hibernate - database access layer

struts - web layer

FJBs - service like transactions, security and messaging layer

Log4j - logging

Problem:

applications that use these varied frameworks and services become difficult to maintain as it grows includiing testing.

Need :

-> applications using multiple frameworks and services must be simple and manageable

-> codes should be loosely coupled with frameworks so that testing and reuseability becomes easy

rod johnson rasied a question, why are we doing java like this.. and he introduced spring.

\*\*Spring helps java developers in agile development by providing proxy objects and eliminates dependencies aross project

Spring framework can be described as light-weight container, as it does not have any installation, confirguration, start and stop activities of containers etc.

Spring employs 4 key strategies:

-> light weight and minimally usage of POJO

-> loose couling through dependency injection

-> declarative programming through aspects and common conventions

-> boilerplate reduction through aspects and templates

Spring Feature/Modules:

-> Inversion of Control

-> Aspect Oriented Programming

-> Data Access

-> Model-View-Controller

-> Remote Access framework

-> Remote Management

-> Messaging

-> Testing

-> Convention over Configuration

-> Authentication and Authorization

IOC - inversion of control -> making low level items to depend on high level items

Spring configuration - 3 methods

1. XML based config file

2. Annotation-based config file

3. Java-based config file

in the world there are 3 but spring supports 2 types of injections

1. setter injection

2. constructor injection

----------

setter injection

- a class must have a setter method to inject an object into it.

public class B{

A a; // in java it is an instance variable,,, but in spring.. this is no more an instance variable,, this is a dependency

public void setA(A a){ this.a = a;}

}

ClassPathXmlApplicationContext is a implementation of application context

init-method in xml will be invoked immediately after bean instantiation

destroy-method in xml will be invoked just before bean dying

BeanFactory interface

- a root interface for accessing Spring bean container

DTD - Document Type Definition

- it is going to validate XML file

spring says,, give me all the beans in beans.xml file,, then i'll bring you all dependencies

https://docs.spring.io/spring-framework/reference/core/beans/factory-scopes.html#:~:text=When%20you%20create%20a%20bean,instances%20from%20a%20single%20recipe.

DTD became older..

later.. 'XML schema' cmae to validate XML documents

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

Spring MVC - is a architechtural pattern and not a design pattern

challenges will be faced if our servlet containers..

-> contain business logic

-> presentation login

-> request processing logic

MVC is all about decoupling these three layers

Controller

-> directly or indirectly a servlet

-> request from web hits the controller first

Model

-> makes changes and stores values

-> this has business logic and data access logic also will have state of objects

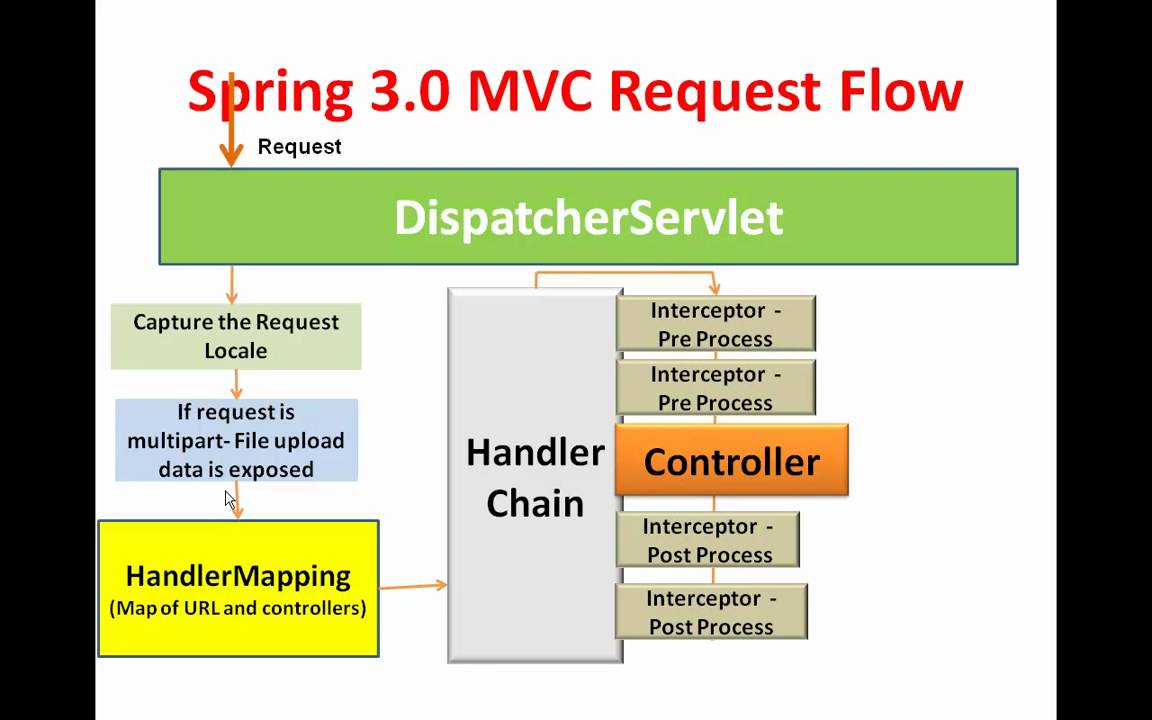
-> model is like Entity in hibernate

View

-> we build view with the help of data from the model

https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html

Dispatcher servlet is the front controller of the application



interceptors or preprocessors will execute common logics for all the code

Simple and Straightforward

What?

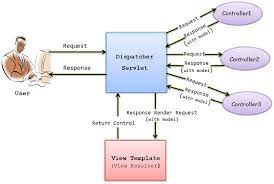
Interceptor is a components which intercets request so that you can perform some task before and after request reaches its handler

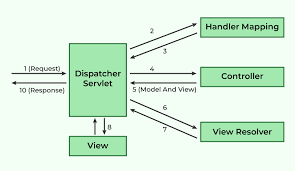
How to use it?

Implement HandlerInterceptor annotate it with “@Component”. Spring will autodetech it from scaning pacakage

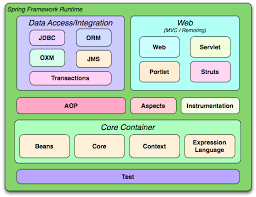
You can define in xml configuration also.

Flow of control is below:





Total Spring framework architecture below:

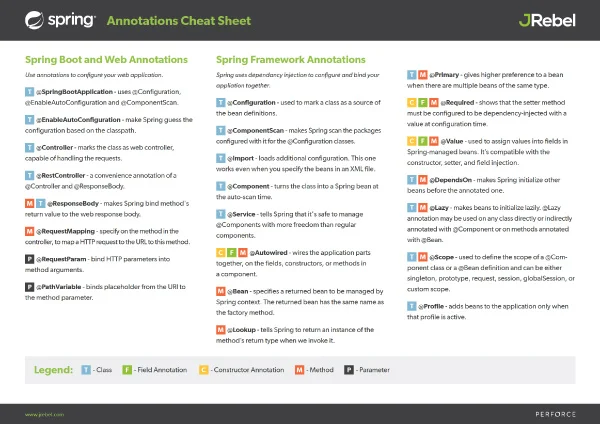


For running spring mvc application, we need

1. Java
2. Spring dependencies + (web.xml, dispatcher.xml, controllers, views)
3. Tomcat
4. Maven with the help of Eclipse

***Spring and spring boot cheat sheet***

<https://www.jrebel.com/blog/spring-annotations-cheat-sheet>



Spring Data Access:

<https://docs.spring.io/spring-framework/reference/data-access/jdbc/core.html>

Now, the hibernate can handled by spring itself by using DAO

—-----------------

Entities are records in database

CRUD operation

DAO interface has values to persist student/employee objects..

EmployeeDAOEmpl should have all method body

Maven helps us for jar and library dependency..

Basically spring is all about object dependency

spring supports injection of objects to its parent interface

good way of software development si to create an iterface first and a concrete class for it

ApplicationContext

- in enterprise applciations, context is where bindings happen

- it is going to create a cintainer more powerfull than bean container.. and ninds all beans in side the container

1. Security
2. Exceptions
3. transactions

Are the three concerns of applications which will be handled by AOP

—-----------

ORM

1. Datasource
2. Session Factory

JDBC only gives one connection at a time,,, after the db task is over, we will close the connection..

But in enterprise application, DatSource will return multiple connections and after completing the task, the connection is not closed, it goes to a pool and we can reuse it for other tasks.