ID从10001开始，因此，可以内建立10000个系统内建角色。***易为平台约定：ID为1的角色是系统管理员角色。***这在系统初始化时由开发者手工初始化，数据表数据如下图所示：



易为平台中，角色必须遵循com.xdfsoft.right.core.IRole接口，角色必须从

com.xdfsoft.right.core.BaseRole类继承，系统内建角色必须从com.xdfsoft.right.core.BuildInRole继承，而用户自建角色则必须从com.xdfsoft.right.core.UserRole继承。各个类的详细信息参见Java API文档。

**功能：**功能是角色授权的主要资源，是提供给用户使用的功能点在计算机系统中的抽象表示。授权就是为角色授予功能及相关资源和操作。一般来说，在管理信息系统中，用户能够可以直接使用的功能都是有人机界面的，因此，功能可以包括一个或者多个以URL为代表的UI界面，本来，有些公共的UI界面也可以给多个的功能使用，但是公共UI界面如果需要控制权限，那么它只能属于一个功能，因此，在易为平台中，一个URL所代表的UI只能属于一个功能。如果一个功能有多个UI界面，但其中只能有一个UI界面是该功能的主界面。

除此之外，一个功能上还可能引用一个或者多个受控的数据资源，比如说：以“仓库”为授权对象进行物资管理，那么“物资管理”功能就需要引用“仓库”这种资源。另外，功能上还可能定义一些操作：比如增、删、改、查、打印、统计……等，这些操作通常会对应UI界面上的一些UI控件，也需要进行权限控制，更为复杂的是，有些功能需要根据某种资源下不同的资源数据控制不同的操作。比如，人员档案管理中，需要控制操作者在不同部门下对人事档案的操作不同，公司总部的人事管理员可以直接更还总部人员档案，但是可以查看分子公司的人员档案。我们将这种情况称为“‘操作’与‘资源’相关”。***易为平台限制：如果一个功能引用多种资源，比如“部门”和“仓库”，同时定义了“操作”，并且“操作”与资源相关，那么操作只能同其中一种“资源”相关。***

综上我们可以将功能分为以下几种情况：

1. 功能没有定义任何资源和操作。
2. 功能上只定义了操作，没有定义任何资源。
3. 功能上只定义了资源，没有定义任何操作。
4. 功能上既定义了资源又定义了操作，但是二者毫不相干。
5. 功能上既定义了资源又定义了操作，但是操作与资源相关。

易为平台中，所有功能都遵循com.xdfsoft.right.core.IFunction接口，功能有层级关系，如果一个功能有多个子功能，那么这个功能就是我们常说的“模块”，在易为平台中，用com.xdfsoft.right.core.IModule接口来规范模块，那些没有下级功能的功能就是**终端功能**，在易为平台中用，用com.xdfsoft.right.core.IEndPointFunction接口来规范**终端功能**。抽象基类com.xdfsoft.right.core.BaseFunction实现了IFunction接口，com.xdfsoft.right.core.Module类从BaseFunction类继承，实现了IModule接口。

**操作：**在上面我们已经讨论过操作，操作就是功能上可执行的动作，从人机界面角度看，如果一个人机界面是一个功能，那么操作就代表这个人机界面上的一个或者一组控件，这些控件可能是按钮，也可能是数据录入控件，比如数据表格。在表

# 权限系统配置

## 3.1资源注册与配置

### 3.1.1树形授权资源权限配置

本例权限控制需求说明：不同的客户经理（ROLE）管理不同地区（AREA）的客户，因此地区就是一个授权资源，又因地区具有上下级关系，所以地区是一个树形授权资源。比如地区有：中国、北京市、陕西省、山东省，不难看出北京市、陕西省、山东省上级为中国。本例的代码见附录。

完成树形授权资源配置有两种方式：一种是使用接口ITreeResourceProvider的默认实现com.xdfsoft.resource. DefaultTreeResProvider，其优点是不需要做任何开发，直接进行授权资源的注册即可实现。如果不考虑性能等因素情况下，建议直接使用DefaultTreeResProvider。另一种是自定义实现接口ITreeResourceProvider，其优点是可以满足一些特定的应用需求，比如需要提高数据访问的性能，需要对树形授权资源进行缓存等情况。

#### 3.1.1.1直接使用DefaultTreeResProvider

**1．向权限平台注册授权资源，即向表SYS\_RESOURCE增加授权资源信息。**

下面的SQL语句向表SYS\_RESOURCE插入地区授权资源记录，字段ID为资源代码，是SYS\_RESOURCE表的主键；字段NAME为资源名称；字段TYPE为资源类型，值“2”表示树形资源；字段PROVIDER\_CLASS为资源提供者类的类名，此处直接使用接口ITreeResourceProvider的默认实现“com.xdfsoft.resource. DefaultTreeResProvider”，以上这4个字段不能为空。

INSERT INTO SYS\_RESOURCE

(ID,NAME,TYPE,PROVIDER\_CLASS,SEL\_URL,PARAM,DESCRIBE,RELY\_RES)

VALUES

('20','地区','2','com.xdfsoft.resource.DefaultTreeResProvider',null,null

,null,null)

**2．注册数据来自于数据库的树形授权资源。**

下面的SQL语句向表SYS\_DB\_RESOURCE插入来自于数据库的地区授权资源，字段ID外键参照表SYS\_RESOURCE的字段代码的值；RES\_TABLE\_NAME：资源来源表名；RES\_KEY\_COLUMN：资源主键字段；RES\_NAME\_COLUMN：资源名称字段；RES\_CODE\_COLUNM：资源代码字段；UPPER\_RES\_COLUNM：树状资源上级资源字段，其他暂时未用字段可以为空，不做逐一介绍了。

INSERT INTO SYS\_DB\_RESOURCE

(ID,RES\_TABLE\_NAME,RES\_KEY\_COLUMN,RES\_NAME\_COLUMN,RES\_CODE\_COLUNM,UPPER\_RES\_COLUNM,RES\_CLASS\_NAME

,RES\_DS\_NAME,RELY\_RTABLE\_NAME,RELY\_RES\_CNAME,RES\_CNAME\_IN\_RTABLE)

VALUES (20,'AREA','ID','NAME','CODE','UPPER\_AREA\_ID',null,null,null,null,null)

3．**设置功能引用的授权资源，即向表RC\_FUNCTION\_RES\_REF增加功能与资源的映射信息**。

下面的SQL语句向表RC\_FUNCTION\_RES\_REF插入功能与资源的映射记录，字段FUNCTION\_ID为功能代码，字段RES\_TYPE\_ID为资源代码，本例资源代码的值为20。

INSERT INTO RC\_FUNCTION\_RES\_REF

(FUNCTION\_ID,RES\_TYPE\_ID,RES\_OPERATE\_RELATIVE,DESCRIPTION,RELY\_RES\_TYPE\_ID)

VALUES

(511004,20,0,null,null)

**4．完成以上3步，重新启动网站，即可进行授权操作。**

#### 3.1.1.2自定义实现接口ITreeResourceProvider

1. **实现树形资源提供者类。**

树形资源提供者类需从com.xdfsoft.resource. BaseResourceProvider继承，并实现com.xdfsoft.resource. ITreeResourceProvider接口。

例子中的资源为地区，类com.bepway.customer.AreaResourceProvider从BaseResourceProvider继承并实现了ITreeResourceProvider接口，然后需要将资源实现类部署到测试或者运行环境中即完成本步骤。例子的详细说明请参见附录中的类和方法的注释。本例仅仅只是一个范例，地区树形数据是由类com.bepway.customer.AreaService静态构造的，在实际的生产环境中，地区数据可能存储于数据库中，请参考具体的情况实现。

**小提示：**如何实现接口ITreeResourceProvider的详细信息亦可参见类“com.xdfsoft.organize.DeptResourceProvider”。

2．**向权限平台注册授权资源，即向表SYS\_RESOURCE增加授权资源信息。**

下面的SQL语句向表SYS\_RESOURCE插入地区授权资源记录，字段ID为资源内码，是SYS\_RESOURCE表的主键；字段NAME为资源名称；字段TYPE为资源类型，值“2”表示树形资源；字段PROVIDER\_CLASS为资源提供者类的类名，本例中的类名为“com.bepway.customer.AreaResourceProvider”，以上这4个字段不能为空。

INSERT INTO SYS\_RESOURCE

(ID,NAME,TYPE,PROVIDER\_CLASS,SEL\_URL,PARAM,DESCRIBE,RELY\_RES)

VALUES

('20','地区','2','com.bepway.customer.AreaResourceProvider',null,null

,null,null)

**3．后面步骤参见2.1节的步骤3和步骤4。**

### 3.1.2列表型授权资源权限配置

本例权限控制需求说明：不同的仓库管理员（ROLE）管理不同仓库（Storage）的物资，因此仓库就是一个授权资源，所有仓库是平级的，比如一号仓库和二号仓库之间并没有上下级关系，所以仓库是列表型授权资源。

本章以接口IListResourceProvider的默认实现DefaultListResProvider为例介绍列表型授权资源的权限配置。如果确实需要，可参见类DefaultListResProvider来实现IListResourceProvider接口以满足特定需求，本章不对此做任何介绍。

**1．向权限平台注册授权资源，即向表SYS\_RESOURCE增加授权资源信息。**

下面的SQL语句向表SYS\_RESOURCE插入仓库授权资源记录，字段ID为资源代码，是SYS\_RESOURCE表的主键；字段NAME为资源名称；字段TYPE为资源类型，值“1”表示列表型资源；字段PROVIDER\_CLASS为资源提供者类的类名，此处直接使用接口IListResourceProvider的默认实现“com.xdfsoft.resource. DefaultListResProvider”，这个4个字段不能为空。

INSERT INTO SYS\_RESOURCE

(ID,NAME,TYPE,PROVIDER\_CLASS,SEL\_URL,PARAM,DESCRIBE,RELY\_RES)

VALUES

('21','仓库','1','com.xdfsoft.resource.DefaultListResProvider',null,null

,null,null)

**2．注册数据来自于数据库的列表型授权资源。**

下面的SQL语句向表SYS\_DB\_RESOURCE插入来自于数据库的仓库授权资源记录，字段ID外键参照SYS\_RESOURCE的资源代码字段的值；RES\_TABLE\_NAME：资源来源表名；RES\_KEY\_COLUMN：资源主键字段；RES\_NAME\_COLUMN：资源名称字段；其他暂时未用字段可以为空，不做逐一介绍了。

**注意对于列表型资源，字段RES\_CODE\_COLUNM和UPPER\_RES\_COLUNM的值为null。**

INSERT INTO SYS\_DB\_RESOURCE

(ID,RES\_TABLE\_NAME,RES\_KEY\_COLUMN,RES\_NAME\_COLUMN,RES\_CODE\_COLUNM,UPPER\_RES\_COLUNM,RES\_CLASS\_NAME

,RES\_DS\_NAME,RELY\_RTABLE\_NAME,RELY\_RES\_CNAME,RES\_CNAME\_IN\_RTABLE)

VALUES (21,'STORAGE','ID','NAME',null,null,null,null,null,null,null)

1. **后面步骤参见2.1节的步骤3和步骤4。**

## 3.2授权资源依赖配置

本章以电力系统的客户档案管理为例，介绍如何进行依赖授权资源的配置。一个县级电力分公司包含多个供电所，每个供电所包含不同电压等级（常见的包括0.4KV、10KV、35KV，KV即千伏）的客户，每个供电所一般都有低压营业员、10KV高压营业员和35KV高压营业员，低压营业员管理本供电所0.4KV的客户档案，10KV高压营业员管理本供电所10KV的客户档案、35KV高压营业员可以管理本供电所的35KV的客户，县级分公司的高压营业员可以管理所有供电所的高压客户（电压大于10KV的客户）档案。通过分析我们轻易地提取出两种资源类型，一种是供电所，另一种是电压等级，并且电压等级依赖于供电所。下面开始介绍如何进行配置。前面已介绍过的注册方法，本章以后内容不再累述，只给出SQL代码示例。

**1．向权限平台注册授权资源，即向表SYS\_RESOURCE增加授权资源信息。**

营业区域授权资源注册SQL

INSERT INTO SYS\_RESOURCE

(ID,NAME,TYPE,PROVIDER\_CLASS,SEL\_URL,PARAM,DESCRIBE,RELY\_RES)

VALUES

('51001','营业区域','1','com.xdfsoft.resource.DefaultListResProvider',null,null

,null,null)

电压等级授权资源注册SQL,字段“获取资源的参数（PARAM）”用于设置访问授权资源的参数，对于来自于数据库的列表型授权资源而言，这个参数就是SQL中的where条件部分的内容。

INSERT INTO SYS\_RESOURCE

(ID,NAME,TYPE,PROVIDER\_CLASS,SEL\_URL,PARAM,DESCRIBE,RELY\_RES)

VALUES

('51004','电压等级','1','com.xdfsoft.resource.DefaultListResProvider',null,’TYPE\_ID=140002’,null,null)

**2．注册数据来自于数据库的列表型授权资源。**

INSERT INTO SYS\_DB\_RESOURCE

(ID,RES\_TABLE\_NAME,RES\_KEY\_COLUMN,RES\_NAME\_COLUMN,RES\_CODE\_COLUNM,UPPER\_RES\_COLUNM,RES\_CLASS\_NAME

,RES\_DS\_NAME,RELY\_RTABLE\_NAME,RELY\_RES\_CNAME,RES\_CNAME\_IN\_RTABLE)

VALUES (51001,'A\_BUSINESS\_AREA','AREA\_ID','AREA\_NAME',null,null,null,null,null,null,null)

INSERT INTO SYS\_DB\_RESOURCE

(ID,RES\_TABLE\_NAME,RES\_KEY\_COLUMN,RES\_NAME\_COLUMN,RES\_CODE\_COLUNM,UPPER\_RES\_COLUNM,RES\_CLASS\_NAME

,RES\_DS\_NAME,RELY\_RTABLE\_NAME,RELY\_RES\_CNAME,RES\_CNAME\_IN\_RTABLE)

VALUES (51004,'SYS\_PUBCODE','CODE\_ID','CODE\_NAME',null,null,null,null,null,null,null)

**3．设置功能引用的授权资源，即向表RC\_FUNCTION\_RES\_REF增加功能与资源的映射信息**。

字段依赖资源（RELY\_RES\_TYPE\_ID）：定义给定资源所依赖的上级资源。这样，功能上引用的资源就形成了一颗树，必须从树根开始授权才能完成授权。本例中的RELY\_RES\_TYPE\_ID值为电压等级的资源代码51001，执行以下的语句即可完成供电所和电压等级的关联。

INSERT INTO RC\_FUNCTION\_RES\_REF

(FUNCTION\_ID,RES\_TYPE\_ID,RES\_OPERATE\_RELATIVE,DESCRIPTION,RELY\_RES\_TYPE\_ID)

VALUES

(511004,51004,0,null,51001)

# 第4章 Flex端权限开发