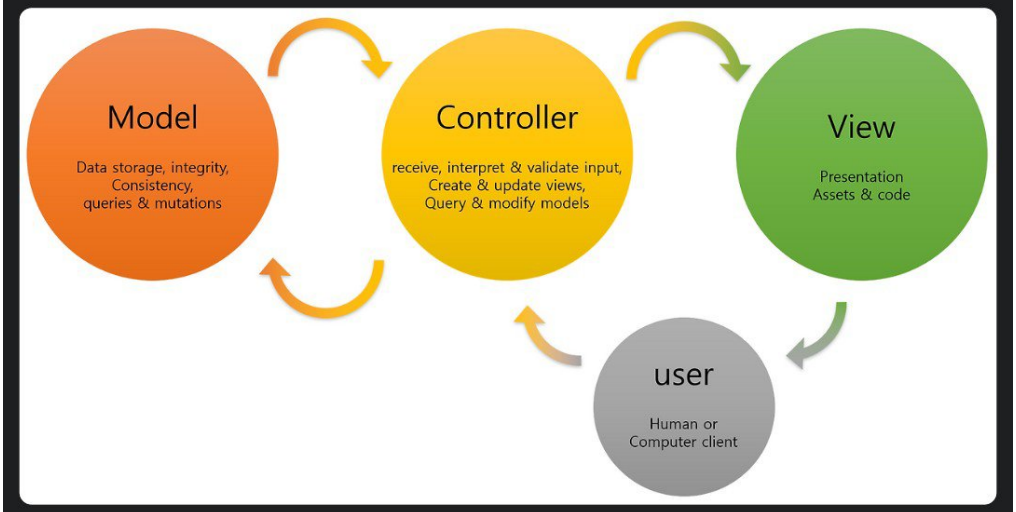
(Model-View-Controller) pattern.

💡Structure of the MVC pattern



MVC Pattern Arcitecture

Model

- **Definition:** Responsible for the application's data and handles business logic.
- **Role:** Process and store data displayed to users, and implement appropriate processing methods when data changes.
- **Implementation:** It is expressed as a Java object and serves to store data of a specific domain.

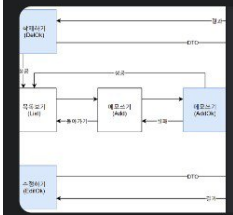
View

- **Definition:** Responsible for visual UI and presents information to users.
- **Role:** Displays data received from the Model on the screen and transmits user input to the Controller.
- **Implementation:** Mainly implemented using template engines such as HTML, JSP, and Thymeleaf.

Controller

- **Definition:** It serves to connect the Model and View.
- **Role:** Receives user requests, executes business logic for the requests, stores the results in the Model, and then passes them to the appropriate View.
- **Implementation:** It is mainly implemented as a Java class and registered with Spring through the @Controller annotation.

DAO, DTO



[JDBC] MVC Model DAO, DTO (Memo Oper…

🍁MVC Model [JDBC] MVC Model (Model, View, Controller) 🍁Servlet + JSP Servlet Advantage: Easy to code Java code because it is based on Java. Disadvantage: Inconvenient client code. JSP Advantage…

isaac-christian.tistory.com

Please refer to the above article for DAO and DTO of the MVC Model.

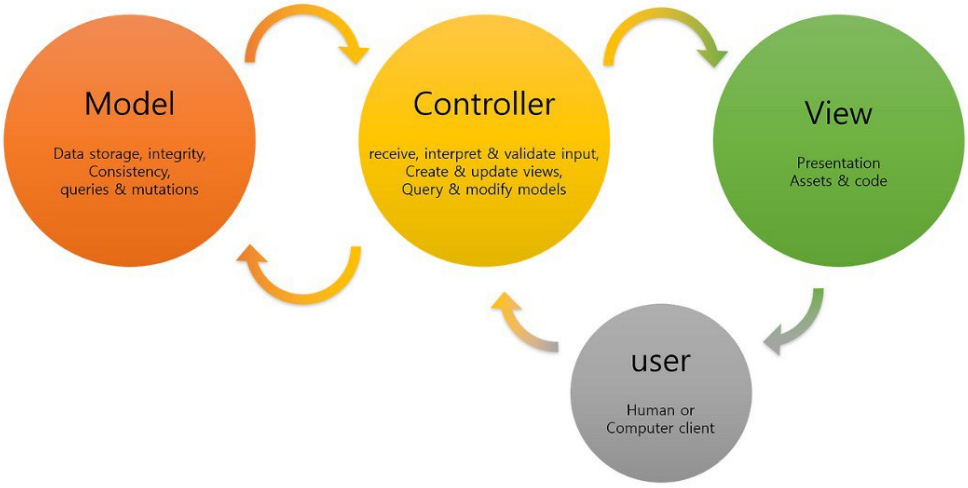
💡MVC1 / MVC2 / Spring MVC

MVC1 vs. MVC2

MVC1 is a structure in which JSP takes on all roles, allowing for rapid development, but has the disadvantage of

Spring MVC Framework is one of the most widely used architectures in web development and is based on the MVC (Model-View-Controller) pattern.

💡Structure of the MVC pattern



MVC Pattern Arcitecture

Model

- **Definition:** Responsible for the application's data and handles business logic.
- **Role:** Process and store data displayed to users, and implement appropriate processing methods when data changes.
- **Implementation:** It is expressed as a Java object and serves to store data of a specific domain.

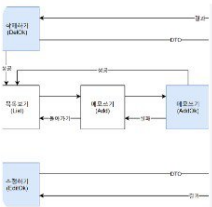
View

- **Definition:** Responsible for visual UI and presents information to users.
- **Role:** Displays data received from the Model on the screen and transmits user input to the Controller.
- **Implementation:** Mainly implemented using template engines such as HTML, JSP, and Thymeleaf.

Controller

- **Definition:** It serves to connect the Model and View.
- **Role:** Receives user requests, executes business logic for the requests, stores the results in the Model, and then passes them to the appropriate View.
- **Implementation:** It is mainly implemented as a Java class and registered with Spring through the @Controller annotation.

DAO, DTO



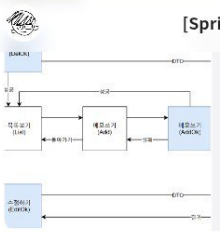
[JDBC] MVC Model DAO, DTO (Memo Oper…

🍁MVC Model [JDBC] MVC Model (Model, View, Controller) 🍁Servlet + JSP Servlet Advantage: Easy to code Java code because it is based on Java. Disadvantage: Inconvenient client code. JSP Advantage…

isaac-christian.tistory.com

Please refer to the above article for DAO and DTO of the MVC Model.

💡MVC1 / MVC2 / Spring MVC



[Spring] Spring MVC Framework: Structure and Operation of the MVC Pattern

[JDBC] MVC Model DAO, DTO (Memo Oper…

🍁MVC Model [JDBC] MVC Model (Model, View, Controller) 🍁Servlet + JSP Servlet Advantage: Easy to code Java code because it is based on Java. Disadvantage: Inconvenient client code. JSP Advantage…

isaac-christian.tistory.com

Please refer to the above article for DAO and DTO of the MVC Model.

💡MVC1 / MVC2 / Spring MVC

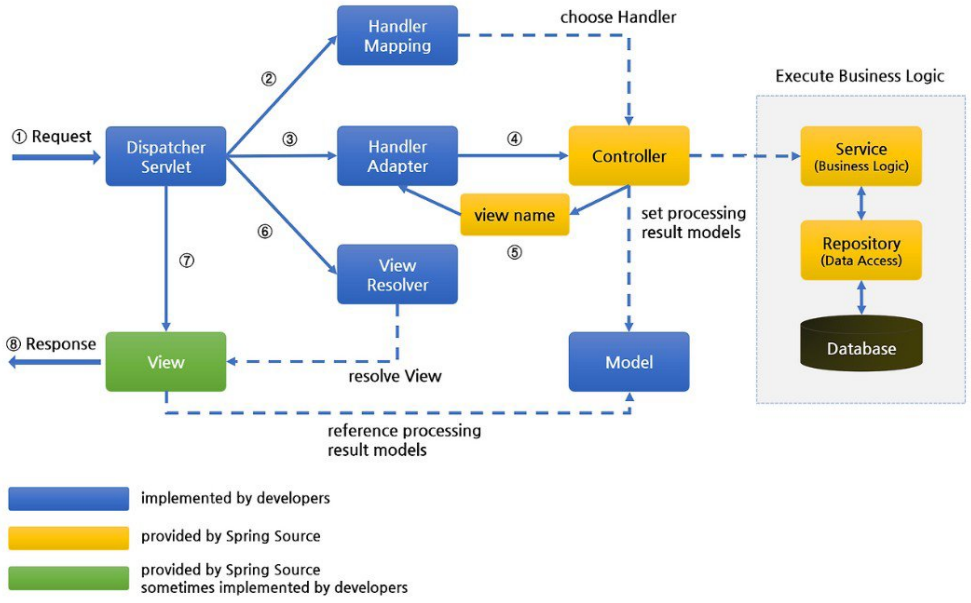
MVC1 vs. MVC2

MVC1 is a structure in which JSP takes on all roles, allowing for rapid development, but has the disadvantage of being difficult to maintain.
MVC2 is a version of the pattern that separates roles and improves extensibility by adding a controller.

Spring MVC vs. Traditional MVC Pattern

Spring MVC is an evolved version of the MVC2 pattern, providing flexibility and extensibility.
It has the feature of increasing productivity and improving maintainability by supporting Spring's DI (Dependency Injection) and AOP (Aspect-Oriented Programming).

💡Structure of Spring MVC Framework



Spring MVC Framework

📌Movement process

1. DispatcherServlet receives the client request.
2. DispatcherServlet calls HandlerMapping to pass request information . It analyzes the request information (URL) and selects the appropriate Controller.
3. DispatcherServlet then calls HandlerAdapter, which finds the appropriate method for the requested URL.
4. The HandlerAdapter delegates the request to the Controller. The Controller processes business logic and stores the results in a Model object to be passed to the View.
5. The controller returns the view name to the DispatcherServlet.

📌Movement process

1. DispatcherServlet receives the client request.
2. DispatcherServlet calls HandlerMapping to pass request information . It analyzes the request information (URL) and selects the appropriate Controller.
3. DispatcherServlet then calls HandlerAdapter, which finds the appropriate method for the requested URL.
4. The HandlerAdapter delegates the request to the Controller. The Controller processes business logic and stores the results in a Model object to be passed to the View.
5. The controller returns the view name to the DispatcherServlet.
6. DispatcherServlet calls ViewResolver to find the appropriate View based on the view name returned by the Controller.
7. DispatcherServlet passes the processing result to the View object and displays it.
8. The View object calls the corresponding View. The View retrieves the objects required to display the screen from the Model object, processes them on the screen, and passes them to the Client.

DispatcherServlet

DispatcherServlet is a front controller that receives and processes all user requests in Spring MVC, and serves as the entry point to the web application.
This servlet's configuration is mainly done through the web.xml file or WebApplicationInitializer .
DispatcherServlet handles requests received from clients by forwarding them to the appropriate controller.

HandlerMapping

HandlerMapping determines which controller will handle the request from the DispatcherServlet. It analyzes the user's request URL and determines which controller will handle it.
It can be configured using an XML file or Java config annotations.

HandlerAdapter

HandlerAdapter requests the execution of a mapped controller and converts the result into a ModelAndView object. This serves to flexibly support various types of controllers, allowing for a variety of handling methods for how each controller is executed and how its results are processed.

Controller

The Controller receives a user request, executes the business logic for the request, saves the result in the Model , and then passes it to the appropriate View .
It is mainly implemented as a Java class and registered with Spring through the @Controller annotation.

ViewResolver

ViewResolver is responsible for finding the actual View object based on the view name returned from the controller. This View object then creates the actual screen and responds to the client.
Spring supports various View templates, and ViewResolver finds the templates.

View

The View is responsible for creating the screen and responding to the client. It uses data received from the Controller to visually present it to the user.
It is mainly implemented using template engines such as HTML, JSP, and Thymeleaf.

References and Materials

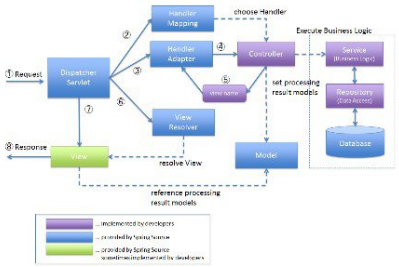
1. MVC Architecture in Java. javatpoint. 2024.01.15. https://www.javatpoint.com/mvc-architecture-in-java
2. [Spring] MVC Pattern & Spring Framework MVC. Aridom's Development. 2024.01.15. https://aridom.tistory.com/61

Spring Reference Document.

> Spring's web MVC framework is, like many other web MVC frameworks, request-driven, designed around a central Servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications. Spring's DispatcherServlet however, does more than just that. It is completely integrated with the Spring IoC container and as such allows you to use every other feature that Spring has.

2.2.1. Overview of Spring MVC Processing Sequence

The processing flow of Spring MVC from receiving the request till the response is returned is shown in the following diagram.

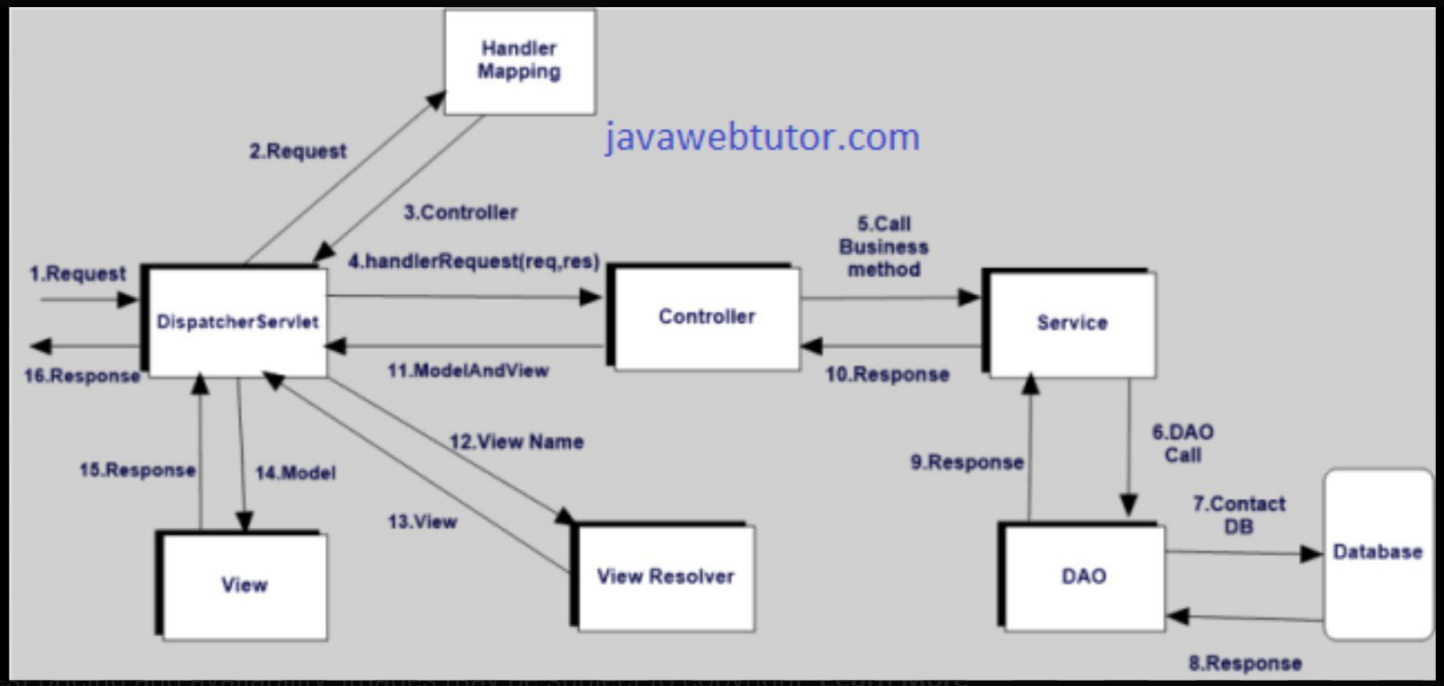


1. `DispatcherServlet` receives the request.
2. `DispatcherServlet` dispatches the task of selecting an appropriate `controller` to `HandlerMapping`. `HandlerMapping` selects the controller which is mapped to the incoming request URL and returns the `(selected Handler)` and `Controller` to

1. `DispatcherServlet` receives the request.
2. `DispatcherServlet` dispatches the task of selecting an appropriate `controller` to `HandlerMapping`. `HandlerMapping` selects the controller which is mapped to the incoming request URL and returns the `(selected Handler)` and `Controller` to `DispatcherServlet`.
3. `DispatcherServlet` dispatches the task of executing of business logic of `Controller` to `HandlerAdapter`.
4. `HandlerAdapter` calls the business logic process of `Controller`.
5. `Controller` executes the business logic, sets the processing result in `Model` and returns the logical name of view to `HandlerAdapter`.
6. `DispatcherServlet` dispatches the task of resolving the `View` corresponding to the View name to `ViewResolver`. `ViewResolver` returns the `View` mapped to View name.
7. `DispatcherServlet` dispatches the rendering process to returned `View`.
8. `View` renders `Model` data and returns the response.

2.2.2. Implementation of each component

Among the components explained previously, the extendable components are implemented.



In Spring Boot, DAO (Data Access Object) and DTO (Data Transfer Object) are distinct design patterns used for different purposes within the application architecture.

DAO (Data Access Object):

Purpose:

The DAO pattern is used to abstract and encapsulate all data access operations related to a specific entity or set of entities. It provides a clear separation between the business logic and the persistence layer.

Functionality:

A DAO typically contains methods for performing CRUD (Create, Read, Update, Delete) operations on a database or other data sources. It hides the underlying data access technology (e.g., JDBC, JPA, Hibernate) from the service layer.

Example:

A `UserDao` might have methods like `saveUser(User user)`, `findUserById(long id)`, `updateUser(User user)`, and `deleteUser(long id)`.

DTO (Data Transfer Object):

Purpose:

A DTO is a simple Java object used to transfer data between different layers of an application, particularly between the service layer and the presentation layer (e.g., REST controllers).

Functionality:

DTOs typically contain only data fields (private fields with public getters and setters) and no business logic. They are designed to encapsulate a specific subset of data relevant for a particular use case

Example:

A `UserDao` might have methods like `saveUser(User user)`, `findUserById(long id)`, `updateUser(User user)`, and `deleteUser(long id)`.

DTO (Data Transfer Object):

Purpose:

A DTO is a simple Java object used to transfer data between different layers of an application, particularly between the service layer and the presentation layer (e.g., REST controllers).

Functionality:

DTOs typically contain only data fields (private fields with public getters and setters) and no business logic. They are designed to encapsulate a specific subset of data relevant for a particular use case, often for API responses or requests.

Example:

If a `User` entity has many fields, but an API only needs to display `username` and `email`, a `UserDto` could be created with just those two fields, reducing the amount of data transferred.

Key Differences and Relationship:

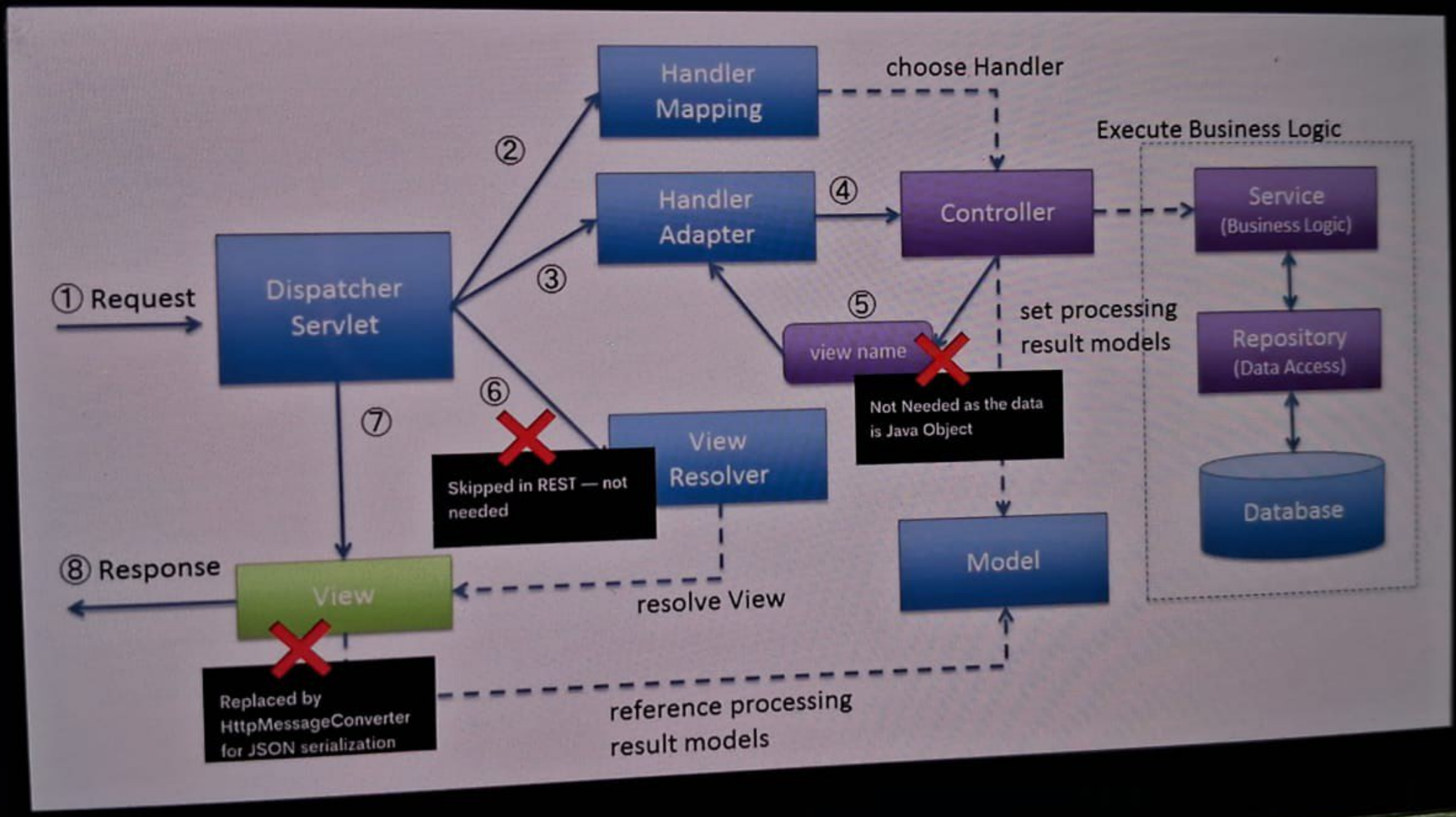
Responsibility:

DAOs are responsible for interacting with the data source, while DTOs are responsible for carrying data between layers.

Data Representation:

DAOs often work with entity objects (which map directly to database tables), while DTOs can represent a subset or a combination of data from multiple entities, tailored for specific communication needs.

Spring MVC with REST APIs



How does a Web Server works in Spring Boot?