Android : open specification : - Android OS (Lg, Xiomi, Samsung, Oppo)

IOS : proprieteray specification : IOS OS (Apple)

--these servers runs all those challenges to develop an EA in the background, so that we can focus on the main business logic, that's why we call them as a f/w software.

--J2EE specification provides lots of api (predefined classes and interfaces) to use those services so that we just need to focus on main business logic.

JAAS api (Java Authentication and Autherization api) / Spring security
Servlet/JSP api ----> web enabled (webapplication).
JTA api (Java Transaction api)
REST api (to generate webservices)
JNDI api
etc.

--we get all these api along with ApplicationServer.

architecture of the ApplicationServer s/w:
==========

Webserver:
=========

--it is a subset or small part of ApplicationServer.

--the most famous java based webserver is Apache tomcat server.

Diff bt WebServer and ApplictionServer:
==========

1. Java based webserver is the small part/ subset of the ApplicationServer.

--AS contains both EJB container and ServletContainer s/w where as Webserver s/w contains only ServletContainer s/w, it does not contains EJB-container.

2.AS can be communicated with any kind of protocol ex: http, smtp, ftp, etc, whereas WS can be communicated only using Http protocol.

3. AS is a heavy weight server where as Webserver is a lightweight server.

Diff bt Webapplication and Enterprise Application:
==========

--WebApplication is a small part of the Enterprise application.

**--Container is a special type of s/w which provides runtime support (execution environment
) to our components (i.e creating our classes objects, calling their method, etc.)**

Note: After spring, with the help of Webserver also we can develop Enterprise application very easily.

Spring f/w:
==========

--spring is a application f/w software to develop EA.

--s/w communities treats spring as a framework of frameworks, because it gives the support of varaious other f/w software also. like hibernate, struts, JSF,etc

--spring is an open source, light weight application f/w that can be used in all the layers of Java based Enterprise application development.

--with the Spring f/w we can develop J2SE related or J2EE related application as well.

----in early days of java based business application development, programmer used multiple java bean classes (normal java classes) to build the business logic layer/service layer.

--The Business logic layer only required the Enterprise capabilities like security, tx-management, logging, mailing, messaging, etc, these enterprise capabilities are also known as extra service to the main business logic or middlewere services to make our main business logic perfect.

--with the Java bean classes, developers are only responsible to define and add these middlewere services to the main business logic, it increases burden to the developers.

//this method contains a business logic
public void transferAmount(int srcAccount, int desAccount, int amount){

//security logic
//logging logic
//tx-management
//messaging
//mailing
/etc.

//check the accno from the DB.
//check the appropriate balance in srcAccount
//deduct the amount from the source account
//add the amount to the destination account
}

--to overcome the above burden to develop a business logic/service logic, sun-microsystem has released EJB technology as part of J2EE specification. in 1998.

--In EJB technology, programmer develop the main business logic and EJB-container provides these extra middlewere services.

--EJB reduced the middlwere service development from the programmer, but it increased the complexities to access these middlewere services.

--EJB component are heavy weight components (Here our java classes need to be developed as EJB component by implementing EJB technology related interfaces, need to override lost of unnessasry methods inside our java classes and need to register these java classes inside various xml files and deploy our EJB component inside the ApplicationServer s/w).

--EJB has beed very powerfull but very complex to the build the Business Logic layer.

--to the build the business logic layer, simplicity of Java bean classes + Power of EJB - complexities of EJB was realised in the industry.

--Rod Jonshon, developed a f/w called "Interface 21" to address the above need and rename it to the Spring and released in mid of 2004.

--Spring f/w allows to write the business logic in a POJO classes, and Spring Container provides the middleware services to our main business class with less process overhead.

--so our business logic related classes need not implement or extends any other classes or interfaces or override any unnessasry methods. no need to install ApplicationServer s/w.

--unlike EJB container, Spring container is a lightweight container, can be activated in any type of Java application (need not install AS s/w).

---> classses, interface

option 1: Has-A relationship

```
class A{   // dependency class

void funA(){
--
}

}


class B{  // dependent class

A a1 = new A(); //Has-A relationship

void funB(){
a1.funA();
}

void funC(){

a1.funA();
}

}
```

Option 2: Is-A relationship:

```
class A{

void funA(){
--
}

}


class B extends A{


void funB(){
funA();
}

void funC(){

funA();
}

}
```

--->In java Has-A comparativly better than Is-A.