In Spring Core, We saw that there is a class called **ClassPathXmlApplicationContext(“config.xml”).** When we create object of that class then spring initialize the IOC Container and whatever **config.xml** name you passed it, spring create all the beans object which have been configured in it and put them into IOC container.

But in case of **Spring MVC,** story is different.

First of all, we have different context class named as **WebApplicationContext.** Secondly, we don’t need to create object of this class manually. Spring will instantiate this class for us. So the question arise is how spring get to know which **.xml** file name it should pass as argument to this class.

To know the answer, we should understand the flow of application first.

* Initially, when spring start executing our application, it always refer to **web.xml** file which tells the spring about **Front Controller**.
* Now what spring does is :
  + Spring first instantiate the front controller i.e. **DispatcherServlet**
  + Spring has already written the code of object creation of **WebApplicationContext** somewhere like below:

**ApplicationContext** context = new **WebApplicationContext(fc+”-servlet.xml”);**

Where, fc is name of the frontcontroller

* + Now, suppose the front-controller name is **myfc**, whenever spring strike the above line it always try to search for a **.xml** file inside **WEB-INF** folder, which name should start with front-controller name and ends with **-servlet**. For ex : if our front controller name is **myfc** then the **.xml** file name should be **myfc-servlet.xml**
  + If the file not found then spring will throw FileNotFoundException.
  + Else spring start scanning the package i.e. defined in the tag <context:component-scan> inside **web.xml** file. Whatever class annotate with @Controller annotation inside the given package spring create their object and put inside WebApplicationContext (Web IOC Container).
* The Controller objects is now in the IOC container. Obviously, every Controller should be map with a specific url through **@RequestMapping** annotation and returning some response with **@ResponseBody annotation.**
* Now whenever user hit the valid url for our application it should be achieve the expected response.