Spring

Framework is an api, it provides its own api for developer applications. built in tool to help programmers to manage application structure more convenient for developing .

What is the spring framework?(lightweight : easy to use somewhere else)

Spring is a framework that comes up on top of java 2 WEE with its own API and it provides you boilerplate logic so that code redundancy will not be there in web applications or standalone and provide an IOC container for a lossy couple of our classes.

Easy Integration with third party api.

Help for programming to finish tasks within the timeline more efficiently.

Spring is a framework developed on Java EE that can be used to develop standalone or web applications.

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Spring provides different features, don’t have to write duplicated code, module programming, or a lossy couple between classes.

manager life cycle of the bean.

Non-invasive, less class, easy integration

it contains with different tool, api,

waste time for third party libraries.

Java EE core standard core concept we implemented, exists library,

J2wee for web application api, servlet, we need to develop our own api, deploy come into.

J2wee api disadvantages: class provided by j2wee is complex, not easy to understand, learn crvue, not easy to learn. write our own logic, sometimes we don't have to write code again, that's why frameworks provide code redundancy.

Framework is not replaced with j2wee, framework provides boilerplate logic, 3rd party integration.

What are the advantages of spring framework?

If you use spring framework you can develop any kind of application whether it is standalone application or web application, easy integration with 3 party API, easy isolated framework from your application, so you want to move to another framework, for spring framework it will not restrict us to implement its own class. support module programming, make you a class lossy couple. no tiely couple,

**What is the purpose of the spring framework?**

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Mainly loss couple.

What is a module in spring?

Module is an API specifically designed for particular requirements.

core module.// is a module which we use before we need to know about core modules, this is the same as core java concept.(Mandatory module everyone needs to learn).

mvc module.//model,view.controller.

JDBC module.

ORM module.

AOP module.aspect oriented programming/ isolated primary business with second business logic.

spring security.

Spring Core

Pojo class. plain old java object. class compile without additional 3 party api.compiler, not add any jar into build path.

1.all the properties must be private.

2.setter and getter method id pojo class

3. default constructor must be in pojo

4. class must be able implement a serializable interface.

poji interface.

interface not coming from 3 party api

strategy design pattern

Spring Core module

The Core Container in the Spring architecture contains the Core, Beans, Context, and Expression Language.

* The Core provides features to fundamental parts of the framework. This includes the Ioc as well the Dependency Injection (DI)

BeanFactory

ApplicationContext

spring bean: all the instances of class managers by spring framework that are called spring bean. The life cycle is managed by the IOC container and is specific in <bean> tag Can we define spring beans??

spring bean,is one of the implementations of the spring framework. for ioc container reference.

spring bean configuration file.

contains all spring bean configuration classes that need to be taken by the spring IOC container.

IOC container.(spring IOC, or spring container) Inversion of control

IOC container is a container that loads configuration files into it, where I have defined all the beans referenced.

is responsible for managing the life cycle of spring beans for a particular class.

we can get passiblity bean objects from IOC containers. read metadata from the configuration file.

What are different ways to implement spring containers?

1. Bean factory is one of the ioc container references,provided by spring framework.
2. Application context.

<bean>

</bean>

BeanFactory factory=new xmlBeanFactor(new class path(xxx.xml));//file location.

read configuration from xml for bean, this object will create ioc container, file loading to IOC, metadata properties.

beanfactory have references of IOC, IOC have metadata of beans.

assocaitation v inheritance

inheritance: sometimes don't want to inherit properties from the parent class.

Multiple inheritance not supported.

association: if the dependency class name changes then the dependent class gets affected.

change the name of the Demo

This is a tiey couple.

we have to use Strategy design pattern

Spring asks programming which class you want he will create for us, we only need to provide id for the instance, spring will look into it for us.

Temp{//depenedent class

Demo demo = new Demo();//depdedency

demo.show();

demo.display();

}

We don't have to worry about creating an object, someone will provide an object for me, my class will be a lossy couple. Responsible shift programmer to spring framework.

Strategy design pattern for lossy couples(a class behavior or its algorithm can be changed at run time.)

1. alway perform composition(association) over inheritance.
2. always go to the interface, and never go to implementation. poji interface.

demo class able to implement demo interface.

1. code should be open for extensions and close for modification.

Spring Core

---------------------------------------------------------------------------------------------------------------------

Dependency injection.

injecting dependency into the class, known as the dependency injection. provided by spring framework, make the class a lossy couple.

if we have two class demo and temp

in demo we use temp

Demo{

Temp temp;

}

if we want to inject dependency into demo, will inject dependency into temp.

What is the difference between dependency injection and autowire?

dependency injection is all about injecting dependency into dependent classes. We can do it in 2 ways. manually wrap bean to another bean dependency, we can use ref attribute to refer to<property >or <constructor> tage.

in autowire we don’t need to wrap dependency, it will be automatically done by autowiring.

Premitive injection:

we are not using any 3rd class, we do not make any relationship between classes, just inject value into primitive type value, not even use dependency.

premitve injection: predefind injection.

Setter injection:property inject:

property tag where we want to inject value into property, only for the setter method only.

<bean id="employee" class="Entity.Employee">

<property name="id" value="apolis056" />

<property name="name" value="Tom" />

<property name="salary" value="60000" />

</bean>

Constructor Injection

all about injecting dependency using constructor. if there are 2 beans, priority for the autowire candidate is not false;

<constructor-arg name=”id” value=”2”/>

<constructor-arg name=”name” value=”XYZ”/>

<constructor-arg name=”salary” value=”6000”/>

Collection Injection:all about injecting dependency for collection objects.

Collections dependency injections.

Deparement{

private List<String> departements;

}

in xml file, using

<list value=”java.lang.string”>

<value>Date Science</value>

<value>Computer Science</value>

</list> tag

Set

Deparement{

private Set<String> departements;

}

in xml file, using

<set value=”java.lang.string”>

<value>Date Science</value>

<value>Computer Science</value>

</set> tag

Properties

Deparement{

private Set<String> departements;

}

in xml file, using

<props >

<prop key=”1”>Date Science</prop>

<prop key=”2”>Bio Science</prop>

<prop key=”3”>Computer Science</prop>

</props> tag

Map

Deparement{

private Set<String> departements;

}

in xml file, using

<props >

<Map>

<entry key=”1”, value=”data science”/>

<entry key=”2”, value=”Bio Science>Bio Science/>

<entry key=”3” , value=”Computer Science/>

</Map>

</props> tag

Autowire!

Spring framework will know which dependency will be injected, automatically injected for you. is all about injecting dependency into a dependent class by IOC container, we don’t need to do the wiring, we don’t need to refer the dependency to my dependent class. IOC container will recognize which dependency to inject will do wiring for you this is autowiring by achieving in 3 ways. by name, by Type, by constructor.

ioc container will automatically

by setter,by name(using setter method)

by constructor(using constructor)

byType(using setter method)

will call the setter method for dependency injection.

byname will check the name of the reference inside the employee class and check if there is any dependency with a match id with the same name as the reference class, if yes. IOC will create an object and inject dependency into it.

bytype it will match the type of the bean and inject the same type of bean into the created, there will conflict, object to avoid this we can disable for the bean autowire-candidate=”false”(this bean will not be consider for use); bean reminder in xml file, but will not work for autowire.

byconstrcutor

all about injecting dependency using constructor. if there are 2 beans, priority for the autowire candidate is not false; check which constructor is defined in the config file first, it will create an object using that one. if it has the same number of parameters the first constructor will be used. objects of that class will be created by the IOC container. if we have an overloading constructor in our class, whatever constructor with max parameters will be called.

1)will call the default constructor if we don't define our own constructor.

2)if bean is missing it will check second constructor

3)if the same amount of bean is there it will check to maxone.

Two type of bean

1. one bean is refer by multiply bean,
2. one bean only used by a particular bean(one bean is defined inside another bean called nesty bean). scale only for outer beans.

Inner Bean as know as nesty bean, beans that are defined within the scope of another bean

Bean aliasing

In my mind, bean aliasing can be helpful in large systems, where you can not manipulate bean names. You have the option to create your own name (alias) specific for your part of the system

Day2--------------------------------------------------------------------------------------------------------------------

Application context

Is one of the implementations of the IOC container.

Difference between application context and bean factory?

Both are implementation of IOC container,

Bean factory is an interface application context is an interface too.

application context is the child of the bean factory.

What does lazy loading mean?

Bean factory lazy loading the bean,BeanFactory only loads the bean when it is required. basically means it will create objects on demand.When we use the bean factory, it will load the file, but it will not instantiate or create a bean object for us.For example we need to call getBean() to instantiate a bean object for us.

.

application context is eager loading.

when we defined application context it will look into metadata,

1. Application context provides more features to the bean than the bean factory.
2. Application context supports “component scan tag” while bean factory does not.
3. Bean factory doesn’t support annotation, while application context supports annotation.
4. Bean factory has scale for the bean as singleton or prototype only, application context support more like singleton, prototype, requires,session, global session.

*ApplicationContext* is considered a heavy IOC container because its eager-loading strategy loads all the beans at startup. *BeanFactory* is lightweight by comparison and could be handy in memory-constrained systems.

Bean inheritance:

Bean inheritance is known as the inheritance of a bean into another bean. (using parent keyword), we get all properties from one bean to another bean. We defined the parent attribute in the bean tag of child bean to specify which bean is the parent of child bean.

Bean Scope:is used to define the scope for the same bean id(not working for different), we will have different scope, by default: scope for the bean is singleton. defined what kind of scope for the bean. scope specific to bean id. scope only discussed in the same bean.

scope=”singleton”,in singleton the same bean will be returned.

scope=”prototype”,in prototype the

collection merge property

using merge keyword

lookup method injection

Create an abstract class, and an abstract factory method responsible for giving me reference to another class interacting with the ioc container. then we have defined <lookup-method>

lookup method to inject dependency into dependent class, this will not using autowire we configure lookup method. alternative of normal setter dependency injection.

a factory method which returns a new bean on every call without us implementing the method.

another way to inject dependency into dependency class, another way fetch object directly from IOC container. if we don't use autowiring.

When to use the lookup method?

when we can’t do the autowiring.

When to use the lookup method: if the situation we can not define dependency injection

init, destory ,

init method: we can define the initialization of our bean or we can define a custom defined jdbc connection.

destroy method: before the ioc container shutdown, ioc will call the destroy method, perform a clean up process.

if we don’t define our own init and destroy the default method will be defined by spring.

initialization means initialized value.

instantiation means creating a class.

**Bean life cycle**

create initialization service drstory

Bean life cycle is all about having different phases for the bean, these phases are managed by the life cycle of the bean, first of all, the spring container gets started, first, instantiated object of the bean. This will call the setter method for initialization and dependency injection, then call the preProcess() method for any operation we want to do before the init method, next call init() method then after init() method postprocess() method being called. Finally, before the termination of the bean we will call destroy() method. After the destroy() method finished, spring containers are closed.

inti(),destroy() method is defined specific bean for inti operation and destroy

Day3--------------------------------------------------------------------------------------------------------------------

Annontation

Annotations are used to provide supplement information about a program

* Compiler instructions
* Build-time instructions
* Runtime instructions

Annotation stereotype

stereotype annotations are used to define any bean using an annotation approach.

context:annotation-config used to enable all time of annotation including stereotype of annotation.

@Component,

Component is class level annotation.

is an annotation that allows Spring to automatically detect our custom beans and Instantiate them and inject any specified dependencies into them.

@Configuration,

Component is class level annotation,makes normal class as configuration file.

can be used by the Spring IoC container as a source of bean definitions

@ComponentScan,

ComponentScan this annotation enables component scanning in Spring for providing package name.

The @ComponentScan's basePackages attribute specifies which packages should be scanned for decorated beans.The @ComponentScan annotation is an alternative to <context:component-scan> XML tag.

@Value,(property level)assign value of property

@Autowire,

Autowire is an annotation that we can implement at method level, property level or abetory method level. it automatically does the wiring for us. Default byType automatically does wire for us and dependency injection.if we do autowiring on property we don't need a setter method.

You can use @Autowired annotation on setter methods to get rid of the <property> element in XML configuration file. When Spring finds an @Autowired annotation used with setter methods, it tries to perform byType autowiring on the method.

@Scope,The scope of a bean defines the life cycle and visibility of that bean in the contexts we use it, we can define many type of scope for example:

* singleton
* prototype
* request
* session
* application
* websocket

@Qualifier,

In the qualifier we defined the name of the class. Help us to fix the conflict if we have multiple implementations for the bean. To tell which bean needs to be injected. If there is a parent reference, multiple children are implemented.

OR

When you create more than one bean of the same type and want to wire only one of them with a property you can use the @Qualifier annotation along with @Autowired to remove the ambiguity by specifying which exact bean should be wired

@Required,

is used on the setter method level, is required for defining specific variables(any type), and this is mandatory. This variable must be defined at configuration time. Required is doing some kind of check to make sure this variable is not the default value.

is used on the setter method level, is required for defining specific variables(any type), and this is mandatory. This variable must be defined before instantiation of the bean and at the time of the initialization of the bean. Required is doing some kind of check to make sure this variable is not the default value.

@Bean

@Bean is used when annotating factory methods in configuration files using @Bean annotation. Configuration class will load by IOC container, IOC container will look into configuration file, read configuration file and see there is Bean. When IOC sees Bean, annotation will take whatever the object is returned by factory method, make that object as value, and method name will be the bean Id. Also known as, method name as key and return value as value.

key: hcl value:hcl object

We do autowire for hcl beans, going to map and see which object is there.

Bean is an instance of class life cycle managed by IOC container,

3 ways

bean tag<Bean></Bean>

configuaration

@configuration

factory method inside configuration class.

stereotype annotation

@service,

@Repository

@Conponemt

@Controller

Component Annoation

*@Component* is an annotation that allows Spring to automatically detect our custom beans.

In other words, without having to write any explicit code, Spring will:

* Scan our application for classes annotated with *@Component*
* Instantiate them and inject any specified dependencies into them
* Inject them wherever needed

tag-context:component-scan base-package(we don’t need to write this if we are using @Component)/using tag-context:component-scan base-package=”com.apolis” base package will scan all components for our class.

@Configruation now we don’t use configuration annotation which using **new AnnotationConfigApplicationContext**(com.apolis.common.AppConfig.class);

tag-context:component-scan== **AnnotationConfigApplicationContext**

POC on fully annotational approach

@Autowrie can be applied to different levels within class.

@Qualifier has to fix the conflict when we have multiple bean implementations of the same bean. setter level method, or property

@required will not allow you to create an object without initializing the required value.Apply on setter method.

Dependency check: check any dependency before any initialization, make sure dependency is valid before injection. Same as @required

1:08:00

@Bean annotation factory method in config file, return that object put it inside IOC container as bean object, method name becomes if for whatever object we return out.

How to define 2 different beans?

If we define 2 beans, one is a configuration class file and an IOC container.

Spring JDBC

SpringJDBC:

Why do we need SPring JDBC?

spring jdbc develops on its own api provided by spring, it develops one top jdbc api. It provides a more simplified way for persistence data.

Driver

Driver will enable your application to work for a specific kind of database.

DriverManager class:

This is the interface between the user and the driver. DriverManager acts like a watch on the driver or manage on driver, for connection between the database and the driver. which is able to manage which database to connect.

Registor our driver

Install driver, load driver, class forname load you driver into application.

DataSource creates connections with databases and our applications and uses them.

Difference between driver manager and datasource?

If we use driver managers application performance gets reduced using driver manager, compared to data source improves application performance, it will not create connection using class, it uses application server. Application server created at runtime, will not create connection statically like driver manager does.

Driver creates a connected static datasource using application server, application server.

Sometimes we want to create many Driver Managers, and there is a limitation of the amount of concurrent connections which we can create. Data Source allowed us to create N amount of connections.

In Java, The PreparedStatement interface is a subinterface of Statement. It is mainly used for parameterized queries, not hard coding the queries. Also able to create queries with runtime arguments.

Maven:pom.xml(Project Object Model)configuration details used by Maven to build the project

Disadvantage of Java JDBC

Problem for JDBC api

If we are using JDBC api first problem is

1.code redundency

second problem is

2.everytime we define a connection this code might throw an exception, everytime we have to define try and catch, unnecessary use.

3.eveytime we have to create an object for each result set row according to their object type unnecessary process the result set. put more effort into the result set.

4.JDBC does not support Transaction management.

Spring JDBC solves Java JDBC problem by providing Spring JDBC template

* JDBCtemplate provides us with some methods, we just execute the method for our queries, so we don’t have to worry about opening connections and closing connections.
* We don't have to handle exceptions in the JDBC template.
* provide transaction management
* Support for result set
* Support Name parameter and position parameter.

Spring JDBC only works with dataSource, not driver manager.

[Spring JdbcTemplate Tutorial - javatpoint](https://www.javatpoint.com/spring-JdbcTemplate-tutorial)

Day4

Spring AOP-----------------------------------------------------------------------------------------------------------------

@Repository

Repository is a class directly interacting with a database, getting data from the database and persistence data from the database. Class has persistence logic in it, this class is known as the persistence class.

name parameter using :

Compared to position parameters, easy to understand which position to be pass

For name parameter are specified by assigning values to their names

class is a template class with a basic set of JDBC operations, allowing the use of named parameters rather than traditional '?' placeholders.

using the position parameter ?

when we use position parameters while creating the queries. After passing the parameter to the update method, we have to check the sequence of parameters. we passed an object array for position parameters to make sure order matters for our array.

Cyclic(circler) dependency

Circular dependency happens when a bean A depends on another bean B, and the bean B depends on the bean A. or more Bean A → Bean B → Bean C → Bean D → Bean E → Bean A.

It can happen in Spring when using constructor injection; if you use other types of injections this not a problem, since the dependencies will be injected when they are needed and not on the context loading

For this problem we need to use setter injection instead of constructor injection.

AOP

Aspect Oriented Programming.

To separate primary business logic and secondary business logic.

AOP is a concept related to the business layer. 2 types of business logic, primary business logic and second business logic. Primary business is main logic where the functionality works, secondary business logic performs support of primary business logic.

If we write primary and secondary business logic together there will be issues with code duplicity, maintainability.

2:30:00

Aspect is: generlize cross cutting functionality

Advice is: implementation of cross cutting functionality.

JoinPoint: is a point at which cross cutting functionality is combined with primary business logic.

PointCut: is a collection of the joinPoint, using regular expression that matches a joinpoint. Each time any join point matches a pointcut, a specified advice associated with that pointcut is executed. to defend the joint point.

Adviser: Group ‘Advice’ and ‘Pointcut’ into a single unit, and pass it to a proxy factory object

Target(method): contains primary business logic.

Proxy class :When primary business logic and cross cutting logic combined together, and one proxy class will be create

Proxy: Before creating a proxy class, aop will use the object of the target class. but after creating a proxy class aop will use a proxy object class.

Weaving: process converts target class to proxy class, this kind of process can be done at any point of time compile, runtime, or at the time loading the class.

Type of advice

Before advice(before primary business logic) any advice which secondary business logic wants to execute before your target method(also known as primary business logic).

After advice/Finally advice: execute after primary business logic. Finally advice will be executed no matter if there is an exception or not.

Around advice:secondary business logic will be executed before your primary business and after primary business logic.

After returning advice:execute only target method execute successfully. if there is an exception after returning, advice will not be executed.

After throwing/throws advice: if execute primary business logic as known as target method getting throw exception then after throwing will return. If no exception after throwing will not return.

AOP annotation:

@Aspect: class level annotation, place at top of advice class. If apply Aspect annotation to any class, which means that class contains some cross cutting logic or advice method. This class will be enabled as an aspect inside of your application. AOP finds this annotation and starts to use it as an aspect. contains annotation before, after, afterreturn

Is annotation for cross cutting logic

or Mark a class as a class containing advice methods.

A class in which we put all aop related configurations that class known ass aspect class.that class should be annotated with @aspect

Aspect Oriented programming.

@Before:method level, advice before, this advice method will execute before target method.

@After: method level, this method is executed after primary business logic(or target method).

@AfterReturning:apply one cross cutting logic or on advice method level, only executed when primary business logic or target method executed successfully.if there is an exception after returning advice will not be executed.

@AfterThrowing: if execute target method getting throw exception then after throwing will return. If no exception after throwing will not return.

@Around: This advice executed before primary business logic and after primary business logic.

@Pointcut:PointCut is a set of one or more JoinPoint where an advice should be executed

using to create a pointcut signature. example: (“customerPointcout()”)

* @PointCut("execution(\* com.tutorialspoint.\*.\*(..))")
* @PointCut("execution(\* com.tutorialspoint.Student.getName(..))")

3 cross cutting functionality:

logging, security, transaction.

This cutting method is executed before my primary business logic.

Day5--------------------------------------------------------------------------------------------------------------------

Srping ORM

ORM stands for object relational mapping

Spring ORM(all about web application)

59:00 facilitate

Spring ORM is a model or api which is developed on top of ORM tools like hibernate,JPA(java persistence api), Java data object (JDO). Spring ORM facilitates integration concepts, and is all about integration of spring ORM into 3rd party tools. Also it is a technique for converting data between relational databases and object oriented programming languages

ORM tool: transfer objects from one layer to another layer. map model instance variable of the class map to column of the database and class name as table name. to get objects from the database instead of the result set.

hibernate configuration file

hibernate mapping configuration file

@Entity annotation. We must specify this annotation at the class level. @Entity annotation use for POJOs representing data that can be persisted to the database. An entity represents a table stored in a database. Every instance of an entity represents a row in the table.

@Table mapping the current class of the object with database table name.

@Id mapping the current instance variable as id for primary key in the database.

@Column mapping the current instance variable to column in our database.

Hibernate ORM (or simply Hibernate) is an [object–relational mapping](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping) tool for the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language. It provides a [framework](https://en.wikipedia.org/wiki/Software_framework) for mapping an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) domain model to a [relational database](https://en.wikipedia.org/wiki/Relational_database)

What are the advantages of hibernate?

* Hibernate provides transparent persistence for [Plain Old Java Objects](https://en.wikipedia.org/wiki/Plain_Old_Java_Object)
* Provide Hibernate Query Language(HQL)
* Hibernate can be used both in standalone [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) applications and in [Java EE](https://en.wikipedia.org/wiki/Java_EE) applications using [servlets](https://en.wikipedia.org/wiki/Java_Servlet)

What is a hibernate template?

P2:48:00

provide you with predefined boilerplate logic that automatically handles the getting the connection or to opening the session, opening the transaction, closing transaction for us. This is the way to avoid code redundancy in our application.

What is the advantage of spring ORM api over normal api or hibernate api?

1) If we use hibernate directly, code duplication will be one of the issues. Spring provides you a lot of template tools for programmers to use so duplicated code will not be there.

2) For whatever ORM tools we use, the program will throw different kinds of exceptions. However, spring ORM provides the same exception for all exceptions. It provides a common entry point for all exceptions.

Day6--------------------------------------------------------------------------------------------------------------------

1) What is MVC?

The MVC (Model-View-Controller) is a software architectural design pattern. It separates the functionality of an application into three interconnected parts - Model, View, and Controller.

Sring MVC

A Spring MVC is a Java Framework which is used to develop dynamic web applications. It implements all the basic features of a core spring framework like Inversion of Control and

Dependency Injection.

What are the advantages of Spring MVC Framework?

* Separate roles - The Spring MVC separates the application into three interconnected layers where each layer has its role.
* Light-weight - It uses light-weight servlet containers to develop and deploy your application.
* Powerful Configuration - It provides a robust configuration for both framework and application classes that includes easy referencing across contexts, such as from web controllers to business objects and validators.
* Rapid development - The Spring MVC facilitates fast and parallel development.
* Reusable business code - Instead of creating new objects, it allows us to use the existing business objects.
* Flexible Mapping - It provides the specific annotations that easily redirect the page.

**spring mvc Architecture pattern 3 layer**

1. some server layer(business layer/logic )
2. view layer(presentation layer)
3. model layer DEL layer. interacting with database

MVC pattern separates(ioslated) all the layers.

1. We have one more controller, which controls the flow of the application. Requests will come directly to servlet, servlet is the only class responsible for managing http requests.

Port number

Port number is a communication endpoint, for any kind of software. anytime when we create an application and want our application to communicate with the outer world, that application will be accessed through the port number. everytime if we want to access that server we have to mention the port number with ip address follow by port number

When we install any kind of application, any application comes with its default port number.

localhost represents the local machine ip address.

IP address.

macine unique id.

2Type of web application:

Static web application:There is no processing of content on the server (according to the user) in Static Websites. Web pages are returned by the server with no change therefore, static Websites are fast. There is no interaction with databases.

Dynamic web application: In Dynamic Websites, Web pages are returned by the server which are processed during runtime means they are not pre built web pages but they are built during runtime according to the user’s demand

If we want to run servlet program

server responsibility to execute servlet program, servlet will responsible to handle http request, server will interact with servlet it will execute servlet program which was java program, servlet take the input parameter, execute logic interacted with database and render data to the view.

Difference between design pattern and architecture pattern?

Design pattern: provide solution for design recursion problem.

Architecture pattern: related to architecture structure, does not improve application performance. (example: spring mvc, microservice)

model 1 complex code

model 2 or 3 complex code very less.

JSP

**Java Server Pages(basically to create view is a view page)**

JavaServer Pages (JSP) is a Java standard technology that enables you to write dynamic, data-driven pages for your Java web applications. We use html and css tags. multiple libraries support view pages. Inside of the JSP we have a jsp engine inside our server ,JSP is built on top of the [Java Servlet](https://www.javaworld.com/article/3313114/learn-java/what-is-a-java-servlet-request-handling-for-java-web-applications.html).

What is server?

Server is a software application which we need to install in the machine, after install server we call it a server machine. We have different types of server, application server or web server.(example tomcat).

Tomcat

1.Tomcat is one kind of server, it takes less time to boot or start, saving development time.

2.lightweight does not require a lot of resources.

3. We could directly use the zip file.

Servlet

servlet is a java class instantiated by the server that responds to a particular type of network request - most commonly an HTTP request. Basically, servlets are usually used to implement web applications.

Servlet container

is responsible for managing the life cycle of the servlet, it will instantiate the servlet, it will process the servlet. and at the end it will destroy the servlet. when a client sends some request to the server, and the server will not directly interact with the servlet, it will interact with the servlet container.

Servlet container: , they are responsible for accepting a request, processing it, and sending a response back., way of managing http requests and responses.

Dispatcher servlet:

1)main controller or front controller of spring mvc project.

2)This servlet can only handle http protocol requests, because it is a sub class of HTTP servlet.

3) To manage the flow of the spring mvc project, the dispatcher servlet needs a spring configuration file.

4)Dispatcher servlet starts the spring web application context container and reads the required bean from the spring container.

Component of web application

web.xml

in web xml file we want to define whatever dispatcher servlet we want to instantiated by

servlet container.

ServletContext class-load web.xml file read metadata of web.xml file, and provide all information to the servlet container.

MVC flow

first request come to server, server interacting with servlet container, then servlet containers instanits ServletContext class one time only and load metadata of web.xml, then servlet container read dispatcher servlet details from servlet context object, finally, containers will start interaction with the front controller.

first request come to server-> server interacting with servlet container->servlet containers instanits ServletContext class one time only and-> load metadata of web.xml-> servlet container read dispatcher servlet details from servlet context object.-> containers will start interaction with the front controller.

27:00

Handling mapping is a helper class of dispatcher servlet. help dispatcher servlet to identify appropriate controller beans for the given request. for each request dispatcher servlet will call a handler mapping class to identify associated controller beans. it will create an object of bean and return to dispatcher servlet.

Dispatcher servlet will connect to handle mapping class, handle mapping class tell dispatcher servlet for this request that is controller responsible to handle this request.

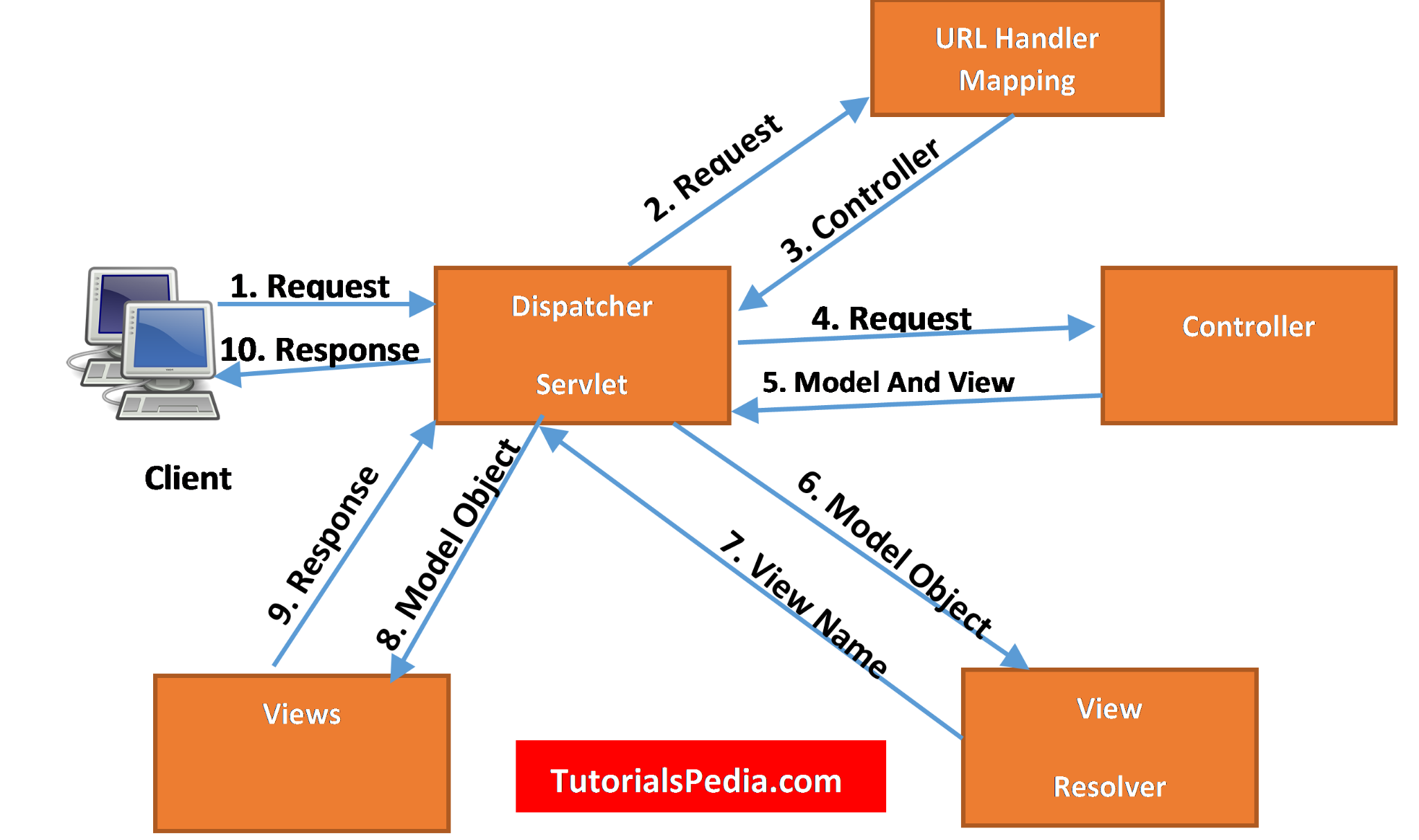
Bean name url handle mapping

*BeanNameUrlHandlerMapping* is the default *HandlerMapping* implementation. *BeanNameUrlHandlerMapping* maps request URLs to beans with the same name. or maps request URL to related bean name For example, an incoming URL *“/foo”* maps to a bean called *“/foo”*. An example of pattern mapping is mapping requests to *“/foo\*”* to beans with names starting with *“/foo”* like *“/foo2/”* or *“/fooOne/”*.

after mapping return corresponding bean object from ioc container given to dispatcher servlet.

**workFlow of spring web mvc** 39:00->new P1 18:00

let's say request come from home.html, it come to dispatcher servlet, dispatcher servlet will looking into ioc container, see what kind of handle mapping we have configure, if configure using bean name, it will call bean name class create it object call in constructor, then it will take related url and match with bean, if match it will get bean object from ioc container give to dispatcher servlet, now dispatcher servlet has object of bean, now dispatcher servlet can easy call that method by using reference of that class. that request is being executed.



Client sends the request to the dispatcher servlet, dispatcher servlet is the front controller(Dispatcher Servlet is a servlet specified in **Web.XML** file or in the Web Configuration class).

Dispatcher servlet go to spring bean configuration file and which URL handler mapping is configure(beanName, simpleUrl, defaultAnnotation), and identifies the appropriate controller bean,return this controller bean object to dispatcher servlet, using this controller bean object to call the request method of the controller bean. the method will execute, this method will return you view logic name or model data. view logic and model data will return to the dispatcher servlet to identify the location for the view located. then dispatcher servlet will take this model object into view resolver, which view resolver helps to return the complete of the url or view path, then it will wrap that details into view object and send back to dispatcher servlet. Finally the dispatcher servlet will render the view page and send it back to the client.

attribute servlet

kinds of controller,

An attribute in servlet is an object that can be set, get or removed by the following aspects,

**setAttribute().**

1. Request Scope
2. Application Scope
3. Session Scope

request come to dispatcher servlet and interacted with

dispatcher servlet wants to interact with IOC containers.

Web application context implementation of IOC container

51:00

View resolver class

View resolver is a class that has a dispatcher servlet to identify an appropriate view page out of multiple pages.

View resolver class

1)One way is to help us to identify where the view page inside of our project.

2)Second, the view resolver will match any match view with which we have returned from the controller class. After the match view, it will wrap details into the view object and send the return view object to the dispatcher servlet.

InternalResourceViewResolver

<bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name = "prefix" value = "/WEB-INF/jsp/"/>

<property name = "suffix" value = ".jsp"/>

</bean>

JSP tag we want to write some java code using jsp tag

spring form tag: <form:input path="name" />

BeanNameUrlHandlerMapping

*BeanNameUrlHandlerMapping* maps request URLs to beans with the same name. or maps request URL to related bean name after mapping return corresponding bean object from ioc container given to dispatcher servlet.

BeanNameUrlHandlerMapping is the default HandlerMapping implementation. BeanNameUrlHandlerMapping maps request URLs to beans with the same name.For example, an incoming URL “/foo” maps to a bean called “/foo”. An example of pattern mapping is mapping requests to “/foo\*” to beans with names starting with “/foo” like “/foo2/” or “/fooOne/”.

SimpleUrlHandleMapping

This simple url handler mapping will have one mapping of collection property with key and value pair, key will be defined as related url, for value we will define for controller bean. For each request we have to match each apporiore bean.after mapping return corresponding bean object from ioc container given to dispatcher servlet.

SimpleUrlHandlerMapping is the most flexible HandlerMapping implementation. It allows for direct mapping between either bean instances and URLs or between bean names and URLs

DefaultAnnotationHandlerMapping

In default annotation handler mapping, we don’ t use any configuration file, which we use annotation approach and component-scan for scan through the annotation in our file to match url with @RequestMapping value element.

Controller Class

We typically use @Controller in combination with a @RequestMapping annotation for request handling methods.

Request Handler Method

The HandlerAdapter is basically an interface which facilitates the handling of HTTP requests in a very flexible manner in Spring MVC.

It's used in conjunction with the HandlerMapping, which maps a method to a specific URL.

The DispatcherServlet then uses a HandlerAdapter to invoke this method. The servlet doesn't invoke the method directly

Steps to develop the SpringMVC application

create maven

config maven dependencies

model POJO

@Component is a generic stereotype for any Spring-managed component.

@Service annotates classes at the service layer.

@Repository annotates classes at the persistence layer, which will act as a database repository.

Controller

Presentation (View) layer

Command class

A command is an object whose role is to store all the information required for executing an action, including the method to call, the method arguments, and the object (known as the receiver) that implements the method.

@ModelAttribute

@ModelAttribute("command")

@ModelAttribute can be used either as a method parameter or at the method level.

When the annotation is used at the method level it indicates the purpose of that method is to add one or more model attributes to the injected particular object.@ModelAttribute methods are invoked before the controller methods annotated with @RequestMapping are invoked.

---------------------------------------------------------------------------------------------------------------------------