Spring Framework
=================

Framework	: Application Framework
Version	: 5.3.9 (5.X)
Vendor	: Interface2
Creator	: Rod Johnson
Containers	: BeanFactory container

ApplicationContext container

website : www.tutorialspoint.com
www.roseindia.net
www.javatpoint.com
www.Dzone.com
Books : spring in Action
Download link :
https://repo.spring.io/ui/native/release/org/springframework/spring/

Q)What is spring Framework?
A spring is a non-invansive,light weight, loosely coupled, dependency injection,
Aspect oriented , open source java based application framework which is used to develop all types of applications.

Q)How many modules are there in spring framework?
There are six modules present in spring framework.
CORE

It is a base module for entire spring framework.
It is used to perform following activities.
ex:
1) It obtain containers
2) It performs dependency injections
3) It creates spring beans
AOP

AOP stands for Aspect Oriented Programming.
It is used to inject and eliminate middleware services to/from the application.
DAO

DAO stands for Data Access Object.
It is a abstract layer on JDBC.
While working JDBC we need to handle the exceptions.That problem is eliminated in DAO module.Here checked exceptions converts to unchecked exceptions.So we don't need to handle any exceptions.
ORM

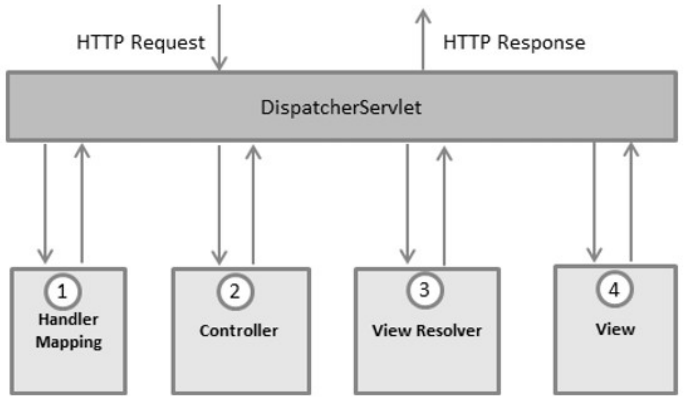
ORM stands Object Relational Mapping.
ORM is a abstract layer on ORM tools like hibernate,ibatis,jpa and etc.
While transfer the data in the form objects there are some drawbacks, those problems solved by ORM module.

JEE

This is module is used to develop middleware services.
MVC

MVC stands for Model View Controller.
It is used to create completely MVC based web application.

Q)Explain spring MVC architecture?



1)After receiving an HTTP request, DispatcherServlet consults the HandlerMapping to call the appropriate Controller.
2)The Controller takes the request and calls the appropriate service methods based on used GET or POST method. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.
3)The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request.
4) Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

Limitations with Spring Framework
====================================
In spring framework, A developer is responsible to
i)Add dependencies/Jar files
ii)Perform configurations.
iii)Arrange a virtual server (Tomcat, GlassFish and etc) to deploy web applications.
iv)Arrange a Physical database (Oracle,Mysql,mongodb and etc) to perform database operations.

We can overcome above limitations by using spring Boot.
Diagram:
Developer
|
Spring Boot
|
Spring

What is Spring Boot?
===================
Spring Boot is an open source Java-based framework developed by Pivotal Team.

Spring Boot provides the RAD (Rapid Application Development) feature to the Spring framework.
Spring Boot is used to create stand-alone, production-grade Spring based Applications
with minimum configuration.
In short, Spring Boot is the combination of
ex:
Spring Framework +   Embedded Servers   + Embedded Database.
(Tomcat/Jetty/Undertow)   (H2/HSQL/Derby)

In Spring Boot, there is no requirement for XML configuration.
XML configurations are replaced by Annotations.

Advantages of Spring Boot (staters)
=====================
It creates stand-alone Spring applications that can be started using Java -jar.
It provides production-ready features such as metrics, health checks, and externalized configuration.
It increases productivity and reduces development time.
It offers the number of plug-ins.
There is no requirement for XML configuration.
It provides opinionated 'starter' POMs to simplify our Maven configuration.
It tests web applications easily with the help of different Embedded HTTP ervers such as Tomcat, Jetty,Undertow etc. We don't need to deploy WAR files.
It offers a CLI tool for developing and testing the Spring Boot application.
It also minimizes writing multiple boilerplate codes (the code that has to be included in many places with little or no alteration), XML configuration, and annotations.

Q)Components of spring boot?
We have four components in spring boot.
1) Autoconfiguration
2) starter
3) CLI
4) Actuators

Q)In how many ways we can create a spring boot project?
There are two ways to create spring boot project.
1)Using IDE's (STS , IntellIJ)
2)Using spring Initializr

STS IDE
===========
Spring Tool Suite is an IDE to develop Spring applications.
It is an Eclipse-based development environment.
It provides a ready-to-use environment to implement, run, deploy, and debug the application.
Step 1:

Download Spring Tool Suite
ex:
https://spring.io/tools#suite-three

Step 2:

Extract the zip file and install the STS.
ex:
sts-bundle -> sts-3.9.9.RELEASE -> Double-click on the STS.exe.

Spring Boot Starters
=====================
Spring Boot provides a number of starters that allow us to add jars in the classpath.
Spring Boot built-in starters make development easier and rapid.
Spring Boot Starters are the dependency descriptors.
In the Spring Boot Framework, all the starters follow a similar naming pattern:
spring-boot-starter-*, where * denotes a particular type of application.
ex:
spring-boot-starter-test
spring-boot-starter-web
spring-boot-starter-validation (bean validation)
spring-boot-starter-security
spring-boot-starter-data-jpa
spring-boot-starter-data-mongodb
spring-boot-starter-mail

Third-Party Starters
==================
We can also include third party starters in our project.
The third-party starter starts with the name of the project.
ex:
abc-spring-boot-starter.

Spring Boot Starter Web
=======================
There are two important features of spring-boot-starter-web.
>It is compatible for web development
>AutoConfiguration

If we want to develop a web application,we need to add the following dependency in pom.xml file.
ex:
```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<version>2.2.2.RELEASE</version>
</dependency>
```

Spring web starter uses Spring MVC, REST and Tomcat as a default embedded server.
The single spring-boot-starter-web dependency transitively pulls in all dependencies related to web development.
By default, the spring-boot-starter-web contains the following tomcat server dependency:
ex:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
<version>2.0.0.RELEASE</version>
<scope>compile</scope>
</dependency>
```

The spring-boot-starter-web ,auto-configures the following things that are required for the web development:

1)Dispatcher Servlet
2)Error Page
3)Web JARs for managing the static dependencies
4)Embedded servlet container

Interview question
================
Q)Explain @SpringBootApplication annotation?
This annotation is used to enable the following annotations.

@EnableAutoConfiguration: It enables the Spring Boot auto-configuration mechanism.
@ComponentScan: It scans the package where the application is located.
@Configuration: It allows us to register extra beans in the context or import additional configuration classes.

Q)Where to do spring boot application configurations?
We can configure application configuration in two files.
1)application.properties file
2)application.yml file

Q**)List of sterotype Annotation?**
@Component
@Configuration
@Service
@Repository
@Controller
and etc.

Q)In spring boot mvc based web application who will pass HTTP request to controller?
ans) RequestDispatcher.

Q)List of embedded servers present in spring boot?
ans) Tomcat, Jetty and undertow

Q)Tomcat embedded server by default runs under which port no?
8080
Q)To create a spring mvc based web application we need to add which starter?
spring-boot-stater-web

Q)How to change tomcat embedded server port no?

ans) application.properties
```
application.properties
----------------------------
server.port = 9090
```

Spring Boot + JSP Application
==============================
Project structure

```
MVCApp1
|
|----src/main/java
|           |
|           |----com.ihub.www
|                       |
|                       |--MVCApp2Application.java
|                       |--HomeController.java
|---src/main/resources
|           |
|           |-----application.properties
|
|---src/test/java
|           |
|           |-----MVCApp1ApplicationTests.java
|
| --
| --
| --
|-----src
|
|----main
|
|----webapp
|
|----pages
|           |
|----index.jsp
|---pom.xml
|
|
```
step1:

Create a spring starter project.
ex:
File --> new --> spring starter project -->
Name : MVCApp1
Group: com.ihub.www
Artifact: MVCApp1

Description: This is Spring Boot Application with JSP
package : com.ihub.www ---> next -->
Starter: Spring Web --> next --> Finish.
step2:
------
create a HomeController class inside "src/main/java".
ex:
Right click to package(com.ihub.www) --> new --> class --> Class: HomeController
-->finish.
step3:

Add @Controller annotation and "@RequestMapping" annotation inside HomeController class.

HomeController.java

```
package com.ihub.www;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {

@RequestMapping("home")
public String home()
{
return "index";
}
}
```

step4:
-----
create a "webapp" and "pages" folder inside "src/main" folder for adding JSP files.
ex:

```
|-----src
|
|----main
|
|----webapp
|
|-----pages
```

step5:
----
create "index.jsp" file inside "src/main/webapp/pages/" folder.
ex:
Right click to pages folder--> new --> file --->
File Name: index.jsp --> finish.

index.jsp

```
<center>
<h1>
I love Spring Boot Programming
</h1>
</center>
```

step6:

Add "Tomcat Embed Jasper" dependency to read the jsp file.

ex:
```
<dependency>
<groupId>org.apache.tomcat.embed</groupId>
<artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

Note:

Embedded Tomcat server does not have Jasper. So we need to add above dependency.
step7:

Configure tomcat server port number and jsp file.

application.properties

```
server.port=9191

spring.mvc.view.prefix=/pages/
spring.mvc.view.suffix=.jsp
```

step8:

Run Spring Boot application.
ex:
Right click to MVCApp2 --> run as --> spring boot application.
step9:

Test the application with below request url.
ex:
http://localhost:9191/home

Spring Data JPA
==================
Spring Data JPA handles most of the complexity of JDBC-based database access and ORM (Object Relational Mapping).

It reduces the boilerplate code required by JPA(Java Persistence API).
It makes the implementation of your persistence layer easier and faster.
Spring Data JPA aims to improve the implementation of data access layers by reducing the effort to the amount that is needed.
Spring Boot provides spring-boot-starter-data-jpa dependency to connect Spring application with relational database efficiently.
ex:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
<version>2.2.2.RELEASE</version>
</dependency>
```

The spring-boot-starter-data-jpa internally uses the spring-boot-jpa dependency.
Spring Data JPA provides three repositories are as follows:

CrudRepository:

It offers standard create, read, update, and delete It contains method like findOne(), findAll(), save(), delete(), etc.

PagingAndSortingRepository:

It extends the CrudRepository and adds the findAll methods. It allows us to sort and retrieve the data in a paginated way.

JpaRepository:

It is a JPA specific repository It is defined in Spring Data Jpa. It extends the both repository CrudRepository and PagingAndSortingRepository. It adds the JPA-specific methods, like flush() to trigger a flush on the persistence context.

Spring Boot application to interact with H2 Database
===
project structure

```
MVCApp2
|
|----src/main/java
|
|---com.ihub.www
|           |
|---MVCApp2Application.java
|
|---com.ihub.www.controller
|
|---EmployeeController.java (Class)
|
|---com.ihub.www.repository
```

```
|
|---EmployeeRepository.java (Interface)
|
|---com.ihub.www.model
|
|---Employee.java (Class)
|
|
|----src/main/resources
|
|---application.properties
|

|-----src
|
|-----main
|
|---webapp
|
|----index.jsp
```
step1:

Create a spring boot starter project i.e MVCApp2.
ex:
starters:
spring web
spring data jpa
H2 Database
step2:

Add "Tomcat Embed Jasper" dependency to read the jsp file inside pom.xml.
ex:
@Controller
step3:

Create a EmployeeController inside "com.ihub.www.controller" package.

EmployeeController.java

```java
package com.ihub.www.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import com.ihub.www.model.Employee;
import com.ihub.www.repository.EmployeeRepository;
```

```
@Controller
public class EmployeeController
{

@Autowired
EmployeeRepository employeeRepository;

@RequestMapping("/")
public String home()
{
return "index.jsp";
}

@RequestMapping("/addEmp")
public String addEmployeeDetails(Employee e)
{
employeeRepository.save(e);

return "index.jsp";
}
}
```

step4:

Create index.js file inside "src/main/webapp" folder.
index.js

```
<form action="addEmp">
<table align="center">
<tr>
<td>Employee Id </td>
<td><input type="text" name="empId"/></td>
</tr>
<tr>
<td>Employee Name </td>
<td><input type="text" name="empName"/></td>
</tr>
<tr>
<td>Employee Salary </td>
<td><input type="text" name="empSal"/></td>
</tr>
<tr>
<td><input type="reset" value="reset"/></td>
<td><input type="submit" value="submit"/></td>
</tr>
</table>
</form>
```

step5:

Create a Employee.java file inside "com.ihub.www.model" package.
Employee.java

```
package com.ihub.www.model;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table
public class Employee
{
@Id
private int empId;
@Column
private String empName;
@Column
private double empSal;

public int getEmpId() {
return empId;
}
public void setEmpId(int empId) {
this.empId = empId;
}
public String getEmpName() {
return empName;
}
public void setEmpName(String empName) {
this.empName = empName;
}
public double getEmpSal() {
return empSal;
}
public void setEmpSal(double empSal) {
this.empSal = empSal;
}
}
```

step6:

Create a EmployeeRepository.java interface inside "com.ihub.www.repository" package.

EmployeeRepository.java

```java
package com.ihub.www.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.ihub.www.model.Employee;

@Repository
public interface EmployeeRepository extends CrudRepository<Employee,Integer>
{

}
```

step7:

Configure server port and h2 database properties inside
application.properties file.

application.properties

```
server.port=9090

spring.datasource.url= jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

spring.h2.console.enabled=true

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
```

step8:

Run the spring boot starter project.
step9:

Test the application by using below request url.
ex:
http://localhost:9090
http://localhost:9090/h2-console