mrsingsing

基於RBAC 權限模型的架構設計

2020-05-15

談到權限控制，實際上在我們的生活中無處不在，例如我們無時無刻都在使用的智能手機，打開手機需要生物指紋或者人臉識別；當你進入公司的時候，你需要工卡在門禁滴一下；又如曾經的你是否有過半夜打開心儀女孩的QQ 空間，欲一探究竟的時候，卻告訴需要添加好友才能訪問。這些都是我們日常生活中體現權限的地方，而在軟件開發中，權限管理也是無處不在的產品設計功能，尤其是後台管理後台無法對權限管理避而不談，今天嘗試深入了解權限管理設計中最常用的RBAC 模型。

權限模型

權限 二字從字面上可拆分為 权力+限制 ，從使用者的角度來說，也就是在限制的範圍內正確地行駛權力，而站在設計者的角度來解讀，則是通過合理的手段控制使用者能訪問到他們能夠訪問到的資源。

在權限設計領域，最常見的權限模型有以下四種：

- ACL（Access Control List）：訪問控制列表 用户 -> 权限
- RBAC（Role-Based Access Control）：基於 **角色** 的權限控制 用户 -> 角色 -> 权限
- ABAC（Attribute-Based Access Control）：基於 **屬性** 的權限控制，該模型根據特殊的規則分配權限（用户、资源、对象属性）-> 权限
- PBAC（Policy-Based Access Control）：基於 **策略** 的權限控制，用戶（組）隸屬於角色，角色隸屬於資源（項目） 用户 -> 角色 -> （资源/项目 + 权限）

第一種ACL 模型，權限能直接賦予用戶，例如將查看訂單列表（權限）賦予某位運營人員（用戶）。但是這種模式的缺點在於，但用戶量達到一定量級的時候，那麼就需要對每個用戶都進行一次授權操作，那麼這個工作量就會相當大。

因此就出現了RBAC 模型，這是軟件設計中最常用的權限管理模型，相比於ACL 模型，RBAC 模型在用戶與權限之間多了一個元素「角色」，通過權限關聯角色、角色關聯用戶的方法來間接地賦予用戶權限，從而實現用戶與權限的解耦。

後兩種大概知道就行，因為一般不會用到，詳情可自行Google 了解。

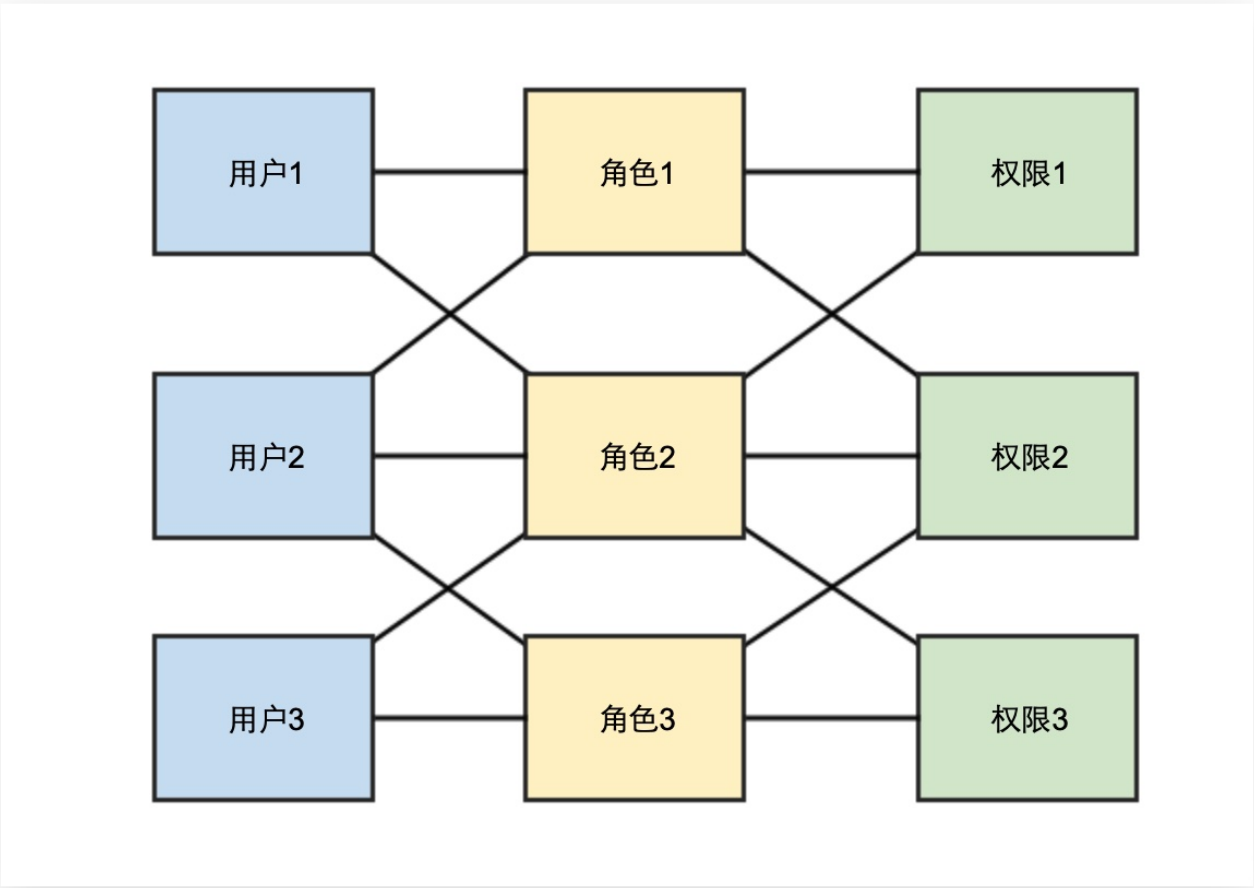
RBAC 模型的演進

在RBAC 模型的演進發展出四個版本，分別是RBAC 0~3。

RBAC0 模型

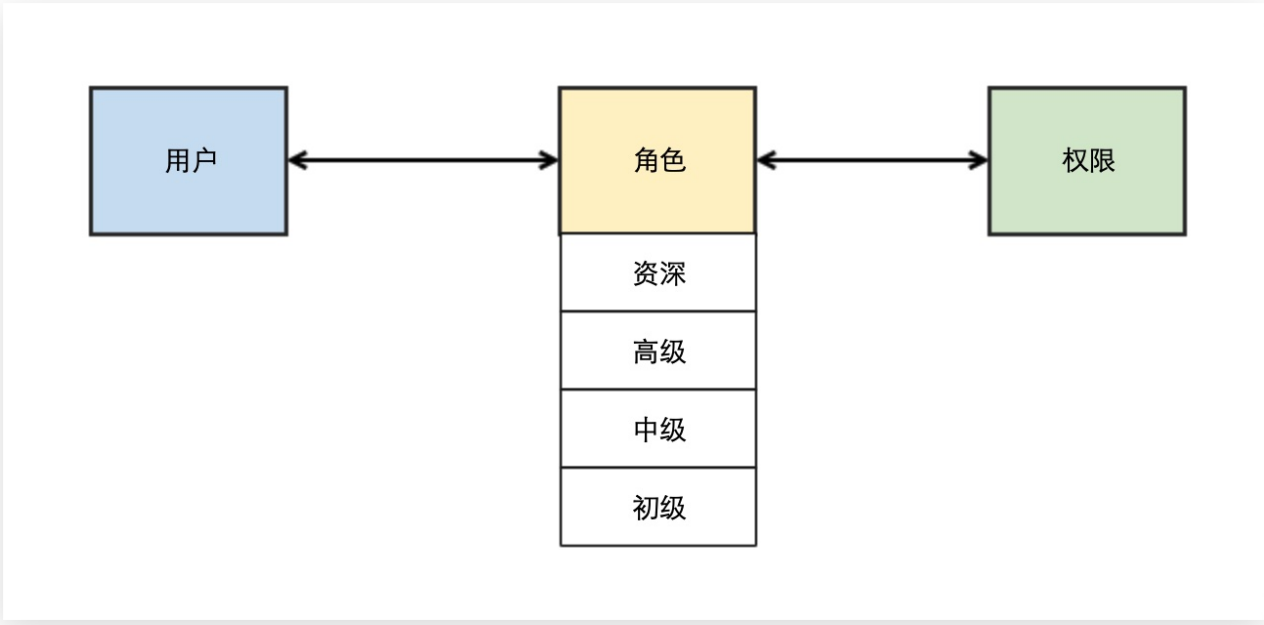
RBAC9 是最基礎的權限模型，很多產品基於該模型就已經能滿足設計需求。用戶和角色之間可以是一對多也可以是多對多關係，而角色和權限之間則多以多對多的關係為主，用戶擁有的權限等於他所有的角色持有的權限之和。

應用場景：例如在電商公司中抽像出幾種管理業務的角色，諸如銷售經理、項目經理、市場經理等等，小陳既擔當營銷部門的市場經理又擔當銷售部的銷售經理，那麼就需要將這兩個角色的賦予給他。



RBAC1 模型

RBAC1 模型其實是RBAC0 的升級版，它對角色這層元素上進行了細分，引入繼承的概念，也就是可以繼承某個基礎角色生成子角色。



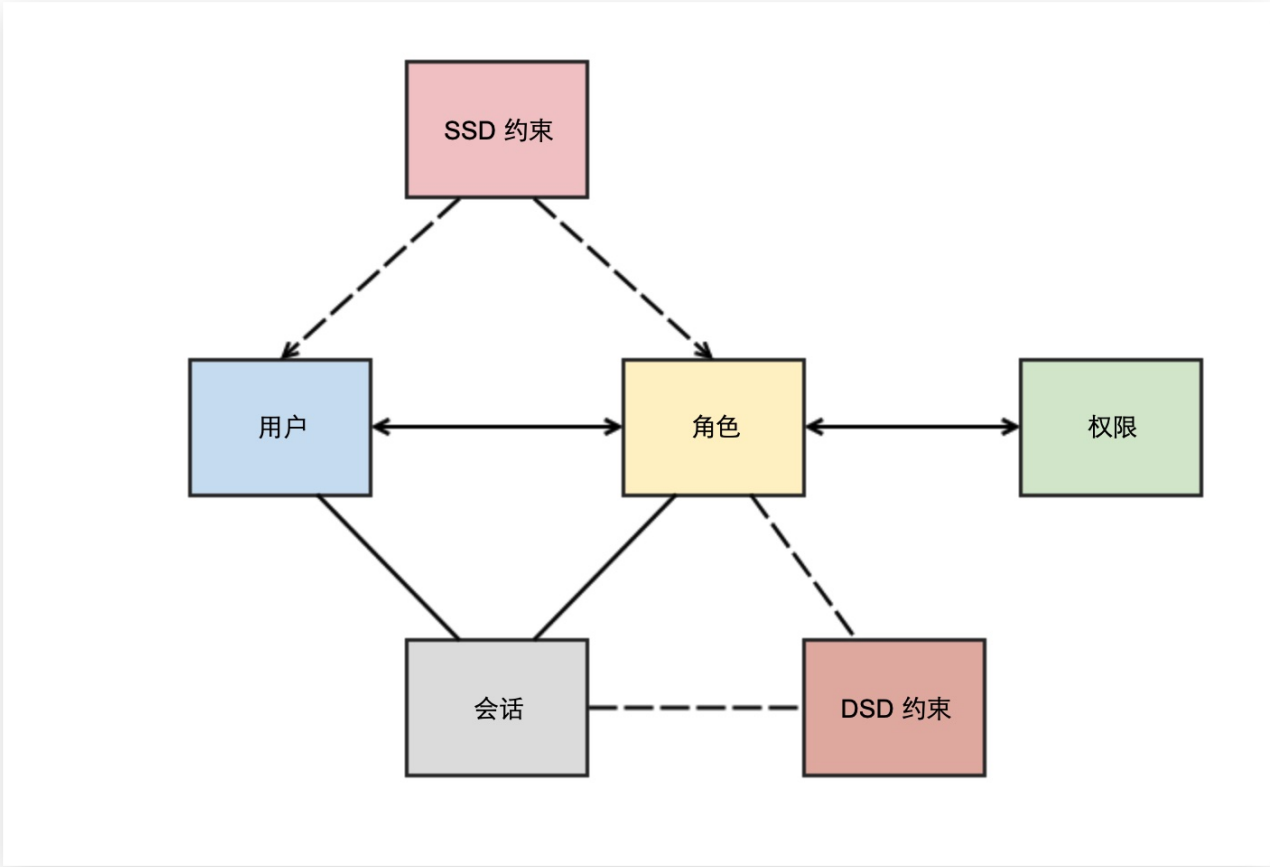
應用場景：承接上面的例子，市場經理崗位可能會分為總監級別、經理級別、副經理級別，這時候如果小陳只是一個副經理級別的，那麼他所擁有的市場經理的權限肯定就沒有總監級別的多。

RBAC1 模型則更好地在角色層面進行細分，更好地映射了企業組織架構中的職能的權限，根據實際的管理權限對相似職能的角色進行刪減以達到級別分明的效果。

RBAC2 模型

在RBAC2 中對角色層面增加了更多的限制：

- 靜態職責分離SSD
  - 角色互斥：相同用戶不能同時擁有互斥關係的角色，例如會計和出納兩個角色就是互斥的
  - 基數約束：角色被分配到的用戶有數量上限，例如公司中只有一個CEO 職位，那麼這個角色的數量就是有限的
  - 先決條件角色：要擁有更高級別的角色權限，需要先獲取到相對來說低級別的一些權限，例如副經理要想獲取到總監級別權限，那麼他需要先獲取到經理級別的權限
- 動態職責分離DSD
  - 動態的限制用戶及其擁有的角色，例如一個用戶可以同時擁有兩個角色，但是運行時只能激活一個角色



RBAC3 模型

RBAC3 模型就是 `RBAC1 + RBAC2` 兩個模型的合集，所以RBAC3 既有RBAC1 的角色等級劃分，也有RBAC2 的角色限制。這種模型只有在系統比較複雜的時候才派得上用場，不然設計得過度複雜，對開發和後期維護也不是好事情。

RBAC 模型的優缺點：

優點

1. 簡化了用戶和權限的關係，權限只要一次賦予角色就能重複使用
2. 對後續的維護更加友好，用戶和權限之間存在更多的斡旋空間，便於基於角色進行權限擴展

缺點

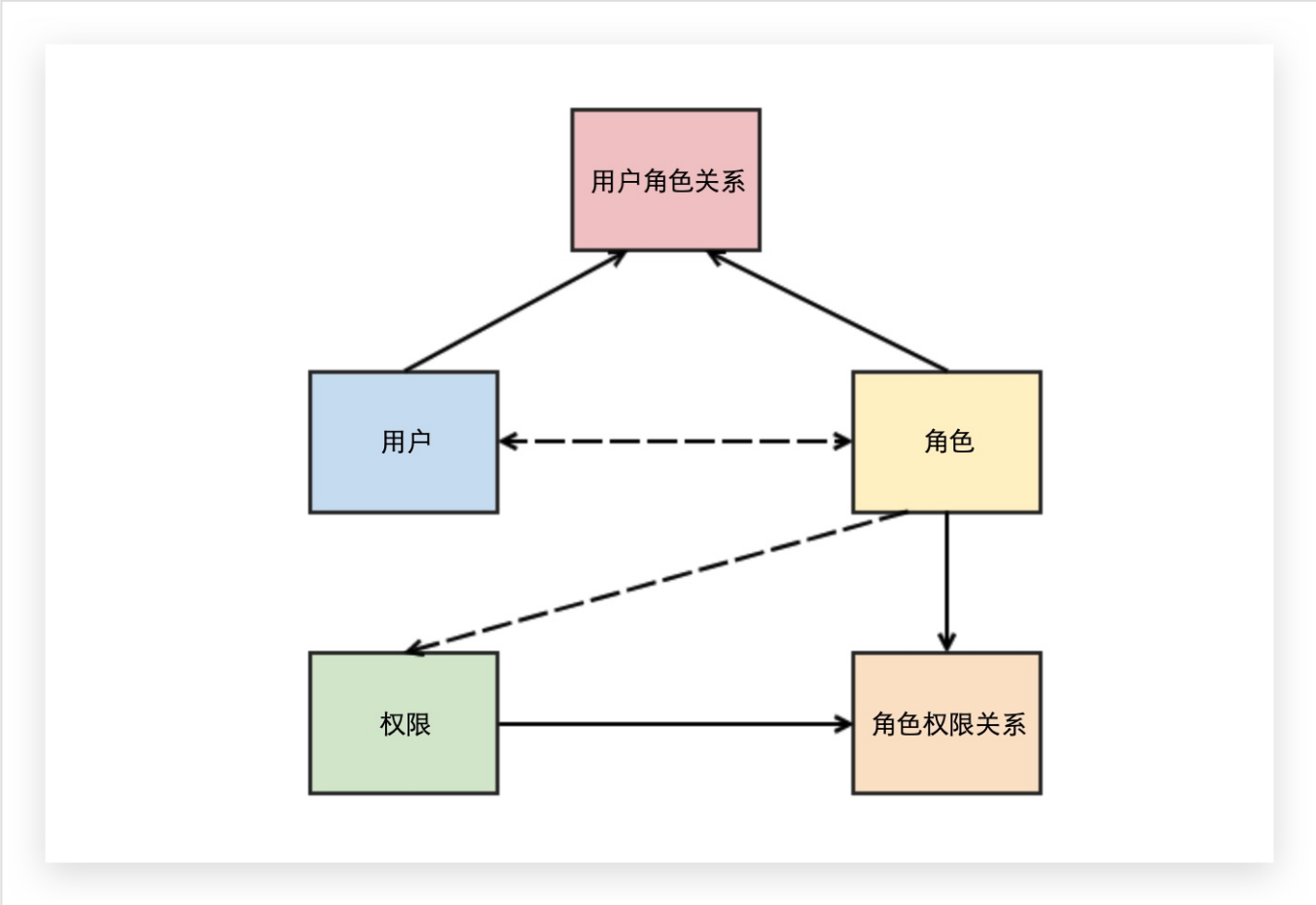
1. RBAC 模型並未提供控制操作順序的機制，會對有嚴格操作順序的系統造成困難

RBAC 架構設計

前面我們提到RBAC 涉及三個層面，就是用戶、角色和權限。

用戶我們可以理解為獨立的個體，例如張三、李四、王五。而角色的定義是很廣泛的，如果我們從職位劃分有銷售經理、市場經理、項目經理等等，按部門劃分的話可以是銷售部、市場部、項目部，所以說角色應該按照系統的業務需求來劃分。而事實上，在我們的架構設計中需要對用戶和角色的關係進行描述，比如其內容應該是張三是銷售經理、李四是市場經理、王五是項目經理。

還有就是權限層面，權限層面可以劃分為查看客戶列表、添加客戶、刪除客戶等等。權限和角色是綁定起來的，所以角色和權限的描述可能是這樣的：銷售經理可以查看客戶、添加客戶、刪除客戶；銷售僅可以查看客戶。



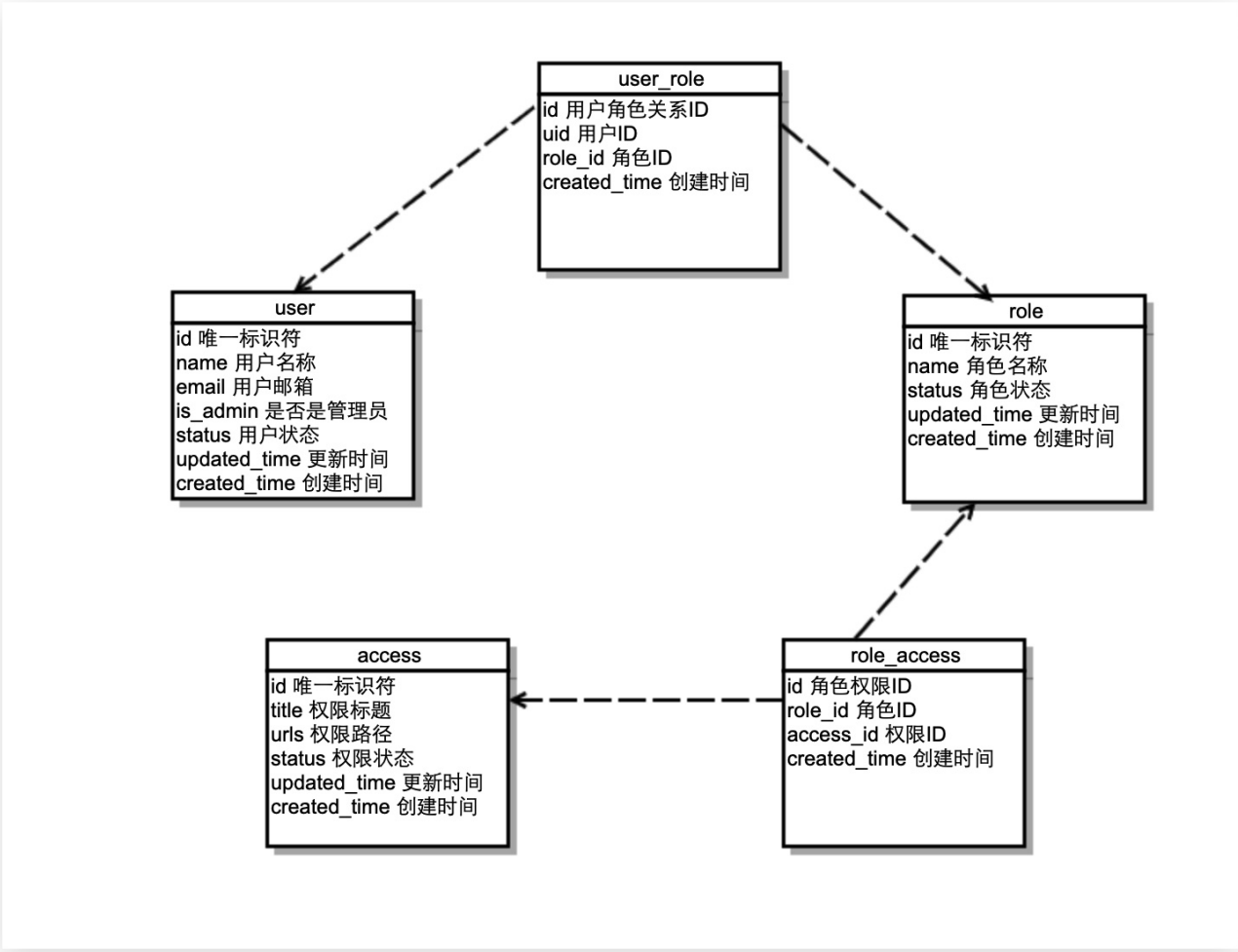
在實際的業務開發中，可以對RBAC 模型做相關的擴展，例如在員工人數眾多的情況下，反复地對某個員工進行權限授予是個繁瑣的行為，我們可以增加 **用戶組** 的概念，直接給用戶組分配角色，再把用戶加入用戶組，這樣用戶除了自身的權限外，還共享了用戶組的所有權限。

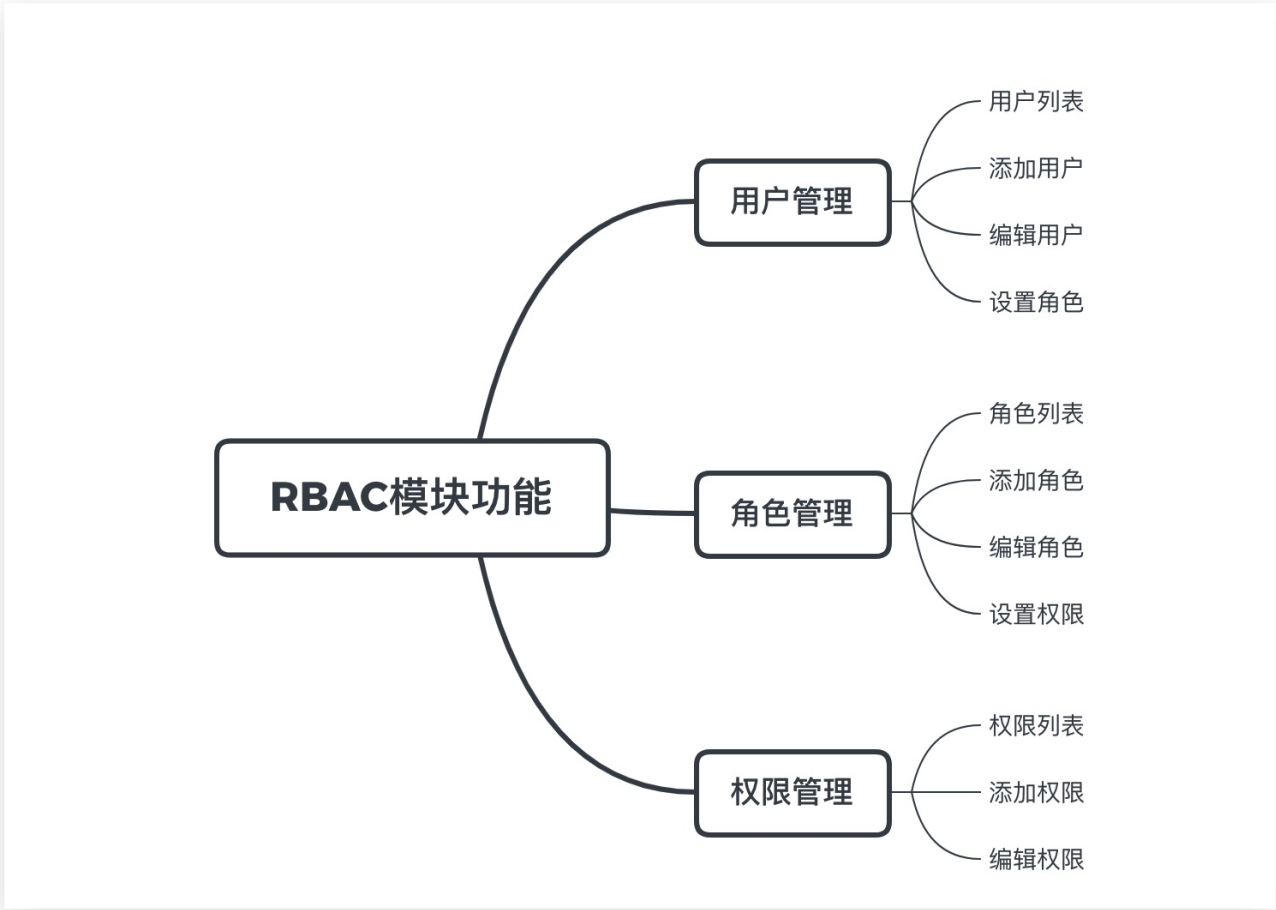
數據庫與功能模塊設計

在架構設計的章節中我們提到了除用戶、角色和權限外，他們相互聯繫的關係共五個層面，映射到數據庫的設計對應的就是五張表。

- user（用戶）：每個用戶都有唯一的UID 識別，並被授予不同的角色
- role（角色）：不同角色具有不同權限
- permission（權限）：訪問權限

- 用戶-角色映射：用戶和角色之間的映射關係
- 角色-權限映射：角色和權限之間的映射



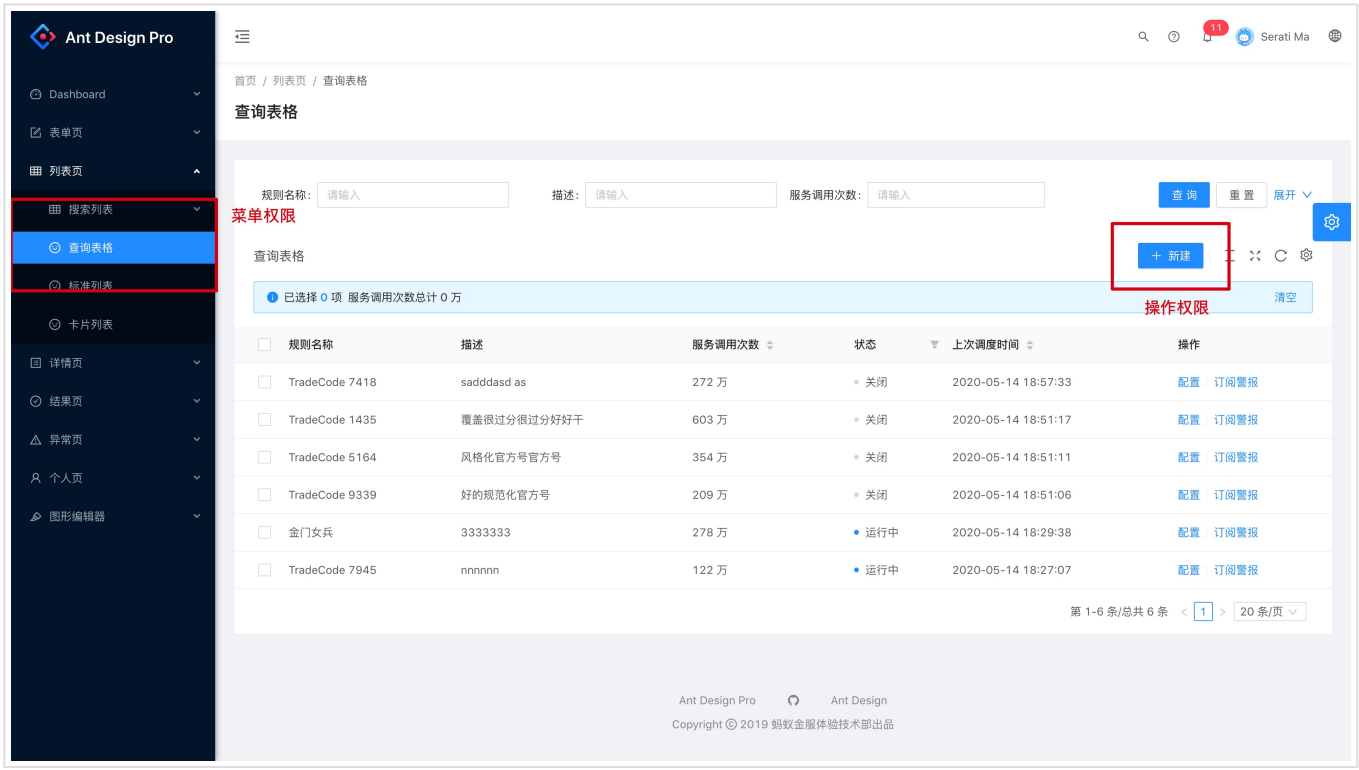


前端權限控制思路

前端權限從本質上來說即根據權限表對視圖層的展示以及對請求接口的訪問進行限制。

下面我們談談前端界面對於權限控制的粒度大小，以傳統的後台系統為例，有以頁面為維度作為權限控製粒度的，擁有權限的人員方可訪問對應的頁面；有以操作作為權限控制的，對應的前端界面中的頁面元素需要隱藏或提示；也存在用戶通過瀏覽器的開發者工具修改DOM 元素以繞過權限直接操作的，這時候就需要對接口請求做攔截。

- 頁面權限（菜單權限）：進入頁面的權限
- 操作權限：增刪改查的權限
- 數據權限：發送接口到服務端的權限



在傳統的後台系統中會在登錄請求中，得到用戶所擁有的權限數據，這部分是需要後端的技術支持的。前端根據權限數據展示對應的菜單，點擊菜單，才能查看相關的界面。

界面的控制：如果用戶沒有登錄，手動在地址欄上跳轉到管理界面地址，則需要跳轉到登錄界面。如果用戶已經登錄，但是手動輸入非權限範圍內的界面地址，則需要重定向到預設的404 界面。

如果用戶通過非常規操作，比如通過瀏覽器將某些禁用按鈕變成啟用狀態，此時發送的請求，也應該被前端攔截。

在React 中，可以通過把權限數據在根節點通過Context API 向下傳遞，通過高階組件對權限控制的菜單、元素進行外層包裹，其次對於數據請求攔截可以在發送HTTP 請求前做校驗。

可以參考Ant Design Pro的權限設計。

參考資料：

- RBAC 模型：基於用戶-角色-權限控制的一些思考
- 基於RBAC 的前端權限控制
- 網易高手：角色權限設計的100 種解法
- 角色權限設計：以阿里產品為例，展開體驗分析
- RBAC 權限管理模型：基本模型及角色模型解析及舉例
- 如何設計網站權限系統?
- 擴展RBAC 用戶角色權限設計方案

‹ Web 前後端應用Docker 容器化獨立部署實踐

常用移動前端開發調試方式 ›