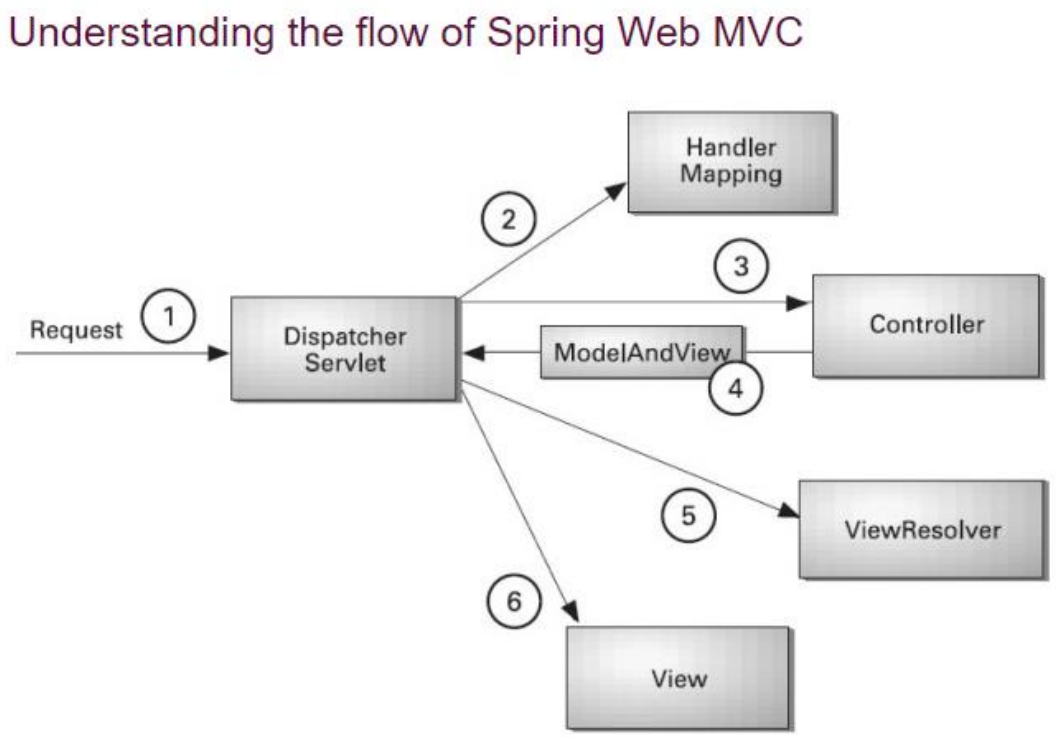


Spring MVC 3.2 Execution Flow ??

Step 1: First request will be received by DispatcherServlet

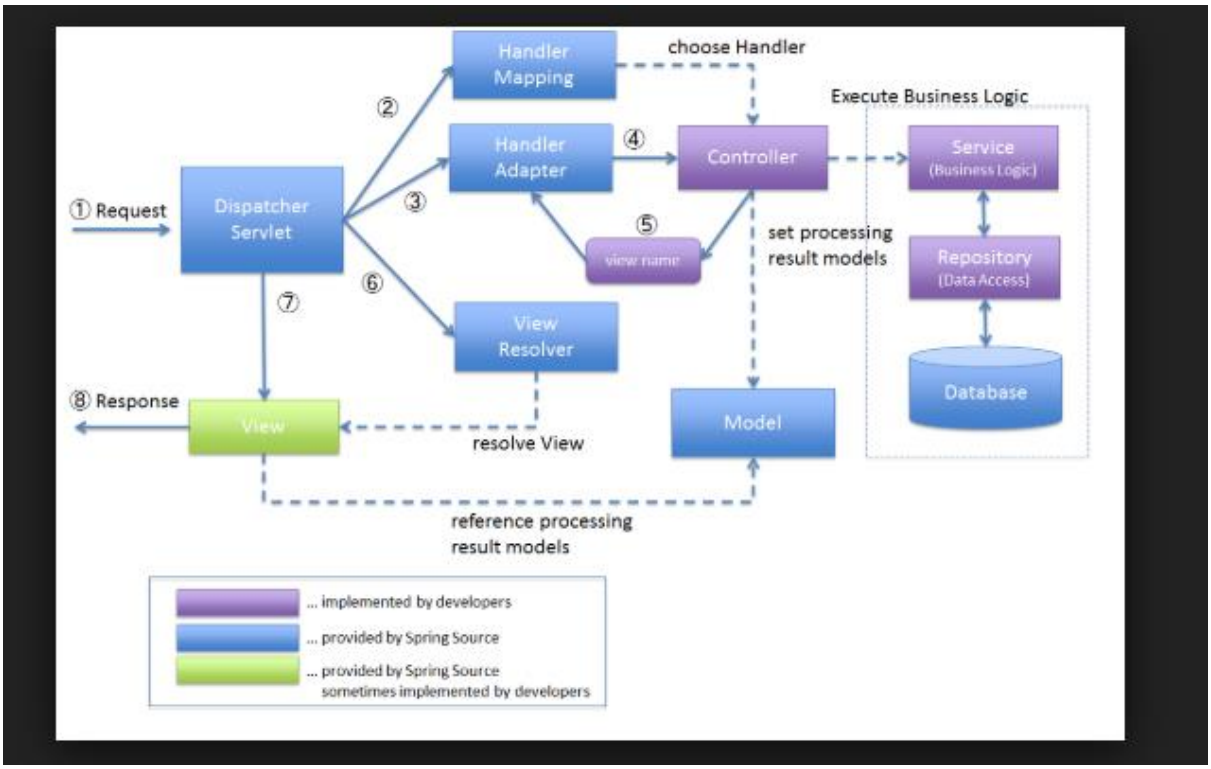
Step 2: DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request

Step 3: So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModeAndView object (contains Model data and View name) back to the DispatcherServlet

Step 4: Now DispatcherServlet send the model object to the ViewResolver to get the actual view page

Step 5: Finally DispatcherServlet will pass the Model object to the View page to display the result.

Spring Application Project Flow ? (Most Important Basic interview Question)



Advantages of hibernate over JDBC ?

- Hibernate supports Inheritance, Associations, Collections
- In hibernate if we save the derived class object, then its base class object will also be stored into the database, it means hibernate supporting inheritance
- Hibernate supports relationships like One-To-Many,One-To-One, Many-To-Many-to-Many, Many-To-One
- This will also supports collections like List,Set,Map (Only new collections)
- In jdbc all exceptions are checked exceptions, so we must write code in try, catch and throws, but in hibernate we only have Un-checked exceptions, so no need to write try, catch, or no need to write throws. Actually in hibernate we have the translator which converts checked to Un-checked ?
- Hibernate has capability to generate primary keys automatically while we are storing the records into database
- Hibernate has its own query language, i.e hibernate query language which is database independent
- So if we change the database, then also our application will works as HQL is database independent
- HQL contains database independent commands
- While we are inserting any record, if we don't have any particular table in the database, JDBC will rises an error like "View not exist", and throws exception, but in case of hibernate, if it not found any table in the database this will create the table for us ?

- Hibernate supports caching mechanism by this, the number of round trips between an application and the database will be reduced, by using this caching technique an application performance will be increased automatically.
- Hibernate supports annotations, apart from XML
- Hibernate provided Dialect classes, so we no need to write sql queries in hibernate, instead we use the methods provided by that API.

Spring framework ?

- Spring is a light weight and open source framework created by Rod Johnson in 2003. Spring is a complete and a modular framework, i mean spring framework can be used for all layer implementations for a real time application or spring can be used for the development of particular layer of a real time application unlike struts [only for front end related] and hibernate [only for database related], but with spring we can develop all layers
- Spring framework is said to be a non-invasive means it doesn't force a programmer to extend or implement their class from any predefined class or interface given by Spring API, in struts we used to extend Action Class right that's why struts is said to be invasive
- In case of struts framework, it will forces the programmer that, the programmer class must extend from the base class provided by struts API
- Spring is light weight framework because of its POJO model
- Spring Framework made J2EE application development little easier, by introducing POJO model

Spring having this much of demand because of the following 3 reasons….

- Simplicity
- Testability
- Loose Coupling

Spring BOOT Advantages over Spring ?

Spring Boot is a framework developed on top of core spring framework. The main aim of Spring Boot is to let developers to create spring production grade applications and services with very less effort.

what it takes to create real-time spring applications?

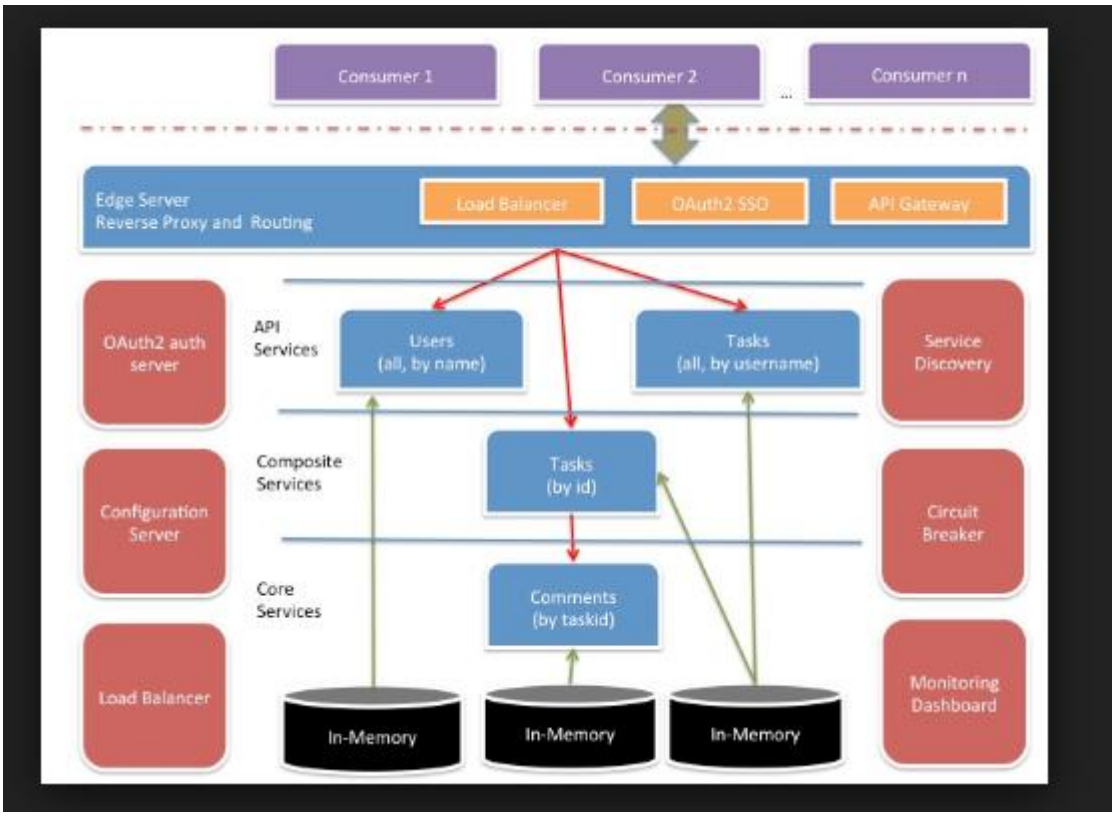
It includes writing many XML configurations, server setting, adding dependencies…etc. But with spring Boot we can avoid all these boilerplate code, writing XML configurations and annotations. We can create a real-time production ready applications with in minutes.

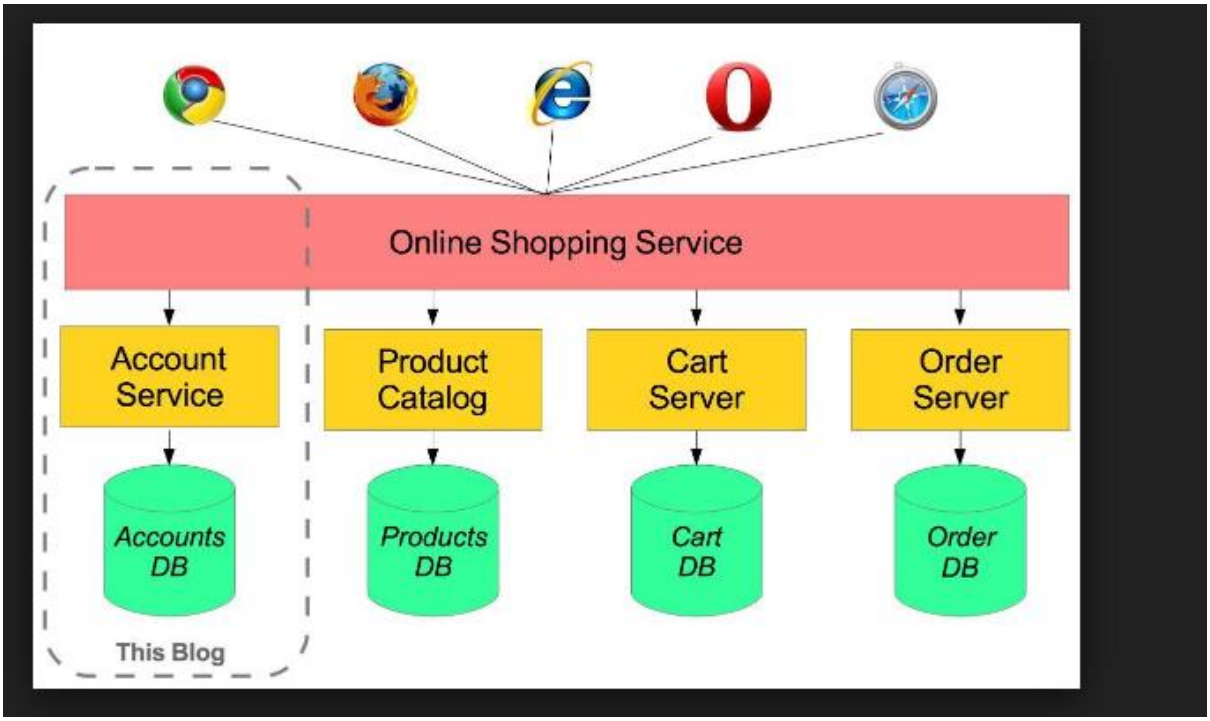
Spring Boot comes with inbuilt server, we no longer have to use any external servers like Tomcat, Glass-fish or anything else, so don't need to deploy WAR files

So, as I said main advantage of Spring Boot is, we can create spring based applications easily in very less time, without need of any XML configurations. The main disadvantage is, it will be little tough to migrate existing spring enterprise applications to Spring Boot.

Remember, we have to use either Maven or Gradle build tool to work with Spring Boot.

Spring BOOT Micro services Architecture ? **(For Basic idea)**





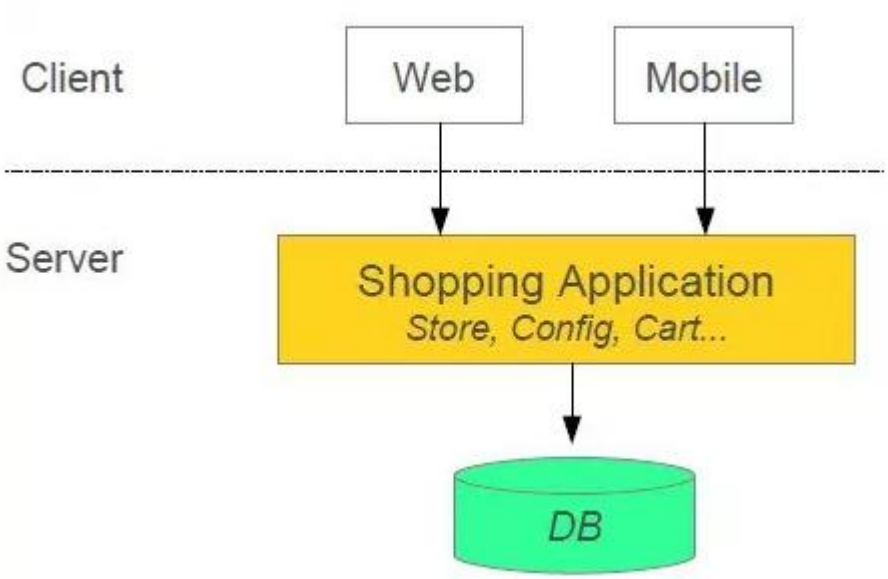
What is Microservices Architecture?

Microservices architecture allows to avoid monolith application for large system. It provide loose coupling between collaborating processes which running independently in different environments with tight cohesion. So lets discuss it by an example as below.

For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts. So these components are inevitably important for such a large online shopping portal. For online shopping system we could use following architectures.

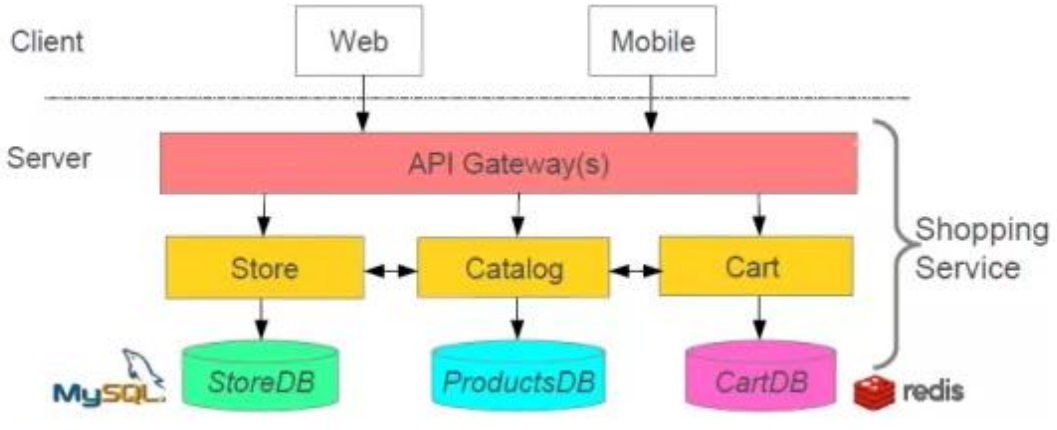
Shopping system without Microservices (Monolith architecture)

In this architecture we are using Monolith architecture i.e. all collaborating components combine all in one application.



Shopping system with Microservices ?

In this architecture style the main application divided in a set of sub applications called microservices. One large Application divided into multiple collaborating processes as below.



Microservices Benefits

Smaller code base is easy to maintain.

Easy to scale as individual component.

Technology diversity i.e. we can mix libraries, databases, frameworks etc.

Fault isolation i.e. a process failure should not bring whole system down.

Better support for smaller and parallel team.

Independent deployment

Deployment time reduce