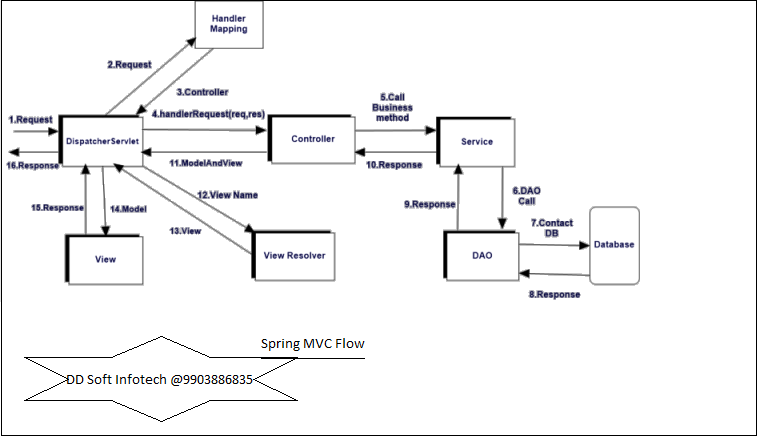
***Spring MVC Flow:***

Spring MVC or Spring Web MVC framework is an open source Spring framework which follows MVC architectural pattern and is used to develop web applications in very easy and simple steps. Spring MVC is a three tier/layer architectural pattern where M stands for Model, V stands for View and C stands for Controller.

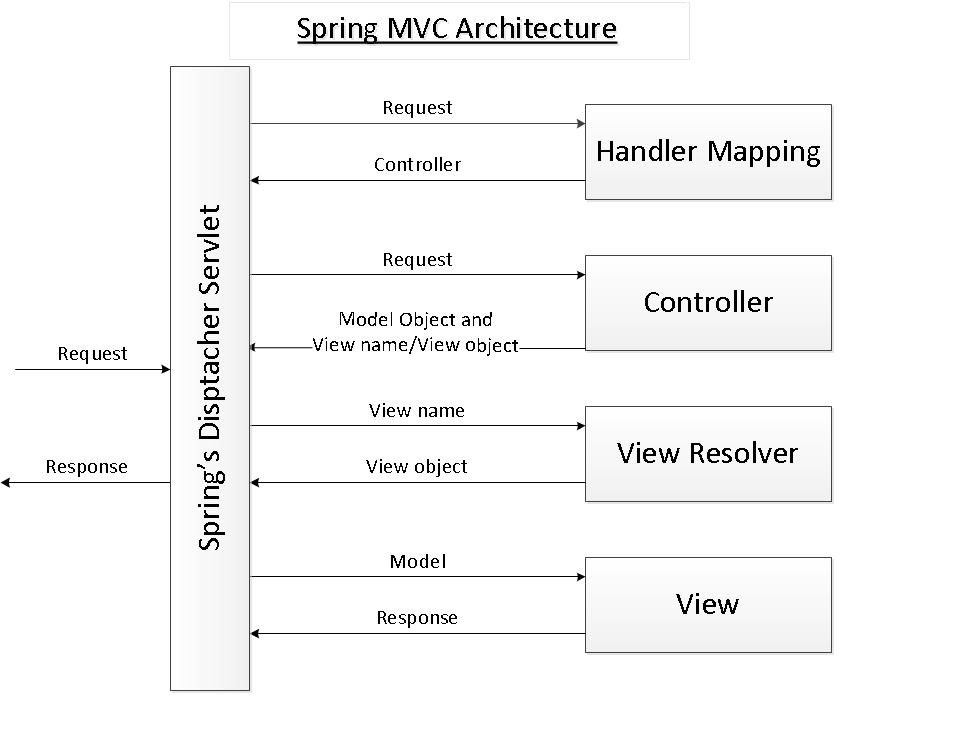
It provides a convenient way to develop a java based web application. It has a central servlet called as DispatcherServlet which is well known as front controller that intercepts all the requests, identify the appropriate handler i.e. controllers and render views to the client. It is defined at org.springframework.web.servlet.DispatcherServlet in org.springframework.web package.

By using Spring MVC we can build flexible and loosely coupled web applications.The MVC design pattern helps in seperating the business logic, presentation logic and controller logic. Models are responsible for encapsulating the application data. The Views render response to the user with the help of the model object . Controllers are responsible for receiving the request from the user and calling the back-end services.

* Client(Browser) requests for a Resource in the web Application.
* The Spring front controller i.e, DispatcherServlet first receives the request.
* DispatcherServlet consults the HandlerMapping to identify the particular controller for the given URL.
* HandlerMapping identifies the controller for the given request and sends to the DispatcherServlet.
* DispatcherServlet will call the handleRequest(request,response) method on Controller. A Controller is developed by writing a simple java class which implements Controller interface or extends its adapter class.
* Controller will call the business method according to business requirement.
* Service class will call the DAO class method for business data.
* DAO interacts with DB to get data.
* DAO returns same data to service.
* Fetched data will be processed according to business requirement and return results to Controller.
* The Controller returns the Model and View in the form of Object back to the Controller i.e, DispatcherServlet.
* The front controller i.e, DispatcherServlet then tries to resolve the actual View which may be JSP,velocity or Free Marker by consulting the View Resolver Object.
* ViewResolver selected view is rendred back to the DispatcherServlet.
* DispatcherServletconsult the particular view with the model.
* View executes and returns HTML output to the DispatcherServlet.
* DispatcherServlet will sends the output to the Browser.







**MVC design pattern**

This design pattern helps us develop loosely coupled application by segregating various concerns into different layers. MVC design pattern enforces the application to be divided into three layers, *Model*, *View* and *Controller*.

**Model:** This represents the application data.

**View:** This represents the application’s user interface. View takes model as the input and renders it appropriately to the end user.

**Controller:** The **Controller** is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

<http://www.codejava.net/frameworks/spring/understanding-spring-mvc?start=1>

https://www.dineshonjava.com/spring-mvc-form-handling-example/

<http://www.wideskills.com/spring/spring-mvc-framework>

<http://javawebtutor.com/articles/spring/spring-mvc-tutorial.php>

## Defining Spring Controller

Following is the controller class which is mapped for /users. This class acts as a handler class in spring mvc. It handles the request that is mapped for /users. It processes the requests and return model and view to the front controller.

The **@Controller** annotation is used to mark any java class as a controller class

**@RequestMapping :** This annotation is used at both the class and method level. The @RequestMapping annotation is used to map web requests onto specific handler classes and handler methods. When @RequestMapping is used on the class level, it creates a base URI for which the controller will be used. When this annotation is used on methods, it will give you the URI on which the handler methods will be executed. From this, you can infer that the class level request mapping will remain the same whereas each handler method will have their own request mapping.

Sometimes you may want to perform different operations based on the HTTP method used, even though the request URI may remain the same. In such situations, you can use the method attribute of @RequestMapping with an HTTP method value to narrow down the HTTP methods in order to invoke the methods of your class.

Using the **@Autowired** annotation the container can wire your beans automatically. By default autowire is done by type.

1. Normally we specify dependencies with the help of constructor arguments or property set. But with the help of auto wire, we don’t need to specify them they are automatically resolved.
2. It reduces the amount of code we write to specify dependencies.