SESSION PROGRAMS(SPRING CORE)

Frameworks in Java

Any Java Framework is a collection of predefined classes and interfaces that is used to simplify the traditional JDBC or Web Development approaches. Typically comes with multiple .jar files which need to be setup in the project ,so as to use that framework ready made Classes /Objects in our code.

Top 10 Java Frameworks You Should Know

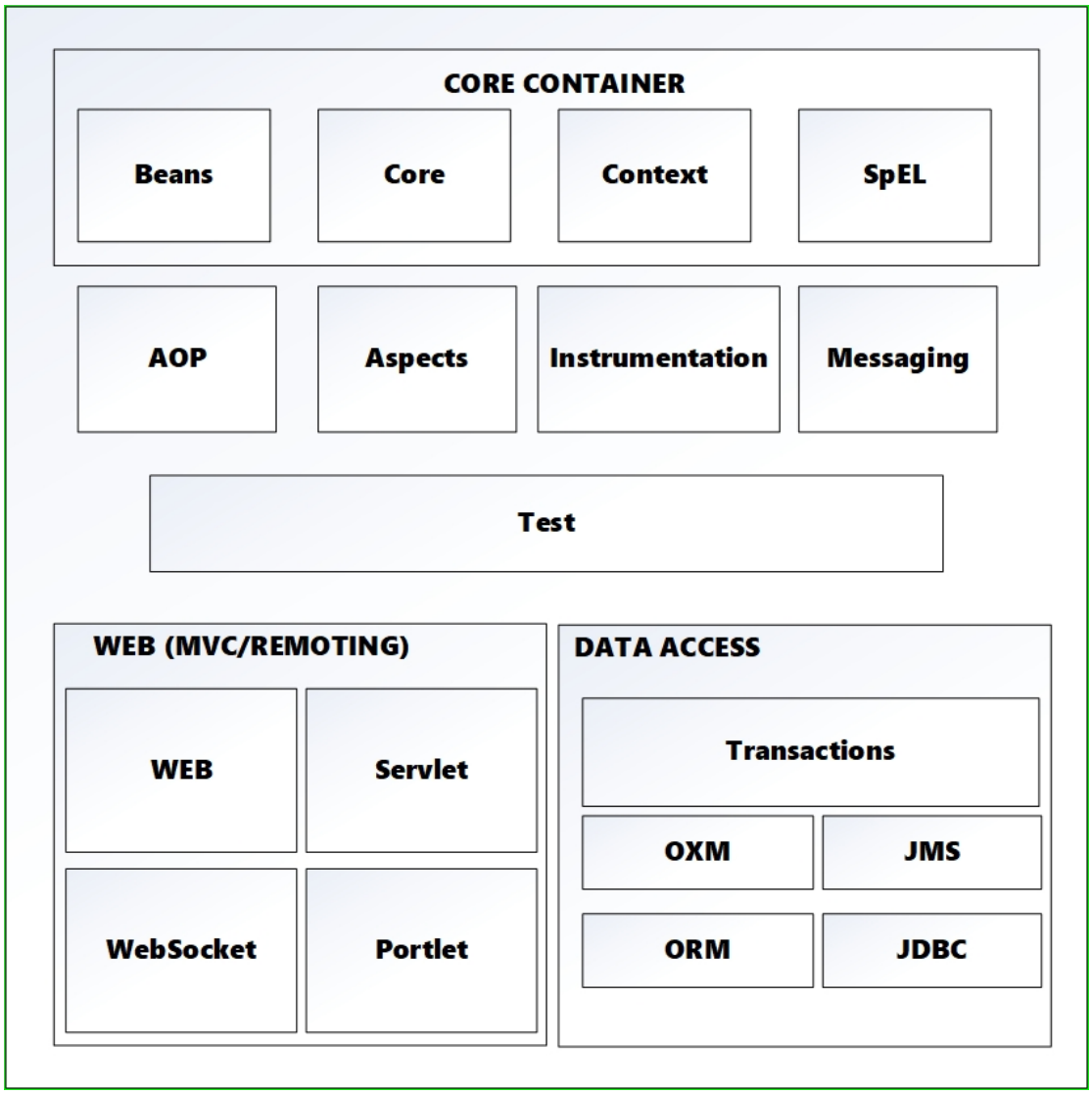
Spring.
Hibernate.
Struts.
Google web toolkit [GWT]
JavaServer Faces [JSF]
Grails.
Vaadin.
Blade.

Note

For all the above Frameworks download their jar files & setup manually in your Project or the more better way is to use Maven Based Projects.

Spring Framework Architecture

Spring Framework is divided into a number of separate modules, which allows you to decide which ones to use in your application.



Spring Framework Core Components

The Core container from Spring consists of four modules: SpEL , Context, Core, Beans. Description for these elements are as follows:

The **SpEL** module provides a powerful expression language for manipulating objects during execution.

Context is built on the basis of Beans and Core and allows you to access any object that is defined in the settings. The key element of the Context module is the ApplicationContext interface.

The **Core** module provides key parts of the framework including IoC and DI properties.

The **Bean** module is responsible for creating and managing Spring Beans - is application context structure unit.

Spring Framework Web
Spring framework Web layer consists of Web, Web-MVC, Web-Socket, Web-Portlet etc.

The **Web** module provides functions such as downloading files, creating web application, rest web service etc.

Web-MVC contains a Spring MVC implementation for web applications.

Web-Socket provides support for communication between the client and the server, using Web-Sockets in web applications.

Web-Portlet provides MVC implementation with portlet environment

Spring Framework Data Access
The Data Access/Integration container consists of JDBC, ORM, OXM, JMS and the Transactions module.

Spring Core Concepts

Dependency Injection

Advantages :

1. Makes application LightWeight i.e. less memory consumption due to less no of object instances created

2. Gives total control over Objection creation & destruction . This gives control over how long the object should be in memory . i.e. for Request or for Session or for entire Application scope

3. Any code changes in the Service Bean Class , would not lead to changes in the Consumer classes , thereby making code maintenance & testing easy

Simple Java Example Code for demonstrating Dependency Injection :

```
public class Table {
        synchronized void printTable(int n){//synchronized method
                for(int i=1;i<=5;i++){
                  System.out.println(n*i);
```

```
                    try{
                     Thread.sleep(400);
                    }catch(Exception e){System.out.println(e);}
                  }
 }
}
public class MyThread1 extends Thread {
        Table t;
        MyThread1(Table t){
        this.t=t;
        }
        public void run(){
        t.printTable(5);
        }
}

public class MyThread2 extends Thread {
        Table t;

        MyThread2(Table t){
        this.t=t;
        }

        public void run(){
        t.printTable(100);
        }

}


public class Main {
        public static void main(String args[]){
                Table obj = new Table();//only one object
                MyThread1 t1=new MyThread1(obj);
                MyThread2 t2=new MyThread2(obj);
                t1.start();
                t2.start();
                }
}
```

Spring Dependency Injection

```
public class Student {

   private String name;

   public String getName() {
       return name;
   }

   public void setName(String name) {
       this.name = name;
   }

   public void displayInfo(){
       System.out.println("Hello: "+name);
   }

}
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd">

            <bean id="studentbean" class="com.springcore.examples.Student">
                    <property name="name" value="Vimal
Jaiswal"></property>
                </bean>

</beans>
```

```
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test {

   public static void main(String[] args) {
       Resource resource=new ClassPathResource("applicationContext.xml");
       BeanFactory factory=new XmlBeanFactory(resource);

       Student student=(Student)factory.getBean("studentbean");
       student.displayInfo();
   }

}
```