**Week 4 Assignment**

Sai Vamsi Kampasati

**Geoserver Code**

Geoserver codebase has few well-written unit tests covering major code paths and components. Few of the unit tests are integration tests to cover all the major components and end to end scenarios. Geoserver codebase is designed following these patterns. The goal of this document is to present these design patterns to people who are working in JAX-RS and to promote using the design patterns in our design of future products. The design patterns that we are currently using and discussing the benefits of each of the design patterns. This document is also intended to serve as a reference to others. It is written to provide information and examples on common design patterns that are used and are considered common design patterns.

The facade pattern is about building a better face on the existing class. The facade class is an adapter to another class, but it wraps all calls to the real class into simpler functions that the client will call. If there is no facade, the client would need to know the internal workings of the class to make any changes. The facade hides these details and makes changes easier. Adapter pattern is used to adapt an existing interface so that it can be used by the same class for different kind of methods. The class being adapt to is called adapter. An Adapter can be used to adapt any class to another. An Adapter should be implemented only by classes that needs to be adapted. It is used to extend the behavior of an existing object or class. Object Composition is a well-known OO design pattern that describes the way a set of classes interact. The Object Composition creates a new instance of each of the objects, adds them to a container and returns the container. In other words, the objects are composed in the composition. An important characteristic of this pattern is that all the classes involved in the composition are visible to the container. This means that all the members of the composed objects will be visible in the container. One feature of the Object Composition pattern is that an instance of an object might hold a reference to a container that holds other objects. The client of an object doesn't need to know about any of its members. An instance of an object is responsible for the lifetime of its own instances. A client doesn't make use of an instance of the object to perform work. Instead, a client uses the methods of the object. Polymorphism is one of the design patterns that explains the fact that different kinds of objects can be used interchangeably for a certain purpose. The most well-known part of polymorphism is subtyping. A subtype may inherit the properties and operations of a supertype, but the subtype is not required to be an exact copy of its supertype. Subtypes can have some limitations and special properties that are not available to its supertype. A software design principle of object-oriented software is inheritance. The inheritance relationship is such that each class extends from a class called the superclass. This means that the superclass defines how the subclass should behave.

<https://github.com/geoserver/geoserver/blob/2.12.1/src/main/src/main/java/org/geoserver/config/impl/DefaultGeoServerFacade.java>

Facade design pattern is the process of separating an abstraction or an interface from its implementation and exposing it as a public interface, where the client application may use the abstraction, but is not required to know its implementation. The facade is a software component that creates a new layer of abstraction between an application or system and its clients. The application or system is not required to be aware of the new layer of abstraction between it and the client. The facade also helps to ensure that the software in the middle layer does not change in the future.

<https://github.com/geoserver/geoserver/blob/2.12.1/src/community/jdbcconfig/src/main/java/org/geoserver/jdbcconfig/internal/ResultSetIteratorAdapter.java>

The code is well commented with detailed examples which are helpful to understand both the patterns. A well-written code always enhances our understanding of programming, hence when it is used in a blog it’s the best example of that. The Adapter design pattern is used to encapsulate one interface or an abstract class with a specific implementation with the goal of hiding the implementation. This is to allow multiple interfaces to communicate with each other while using the same implementation. Since we are using generics, it is enough to write one version of the method, and it will be called from every class that is using that interface. The Iterator design pattern in Adapter pattern is a simple one-line coding.

Text

Description automatically generated

Complexity is a good thing when it comes to writing maintainable code.