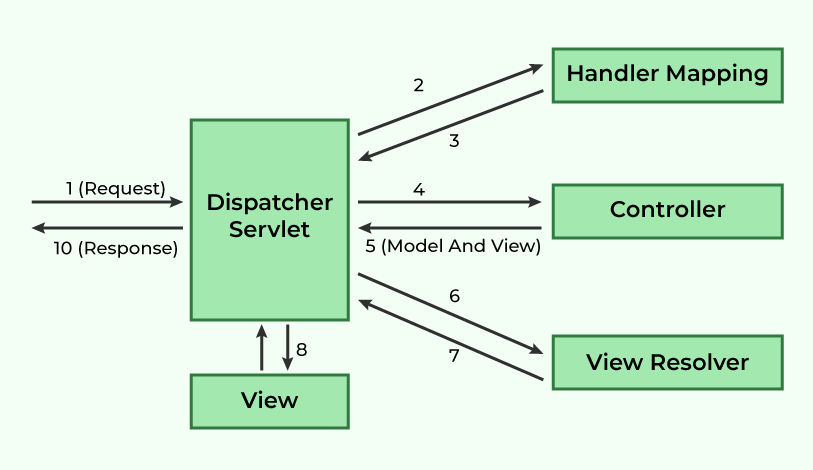
***Spring Web MVC:-***

1. By using Spring Web MVC we can develop web based application as well Distributed application.
2. It is sub framework of Spring framework.
3. It is built on top of Servlet API



***Flow of Request Handling in Spring MVC***

1. **Client Request → DispatcherServlet**
   * The client (browser or API consumer) sends an **HTTP request**.
   * The request is first received by **DispatcherServlet**, which acts as the front controller.
2. **DispatcherServlet → HandlerMapping**
   * The **DispatcherServlet** consults the **HandlerMapping** to determine the appropriate controller method for handling the request.
3. **HandlerMapping → DispatcherServlet**
   * The **HandlerMapping** returns the matched controller method to the **DispatcherServlet**.
4. **DispatcherServlet → Controller**
   * The **DispatcherServlet** forwards the request to the identified **Controller**.
5. **Controller → DispatcherServlet (Model and View)**
   * The **Controller** processes the request and returns a **ModelAndView** object, which contains:
     + **Model** (data to be displayed)
     + **View Name** (logical name of the view to be rendered)
6. **DispatcherServlet → ViewResolver**
   * The **DispatcherServlet** forwards the **View Name** to the **View Resolver** to resolve the actual view.
7. **ViewResolver → DispatcherServlet (Resolved View)**
   * The **View Resolver** returns the fully resolved view object.
8. **DispatcherServlet → View**
   * The **DispatcherServlet** forwards the model data to the resolved **View**.
9. **View Renders Data**
   * The **View** generates the final HTML response using the provided model data.
10. **Response Sent to Client**

* The final response is sent back to the client.

***Components & Their Duties***

1. **DispatcherServlet (Front Controller)**
   * Central controller that manages the complete request-processing workflow.
   * Delegates tasks to appropriate components.
2. **HandlerMapping**
   * Maps requests to corresponding controller methods.
   * Uses configurations like **annotation-based (@RequestMapping)** or XML-based mappings.
3. **Controller**
   * Contains business logic.
   * Processes the request and returns a **ModelAndView**.
4. **ModelAndView**
   * Holds both **model data** and **view name**.
   * Helps in rendering dynamic content.
5. **ViewResolver**
   * Resolves logical view names into actual view implementations (e.g., JSP, Thymeleaf, FreeMarker).
6. **View**
   * The final UI template that presents the data to the client.

MVC Components:-

1. Dispatcher Servlet:-

* It is the class which is used to handle request and response.
* It is also called as Front controller.
* **@Controller:-** It is used to represent Java class as MVC Controller / Component.
  + - It is class-level annotation.
    - It consist **@Component** annotation.
    - It is used to return Web page name form controller method.
    - It consist multiple http request binded methods.

For binding http request to particular controller method we can use :-

**@RequestMapping** annotation

* Sending data through request:-

We can send data using URL by using QueryParameter, http request body, http request header,URL.

***QueryParameter:-***

* Data present in QueryParameter in key-value pair.
* Username=xyz🡪 Key , Password= pqr123 🡪Value
* Data starts with **?** and separated with **&** symbol.
* We can take data from QueryParameter by calling respective key.
* So in backend we can take data by using **@RequestParam** annotation.
* Data always present at the end of URL.

1. HandlerMapper / URL Mapper
2. Controller
3. Model and view
4. ViewResolver

IQ)When we already have RequestParam then why to use ModelAttribute?

IQ)What is InternalResourceViewResolver?

ModelInterface

IQ)Difference between Model & HttpSession?