Spring is famous for dependency injection/ dependency inversion. When there are two objects dependent on each other. We use the idea of decoupling to take away the dependency and use the dependency in the form and the way we need to our agenda, our business need and to perform the core business functionality and not focus on non-functional aspect of the requirement.

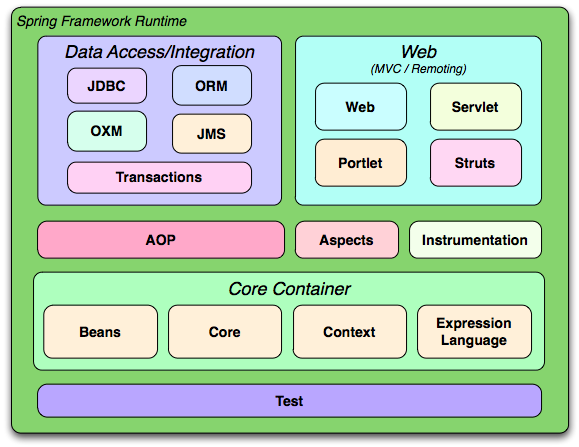
**Web container (Servlet container) Vs Spring Container**

[**https://www.youtube.com/watch?v=xlWwMSu5I70&index=3&list=PLC97BDEFDCDD169D7**](https://www.youtube.com/watch?v=xlWwMSu5I70&index=3&list=PLC97BDEFDCDD169D7)

**About DAO , Dependency Injection with annotations, Dependency injection with xml**

[**http://www.vogella.com/tutorials/SpringDependencyInjection/article.html**](http://www.vogella.com/tutorials/SpringDependencyInjection/article.html)

[**https://www.springbyexample.org/examples/core-concepts-dependency-injection-to-the-rescue.html**](https://www.springbyexample.org/examples/core-concepts-dependency-injection-to-the-rescue.html)



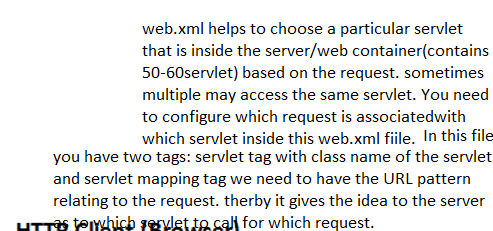
<https://www.youtube.com/watch?v=-weKK-oNuhA&index=4&list=PL9ooVrP1hQOEfi91PCFQMawtBJrPpir7y&t=1153s>

(40:00 onwards)

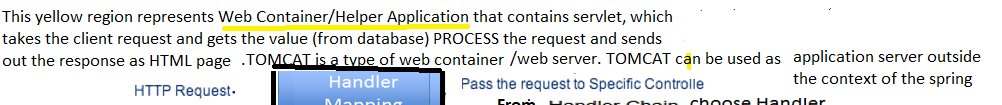


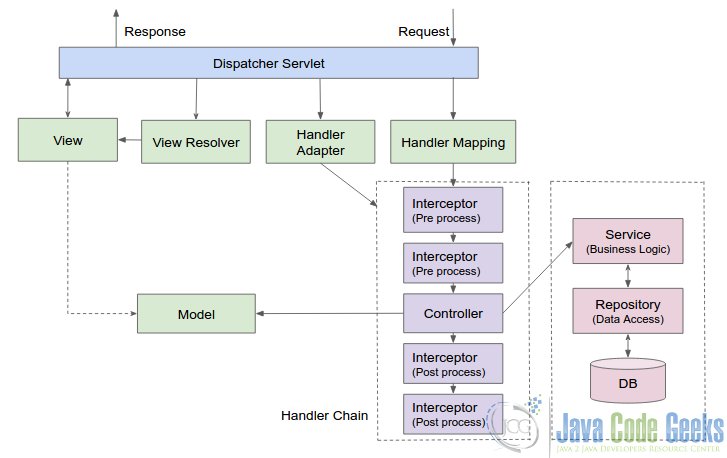
The **Model** encapsulates the application data and in general they will consist of POJO.

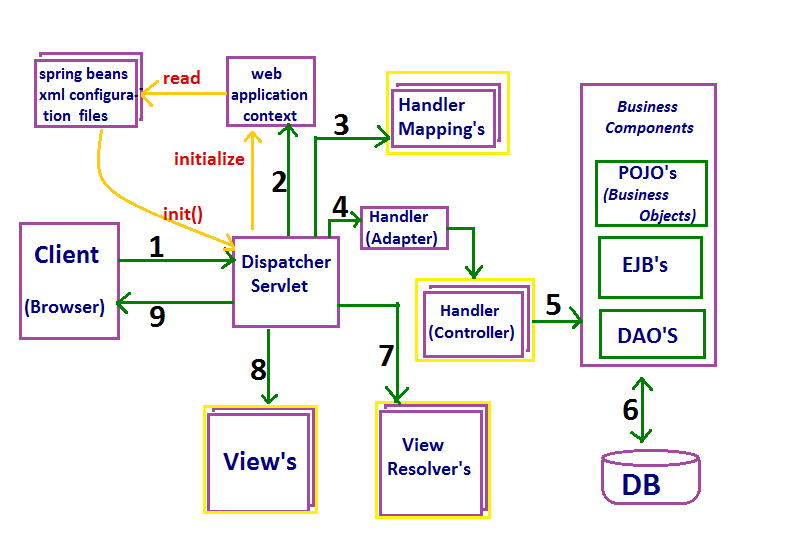
Web.xml

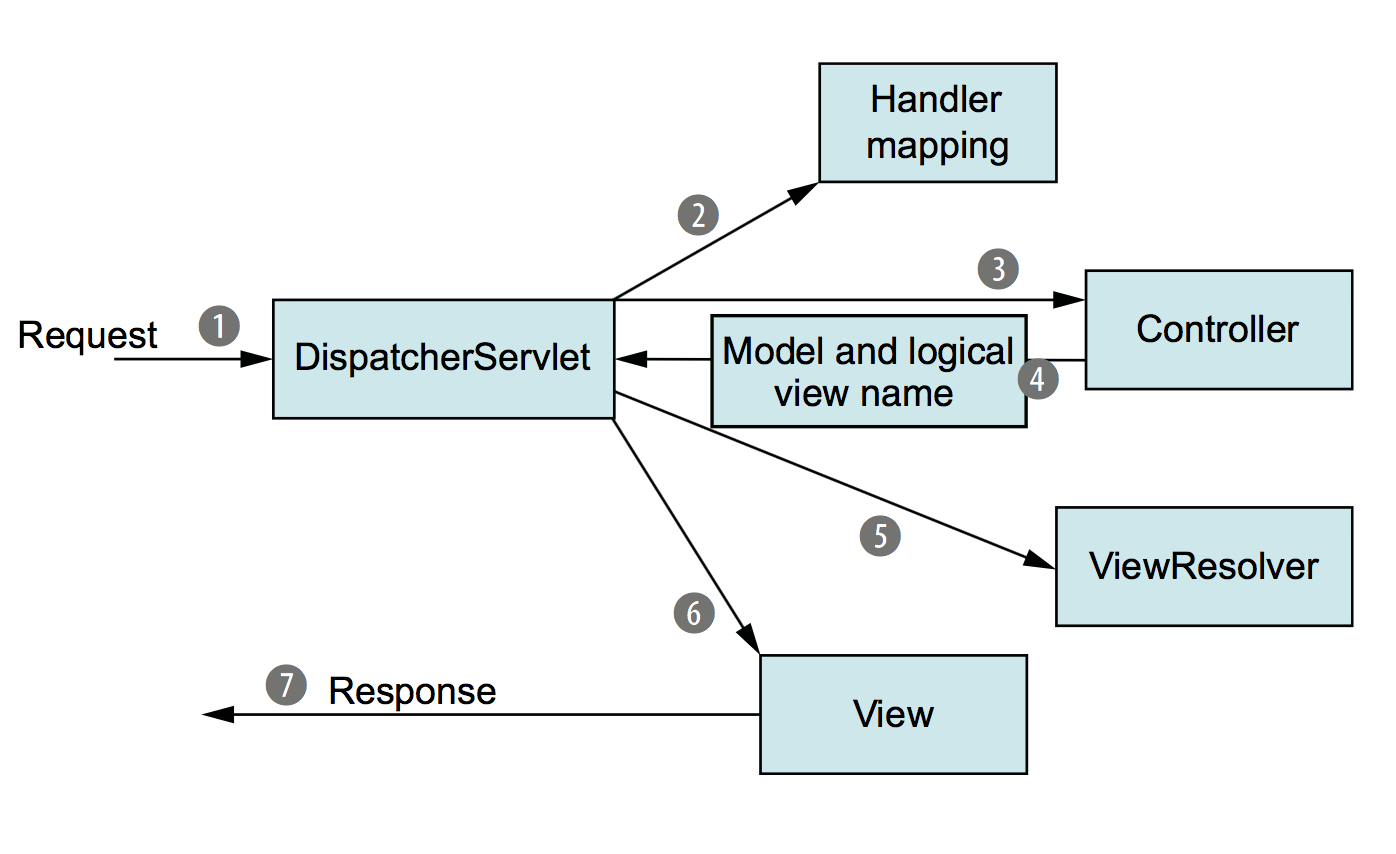


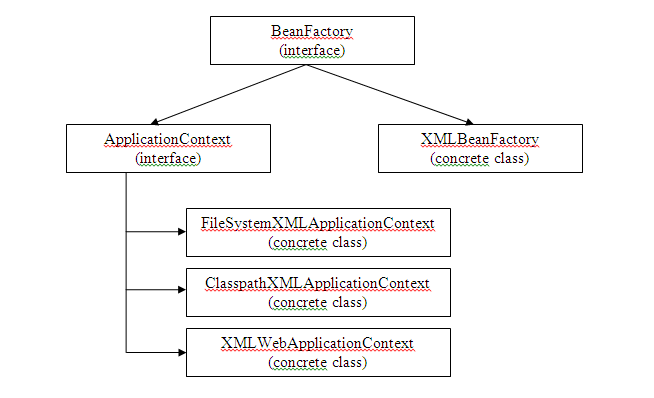
<https://terasolunaorg.github.io/guideline/1.0.1.RELEASE/en/Overview/SpringMVCOverview.html>











28:00 onwards…

<https://www.youtube.com/watch?v=8a2vcpxZdME>

1-10 mins

<https://www.youtube.com/watch?v=RlfnE_OEmF0>

**ioc vs servlet container**

They are very different concepts also if their names are similar.

A **Servlet Container** or Web Container (like Tomcat) is an implementation of various Java EE specifications like Java Servlet, JSP, etc. Put in a simple way, it is an environment where Java web applications can live. A web server + Java support.

A **Spring Container** on the other hand, is the core and the engine of the Spring Framework. It is an IoC Container, infact it handles Spring applications lifecycle creating new beans and injecting dependencies.

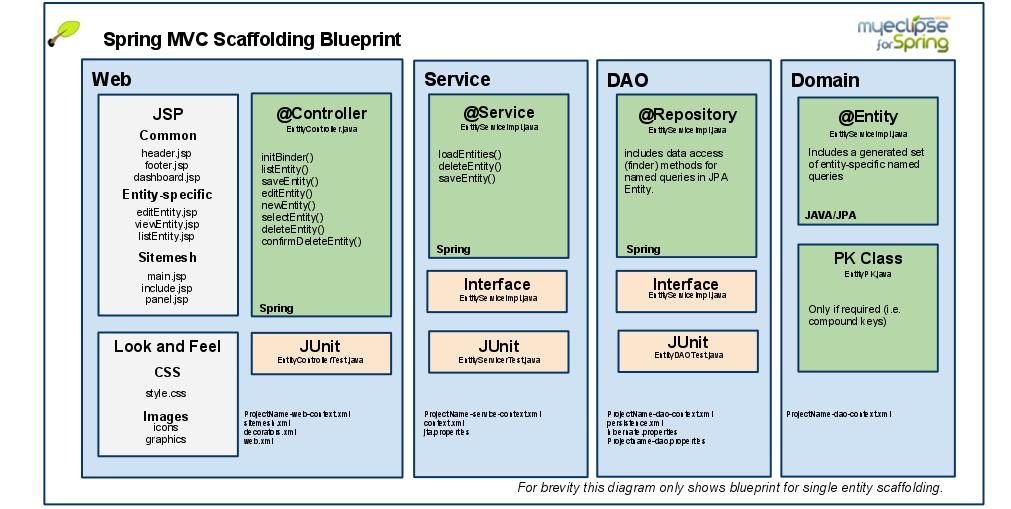
Because a Spring application can be a web application, a Spring Container can "live" inside a Web Container.

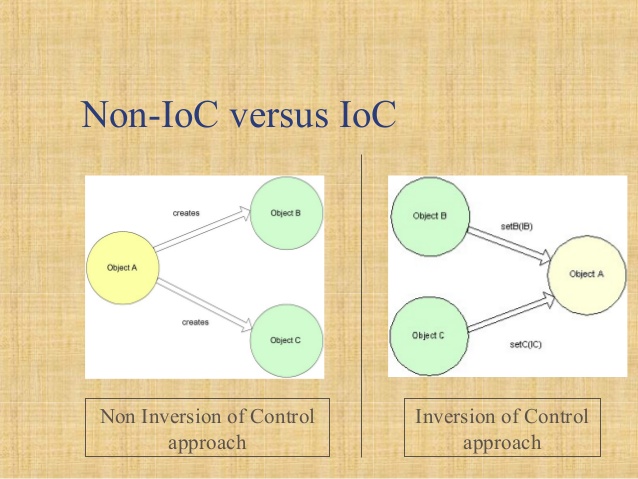
For further information:

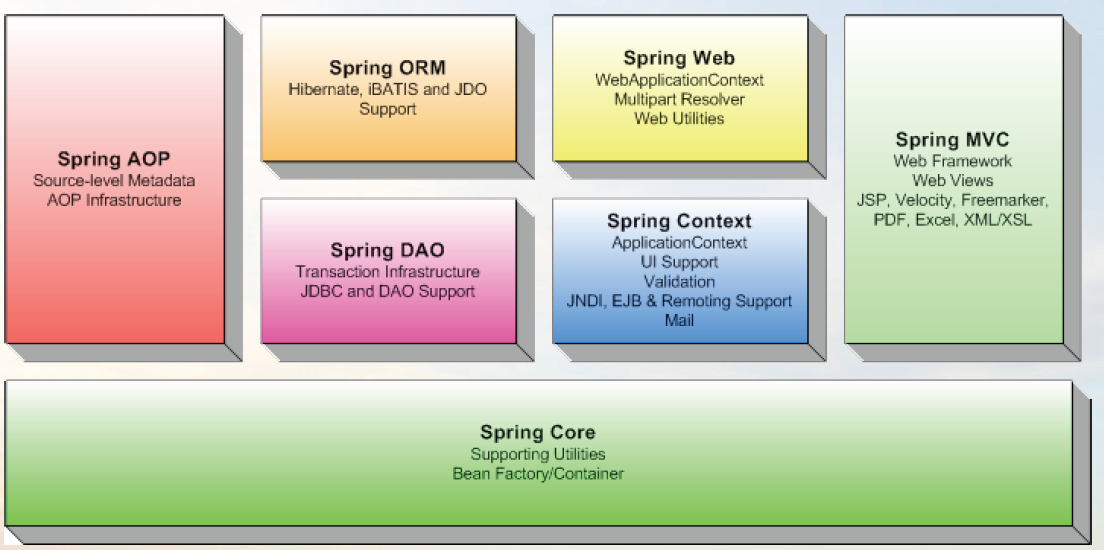
* about Servlet Container: <http://en.wikipedia.org/wiki/Web_container>
* about Spring Container: <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html>

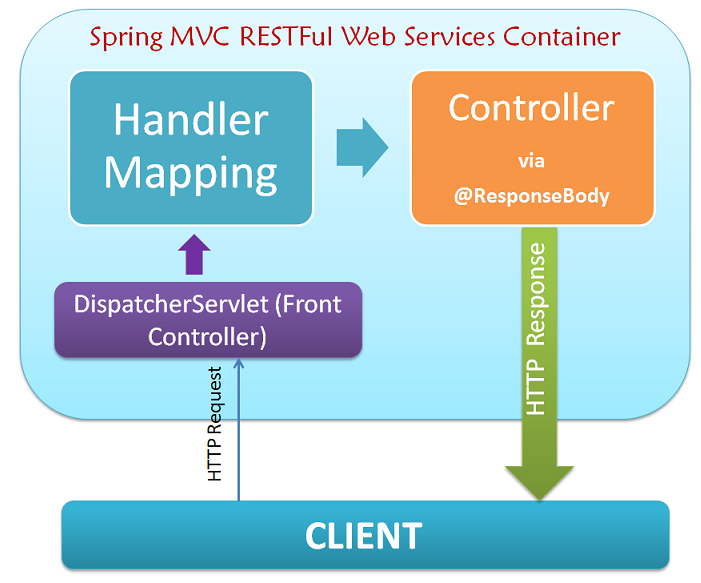
**Explaining the Spring Architecture:**

<https://stackoverflow.com/questions/2769467/what-is-dispatcher-servlet-in-spring>

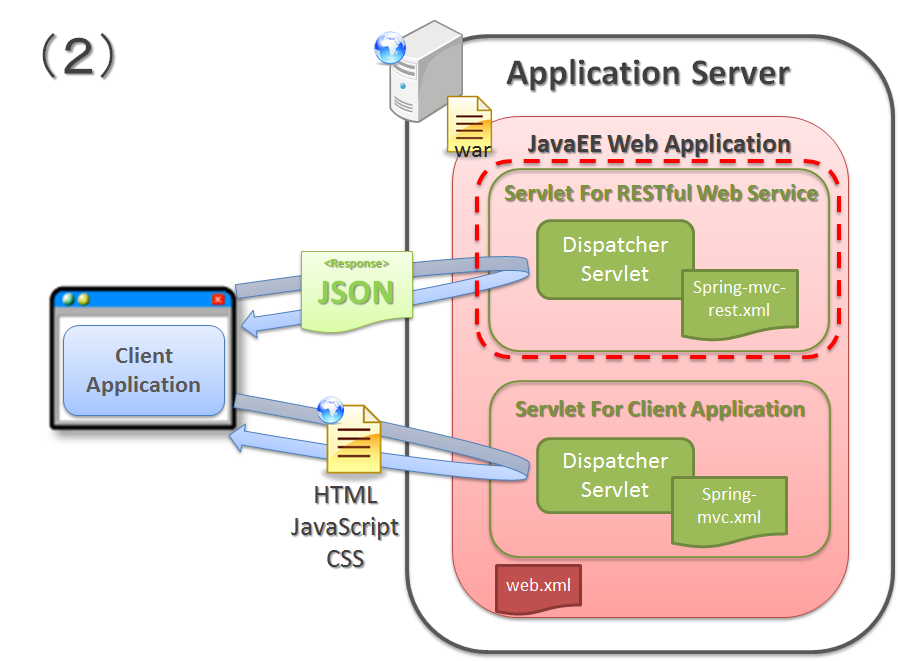








Mvc vs rest



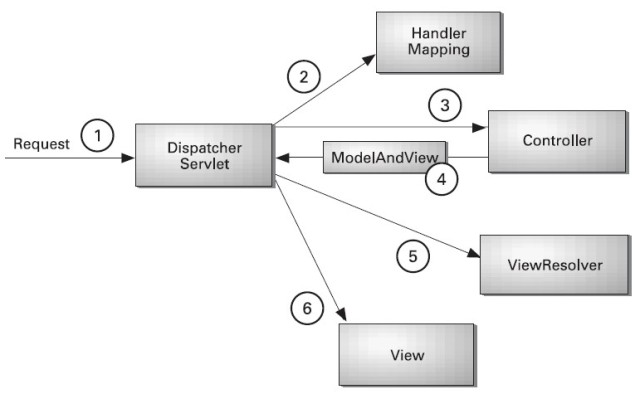
Mvc



<https://www.javatpoint.com/server-web-vs-application>







Annotation vs XML

he Servlet 3.0 specification is quite clear about that, have a look at chapter *8: Annotations and pluggability*, there is nice example at the end of section *8.2.3 Assembling the descriptor from web.xml, web-fragment.xml and annotations*.   
  
When both configs are configuring the same Servlet class under the same name, the web.xml takes precedence. However, the init-parameters are additive unless they have the same name, then, the descriptor takes precedence again (see 8.2.3.n.iii). So, if you had a different init parameter specified in your web(-fragment).xml, you'd see them both.

|  |  |
| --- | --- |
| up vote5down vote | [**Annotation**](https://blogs.oracle.com/swchan/entry/servlet_3_0_annotations) represents the metadata. If you use annotation, deployment descriptor (web.xml file) is not required. But you should have tomcat7 as it will not run in the previous versions of tomcat. @WebServlet annotation is used to map the servlet with the specified name.  @WebServlet("/Simple")  public class Simple extends HttpServlet {  private static final long serialVersionUID = 1L;  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.setContentType("text/html");  PrintWriter out=response.getWriter();  out.print("<html><body>");  out.print("<h3>Hello Servlet</h3>");  out.print("</body></html>");  }  } |

<https://blogs.oracle.com/enterprisetechtips/using-annotations-in-web-applications>

**ALL ABOUT Web.XML**

[**https://www.google.com/search?q=do+we+use+web.xml+when+we+use+annotation&rlz=1C1CHWA\_enUS760US760&oq=do+we+use+web.xml+when+we+use+annotation&aqs=chrome..69i57.20399j0j7&sourceid=chrome&ie=UTF-8**](https://www.google.com/search?q=do+we+use+web.xml+when+we+use+annotation&rlz=1C1CHWA_enUS760US760&oq=do+we+use+web.xml+when+we+use+annotation&aqs=chrome..69i57.20399j0j7&sourceid=chrome&ie=UTF-8)

**DISPATCHER SERVLET**

[**https://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/html/mvc.html#mvc-servlet**](https://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/html/mvc.html#mvc-servlet)

JAVA BEAN vs SPRING BEAN

**Java Beans:** At a basic level, JavaBeans are simply Java classes which adhere to certain coding conventions. For example, classes that

* Have a public default (no argument) constructor
* allows access to properties using accessor (getter and setter) methods
* Implement java.io.Serializable

**Spring Beans:** A Spring bean is basically an object managed by Spring. More specifically, it is an object that is instantiated, configured and otherwise "**managed by a Spring Framework container**". Spring beans are defined in a Spring configuration file (or, more recently, by using annotations), instantiated by the Spring container, and then injected into your application.

Note that Spring Beans need not to be Java Beans always. Spring Beans may not implement **java.io.Serializable** Interface, **can have arguments in constructor** etc.

This is the very basic difference between Java Beans and Spring Beans.

For More Information go through <http://www.shaunabram.com/beans-vs-pojos/>

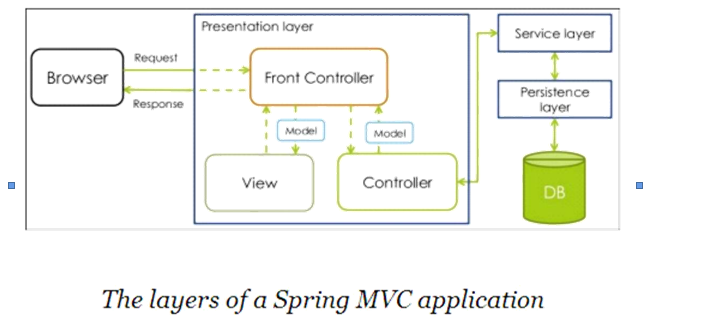
<https://stackoverflow.com/questions/21866571/difference-between-javabean-and-spring-bean>

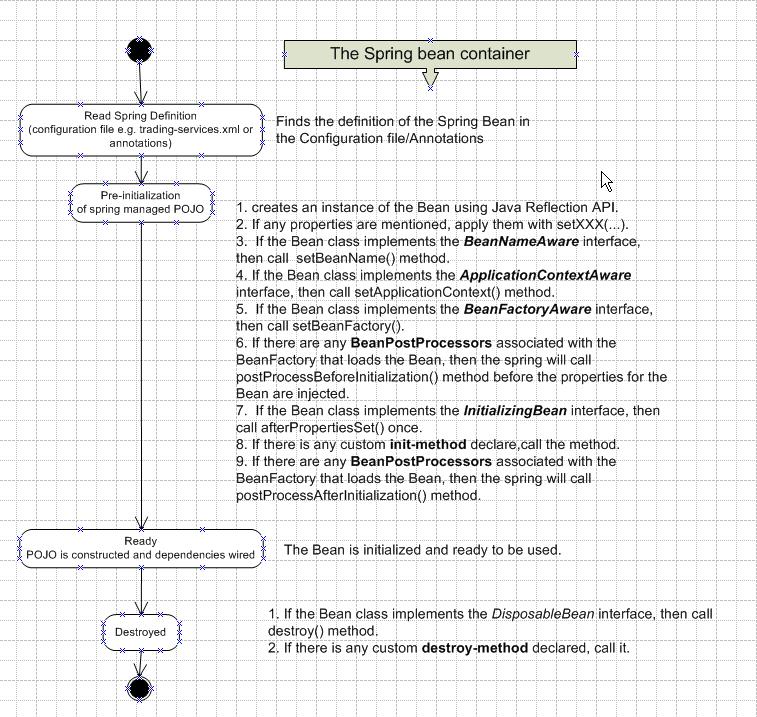
<https://stackoverflow.com/questions/12517905/what-is-java-pojo-class-java-bean-normal-class>

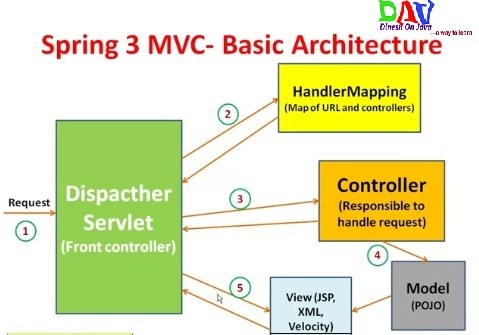
<https://medium.com/@sampadasharma/javabeans-vs-spring-beans-vs-pojos-58f789e8695c>

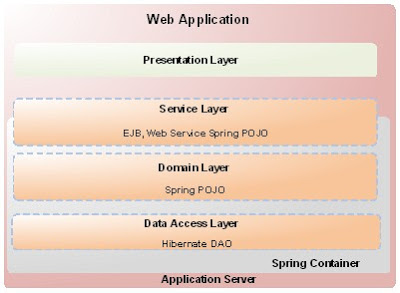


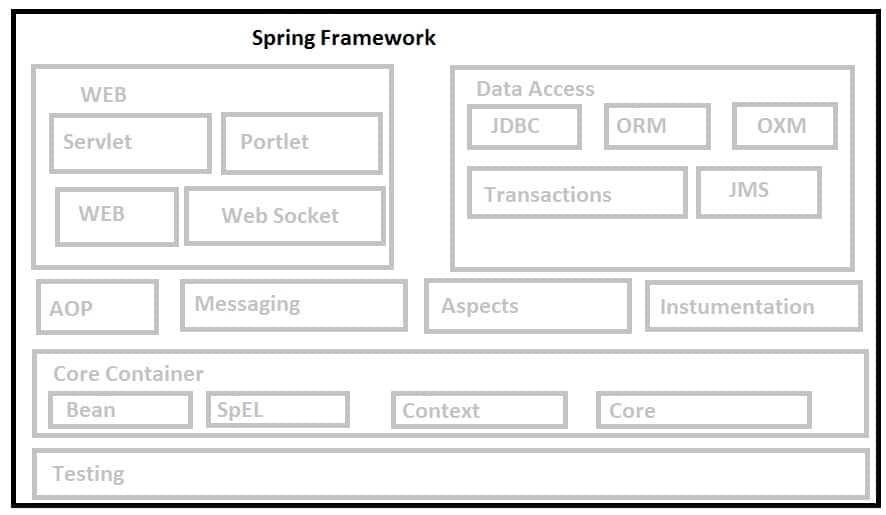
<https://www.linkedin.com/pulse/spring-framework-mvc-architecture-rosie-hong-lee>









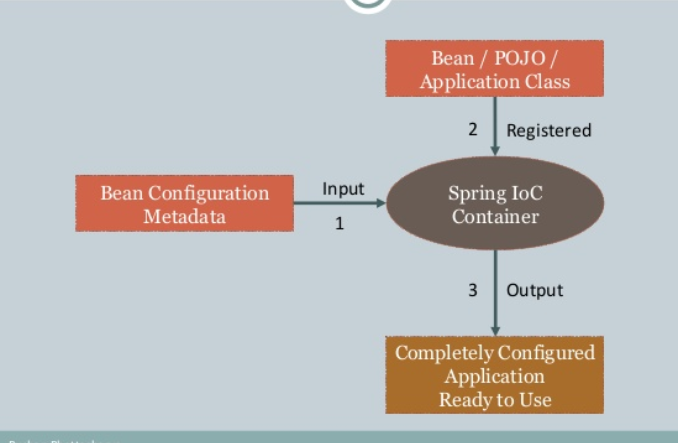


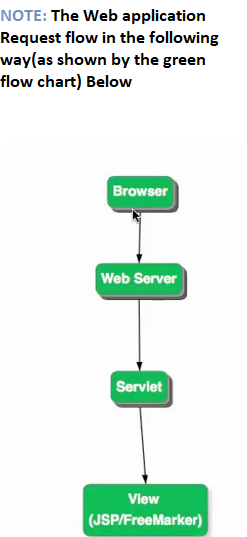
<https://www.google.com/search?rlz=1C1CHWA_enUS760US760&ei=iR-DWvSMD6aG5wKF9YaoCg&q=+where+is+is+java+pojo+or+bean+in+spring+framework&oq=+where+is+is+java+pojo+or+bean+in+spring+framework&gs_l=psy-ab.3...16246.39278.0.39934.29.29.0.0.0.0.116.2418.25j4.29.0....0...1c.1.64.psy-ab..0.16.1320...35i39k1j33i160k1j33i21k1.0.Zwj4Pm14_GQ>

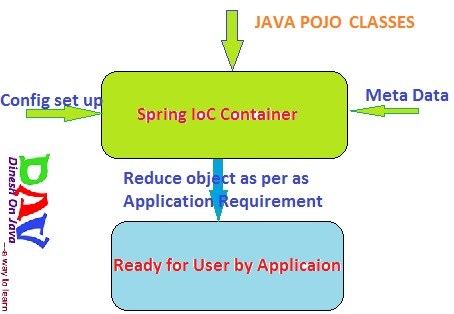
**NORMAL SPRING VS Spring rest**

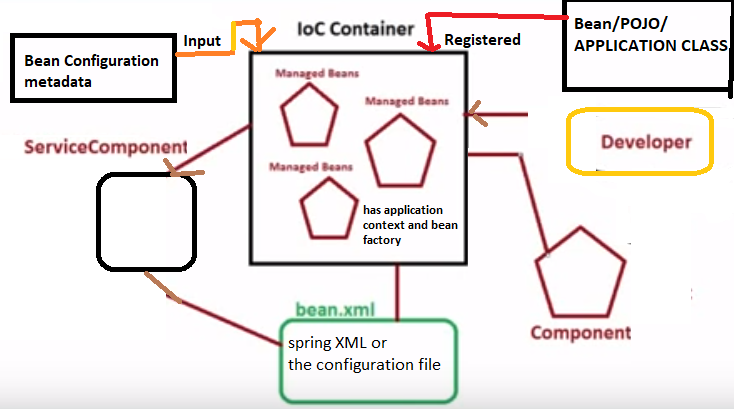
<https://www.quora.com/What-is-the-difference-between-normal-Spring-MVC-and-Spring-RESTful-web-service>

**Spring restVs MVc Vs boots**

****

****



****

**Coupling of General class /service component that has some business logic going on and the component that was developed by the developer on the right hand side has to be reduced (decoupling), and spring IOC helps with that. When the developer component needs the functionality of the service component the container will read about the need in the configuration file, Also by doing so it will manage the life cycle of the service component. Configuration file like the deployment descriptor in the case of servlets. Once reads the configuration file, it will be able to produce the object based on the service component and return it back to the client(developer) for the component. The object that are in the IOC which are instantiated after reading the configuration is called “ managed bean”. Later either by using name or by using class type, you can invoke the objects from the container. So rather than coupling and getting the object directly from the service component, we will use a look up method and either by using name or by using class type, you can invoke the objects from the container and that object will be served by the container for use in the component that is developed by you(the developer). So following factory, the bean (object )is tailor made for use.**

