Facade design pattern

Services should be feature oriented and we have choice to build a giant service class or micro services collections. In first approach, there shall be only one service which shall contain all business logic and shall be provided via Angular Dependency Injection in all components within system. Issue with this approach is, giant service class shall get bloated eventually leading performance issue. Every component shall get injected with service and functionality which is not required to consumer component at all.

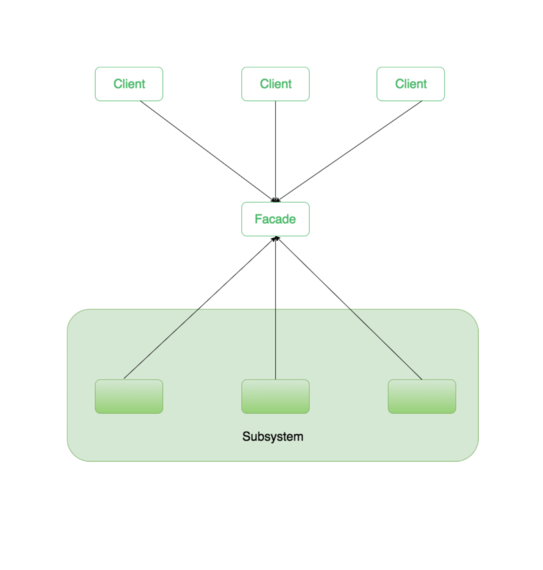
In second approach (followed widely), feature specific micro service gets built. For example, in our system we have Login, Dashboard components then we shall build LoginService, DashboardService and so on. Each service shall contain functionality required for specific targeted component.

But there is a problem. No matter how good naming convention you have followed for building components and services, there shall be time required to figure out specific name of service for specific class. Also you may end up writing duplicate service class with slightly different name for same component than other team built.

We can solve this problem using Facade design pattern.

Facade discusses encapsulating a complex subsystem within a single interface object. This reduces the learning curve necessary to successfully leverage the subsystem. It also promotes decoupling the subsystem from its potentially many clients.

The Facade object should be a fairly simple advocate or facilitator. It should not become an all-knowing oracle or “god” object.



Follow the below steps to build Angular services using Facade pattern:

1. Define all your Angular services as per your business requirement and/or keep adding more as needed
2. Create a service called “FacadeService” (feel free to use any other name here)
3. Create a shared NgModule and provide all Angular services
4. Aggregate all Angular services inside FacadeService and resolve their instances from Angular DI inside “property” access
5. To access Angular DI engine, we need to inject Inejctor Angular’s built-in service inside FacadeService constructor

How to add new service to the app?

1. Define the service as per business requirement
2. Configure the new service as provider in services module
3. Provide singleton implementation for the service in façade service
4. Also provide implementation for methods to access methods in new service
5. Use only façade service to access service methods in any component