设计可伸缩的Angular应用程序

<https://indepth.dev/designing-scalable-angular-applications>

NgRx Facades: 优点和缺点

<https://auth0.com/blog/ngrx-facades-pros-and-cons/>

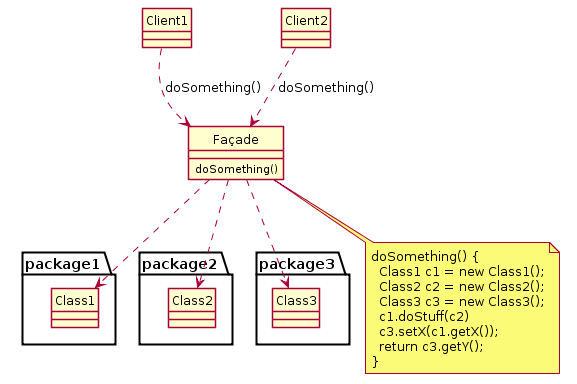
你可能已经听说了最近Angular社区的一个热门话题:NgRx中的facades（facades in NgRx）。我认为写一篇文章来讨论这个问题是个好主意。让我们了解什么是facades，支持和反对它的理由，以及如何实现它。(您可以在这个存储库中找到示例代码。https://github.com/auth0-blog/ngrx-facades)

**What are facades?**

在代码中，术语“facade”指的是facade模式，它是来自著名的《设计模式》(根据作者通常称为“四人帮”)一书中的一种结构设计模式。就像建筑的正面向路人隐藏内部一样，代码中的facade通过只公开某些方法来隐藏底层服务的复杂性。您经常会听到这种对复杂性的隐藏被称为“添加一个抽象层”。抽象是一种更简单、更通用的模型，它只提供消费者需要知道的东西。

facade模式和抽象通常类似于您和您的汽车之间的关系。你只需要知道当你转动钥匙，按下油门踏板，你的车就会向前移动。你不需要关心引擎的基本机械原理或者燃烧科学为了去杂货店。

下面是facade模式在图中的表现方式:

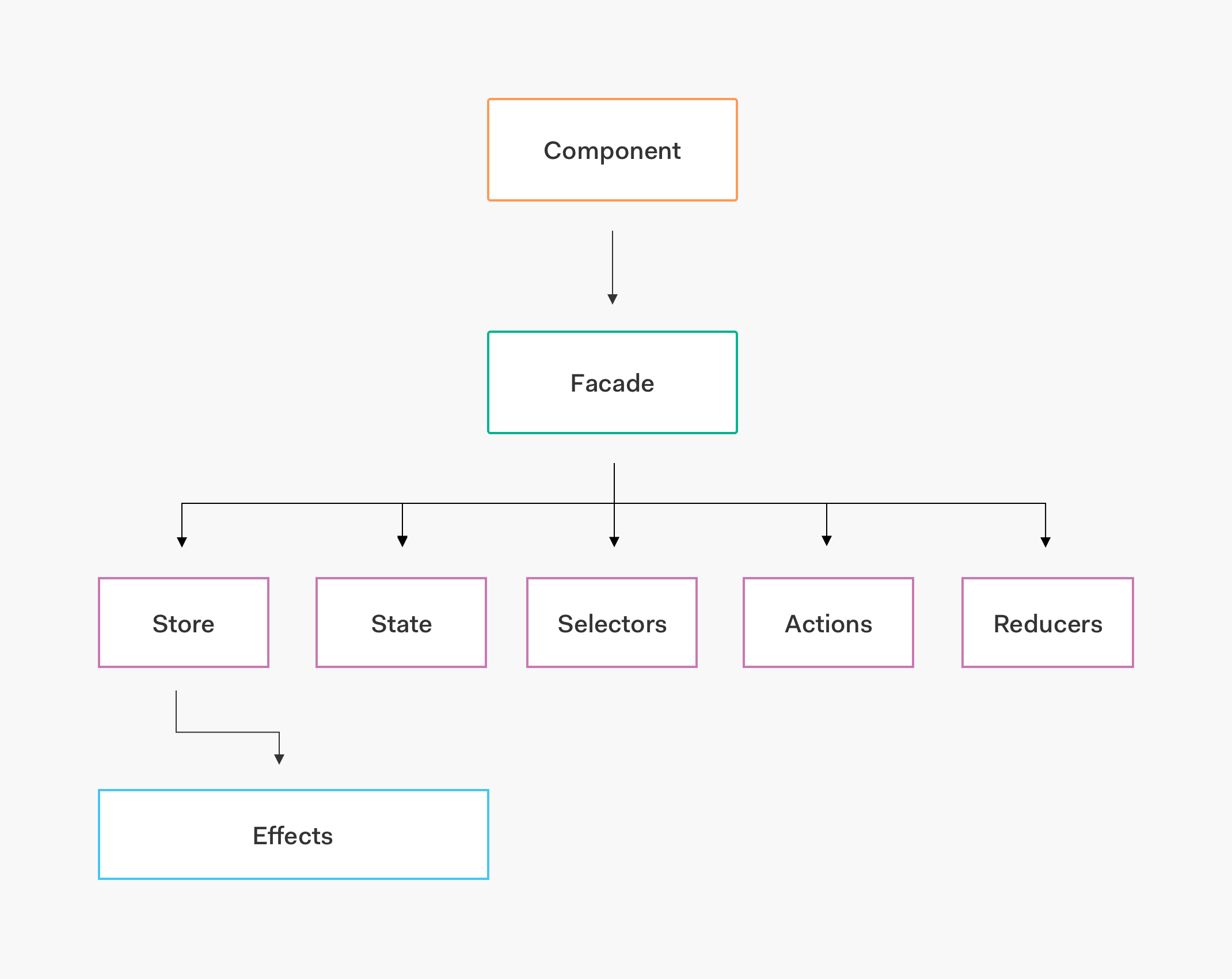


您可以在这里看到客户机如何不知道其他服务的任何信息。它们只是调用doSomething()， facade在幕后处理服务。

**Facades in NgRx**

最近，关于是否使用外观和NgRx来隐藏像存储、操作和选择器这样的部分有很多讨论。这是由Thomas Burleson的一篇名为《NgRx + Facades:更好的状态管理》（[NgRx + Facades: Better State Management](https://medium.com/@thomasburleson_11450/ngrx-facades-better-state-management-82a04b9a1e39)）的文章引发的。facade基本上就是一个处理与存储交互的Angular服务。当组件需要分派操作或获取选择器的结果时，它将调用facade服务上的适当方法。

此图说明了组件、外观和NgRx的其余部分之间的关系:



让我们看一个示例来更好地理解这一点。下面是我的NgRx认证教程的示例代码中的BooksPageComponent(我已经将代码隐藏在组件装饰器中以使其更容易阅读):