Introduction to Java

By Sanjana Bandara

Content

- Design Patterns
- MVC architecture

Desian Patters

Design Patterns

- A design patterns are well-proved solution for solving the specific problem/task.
- A design pattern represents an idea, not a particular implementation.
- By using the design patterns you can make your code more flexible, reusable and maintainable. It is the most important part because java internally follows design patterns.

- Advantages

They are reusable in multiple projects.
They provide the solutions that help to define the system architecture.
They capture the software engineering experiences.
They provide transparency to the design of an application.
They are well-proved and testified solutions since they have been built upon the knowledge and experience of expert software developers.
Design patterns don’t guarantee an absolute solution to a problem. They provide clarity to the system architecture and the possibility of building a better system.

Core Java Design Patterns

1.Creational Design Pattern
Factory Pattern
Abstract Factory Pattern
Singleton Pattern
Prototype Pattern
Builder Pattern

2. Structural Design Pattern
Adapter Pattern
Bridge Pattern
Composite Pattern
Decorator Pattern
Facade Pattern
Flyweight Pattern
Proxy Pattern

3. Behavioral Design Pattern
Chain Of Responsibility Pattern
Command Pattern
Interpreter Pattern
Iterator Pattern
Mediator Pattern
Memento Pattern
Observer Pattern
State Pattern
Strategy Pattern
Template Pattern
Visitor Pattern

Creational Design Patterns

Creational design patterns are concerned with **the way of creating objects.** These design patterns are used when a decision must be made at the time of instantiation of a class (i.e. creating an object of a class).

An object is created by using new keyword in java. For example:
StudentRecord s1=**new** StudentRecord();

The nature of the object must be changed according to the nature of the program. In such cases, we must get the help of creational design patterns to **provide more general and flexible approach**.

Structural design patterns

Structural design patterns are concerned with how classes and objects can be composed, to form larger structures.

The structural design patterns **simplifies the structure by identifying the relationships**.

These patterns focus on, how the classes inherit from each other and how they are composed from other classes.

Behavioral Design Patterns

Behavioral design patterns are concerned with **the interaction and responsibility of objects.**

In these design patterns, **the interaction between the objects should be in such a way that they can easily talk to each other and still should be loosely coupled.**

That means the implementation and the client should be loosely coupled in order to avoid hard coding and dependencies.

MVC Architecture in Java

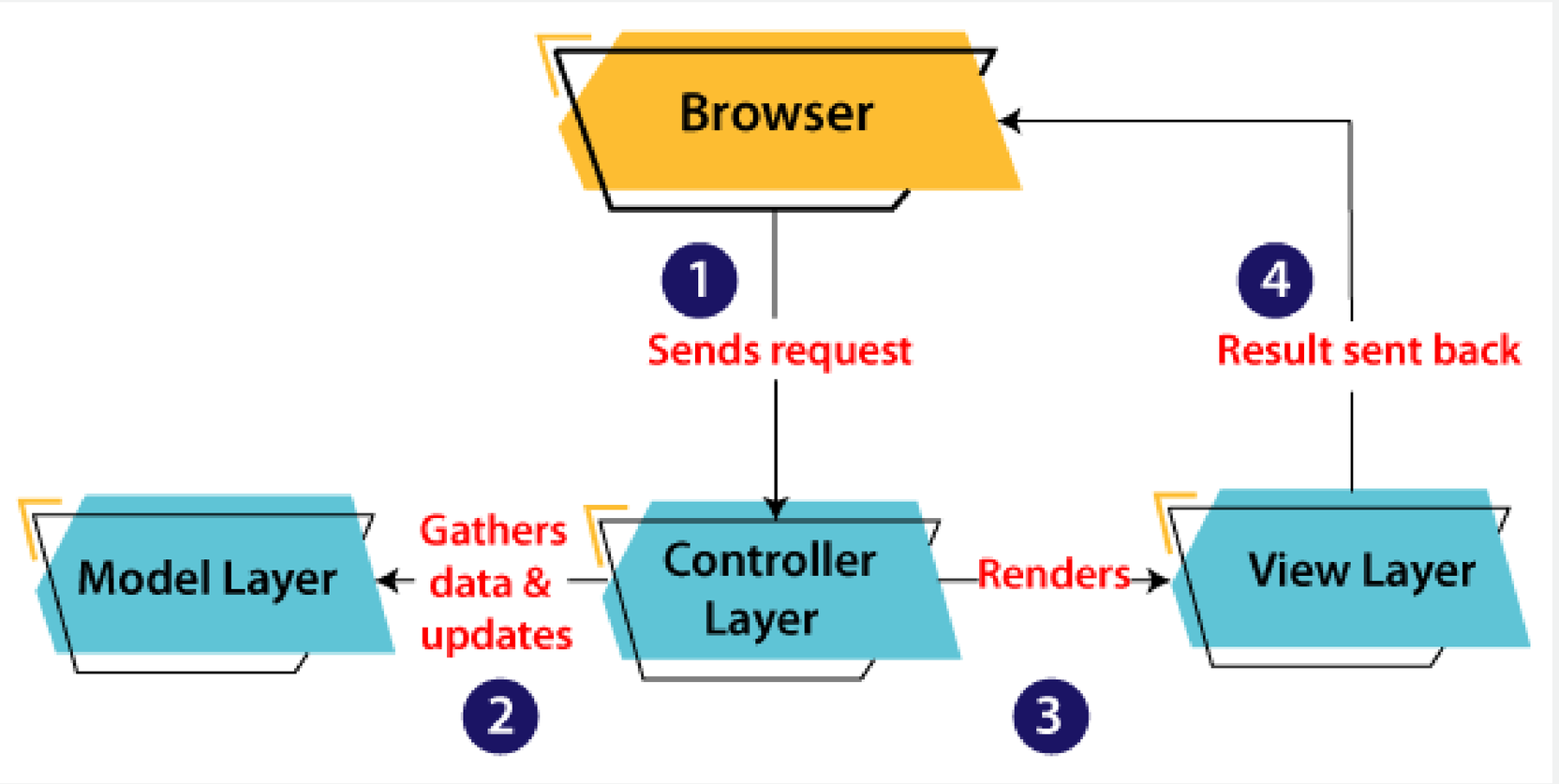
MVC Architecture in Java

The Model-View-Controller (MVC) is a well-known design pattern

in the web development field. It is way to organize our code. It specifies that a program or application shall consist of data model, presentation information and control information.

The model designs based on the MVC architecture follow MVC design pattern. The application logic is separated from the user interface while designing the software using model designs.

- **Model**: It represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed.
- **View**: It represents the presentation layer of application. It is used to visualize the data that the model contains.
- **Controller**: It works on both the model and view. It is used to manage the flow of application, i.e. data flow in the model object and to update the view whenever data is changed.



- A client (browser) sends a request to the controller on the server side, for a page.
- The controller then calls the model. It gathers the requested data.
- Then the controller transfers the data retrieved to the view layer.
- Now the result is sent back to the browser (client) by the view.

Advantages of MVC Architecture

- MVC has the feature of scalability that in turn helps the growth of application.
- The components are easy to maintain because there is less dependency.
- A model can be reused by multiple views that provides reusability of code.
- The developers can work with the three layers (Model, View, and Controller) simultaneously.
- Using MVC, the application becomes more understandable.
- Using MVC, each layer is maintained separately therefore we do not require to deal with massive code.
- The extending and testing of application is easier.

Implementation of MVC using Java - Practical

To implement MVC pattern in Java, we are required to create the following three classes.

Employee Class, will act as model layer
EmployeeView Class, will act as a view layer
EmployeeContoller Class, will act a controller layer

Model Layer

- The Model in the MVC design pattern acts as a data layer for the application. It represents the business logic for application and also the state of application.
- The model object fetch and store the model state in the database.
- Using the model layer, rules are applied to the data that represents the concepts of application.

Controller Layer

- The controller layer gets the user requests from the view layer and processes them, with the necessary validations.
- It acts as an interface between Model and View.
- The requests are then sent to model for data processing. Once they are processed, the data is sent back to the controller and then displayed on the view.

Main Class Java file

- The **MVCMain** class fetches the employee data from the method where we have entered the values. Then it pushes those values in the model.
- After that, it initializes the view (EmployeeView.java). When view is initialized, the Controller (EmployeeController.java) is invoked and bind it to Employee class and EmployeeView class.
- At last the updateView() method (method of controller) update the employee details to be printed to the console.