Tech Mahindra

Model View Controller (MVC)

Objectives

At the end of this session, you will be able to,

- Describe MVC Architecture

Agenda

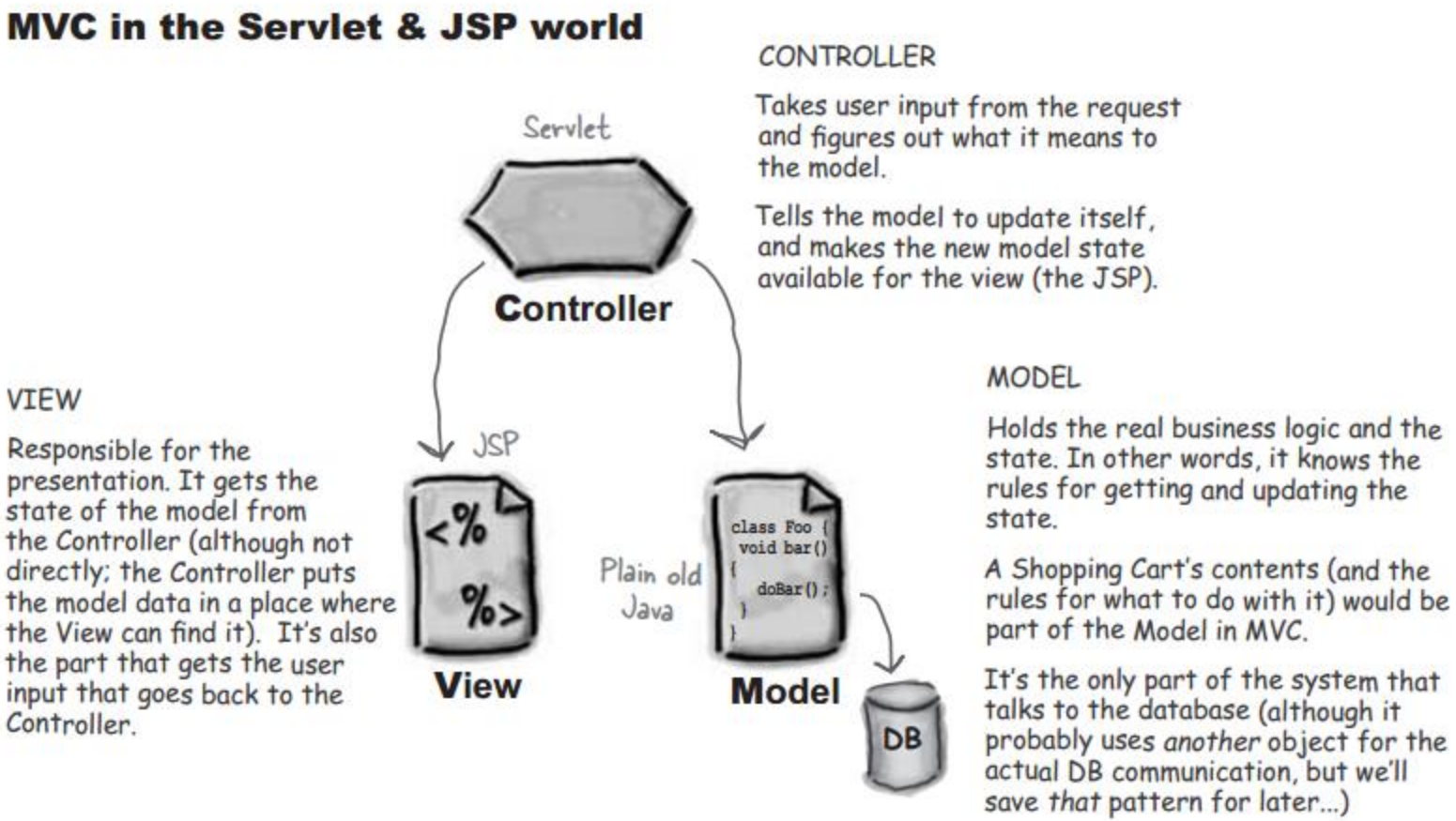
Following topics is to be covered in the session

- MVC Architecture

MVC Architecture

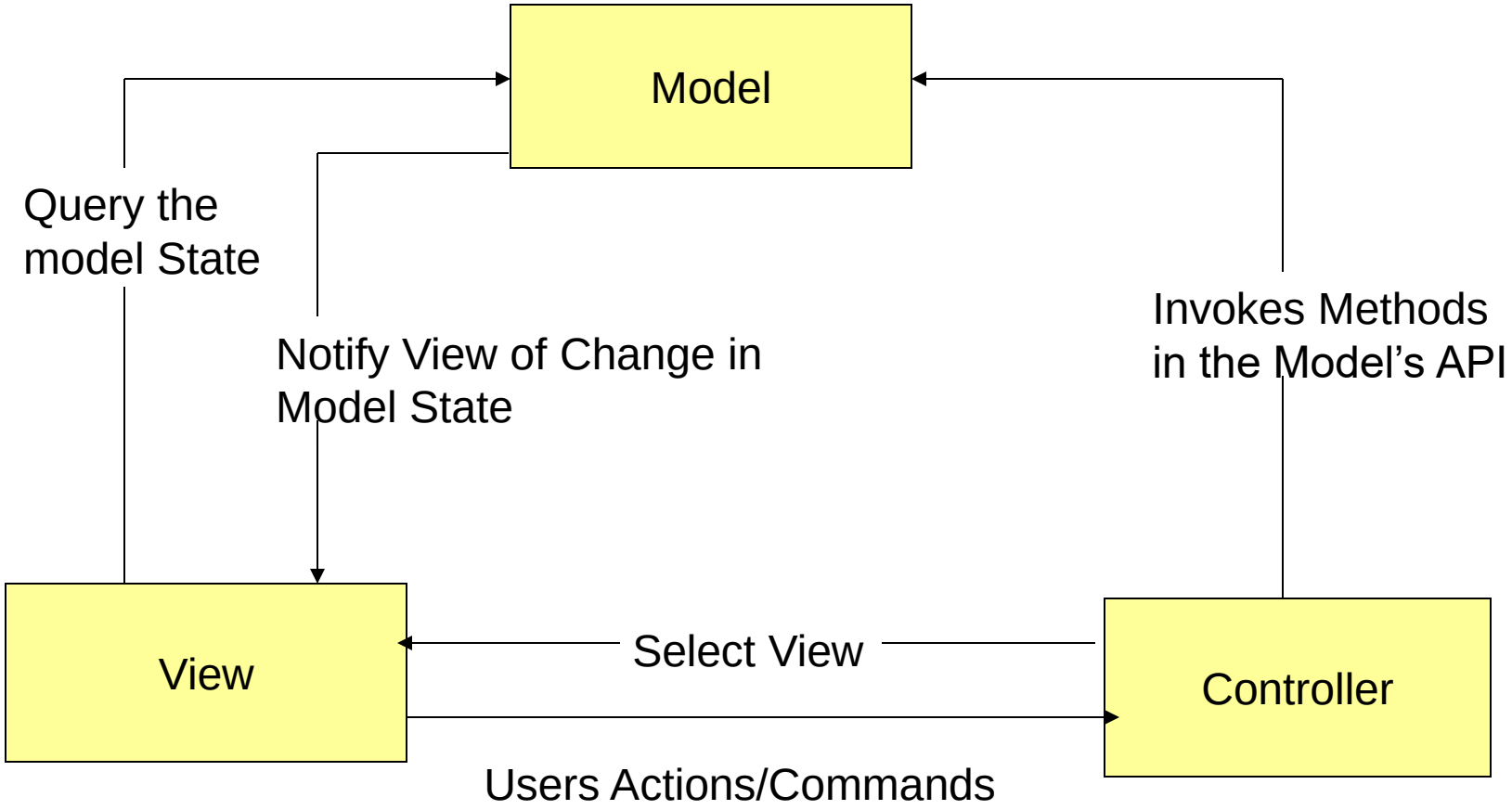
- Model-View-Controller (MVC) is an architectural pattern used in software engineering.
- Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other.
- It is common to split an application into separate layers: presentation (UI), domain logic, and data access.
- In MVC the presentation layer is further separated into view and controller.
- MVC encompasses more of the architecture of an application than is typical for a design pattern.

MVC Concept..



MVC Architecture

- Model - View - Controller



MVC Architecture

Model

- Represents an application’s data and contains the logic for accessing and manipulating that data.
- Any data that is part of the persistent state of the application should reside in the model objects.
- Domain logic adds meaning to raw data (e.g. calculating if today is the user's birthday, or the totals, taxes, and shipping charges for shopping cart items).
  - Many applications use a persistent storage mechanism (such as a database) to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the Model.

MVC Architecture

View

- Is responsible for rendering the state of the model.
- The presentation semantics are encapsulated within the view, therefore model data can be adapted for several different kinds of clients.
- The view modifies itself when a change in the model is communicated to the view.
- A view forwards user input to the controller.
- Multiple views can exist for a single model for different purposes.

MVC Architecture

Controller

- Is responsible for intercepting and translating user input into actions to be performed by the model.
- Is responsible for selecting the next view based on user input and the outcome of model operations.
  - Processes and responds to events, typically user actions, and may invoke changes on the model.
  - MVC is often seen in web applications, where the view is the actual HTML page, and the controller is the code that gathers dynamic data and generates the content within the HTML. Finally, the model is represented by the actual content, usually stored in a database or XML files, and the business rules that transform that content based on user actions.

MVC Architecture

- Though MVC comes in different flavors, control flow generally works as follows:
  1. The user interacts with the user interface in some way (e.g. presses a button).
  2. A controller handles the input event from the user interface, often via a registered handler or callback.
  3. The controller notifies the model of the user action, possibly resulting in a change in the model's state. (e.g. controller updates user's Shopping cart).
  4. A view uses the model (indirectly) to generate an appropriate user interface (e.g. the view produces a screen listing the shopping cart contents). The view gets its own data from the model. The model has no direct knowledge of the view.
  5. The user interface waits for further user interactions, which begins the cycle anew.

- **Best Practices** : Due to various advantages of MVC, it is recommended to use it in developing web applications. Because of MVC, code becomes maintainable and simplified.
- Also it is technology independent i.e. you can use it to create any application like desktop / Single Page Application (using Java Script, MEAN stack) / applications created using .Net framework.
- Various Java frameworks are based on MVC. Like Struts , Spring etc.

- So always use MVC while creating web application.

Knowledge check…

- Who is ideally responsible for following tasks according to MVC : JSP / Java Beans / Servlet ?

No	Task	JSP	Java Bean	Servlet
1	Form designing			
2	Communicating to DB			
3	Receiving user request data			
4	Displaying output /result to user			
5	Controlling the flow in application			
6	Server side validations			

Summary

In this session, we have covered,

- MVC Architecture

Thank You

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for informational purposes and private circulation only and do not constitute an offer to buy or sell any securities mentioned therein. They do not purport to be a complete description of the markets conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Unless specifically noted, TechM is not responsible for the content of these presentations and/or the opinions of the presenters. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.