**Unit 3**. Software architecture implementation technologies: Software Architecture Description Languages (ADLs), Struts, Hibernate, Node JS, Angular JS, J2EE – JSP, Servlets, EJBs; middleware: JDBC, JNDI, JMS, RMI and CORBA etc. Role of UML in software architecture.

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

**Software architecture implementation technologies:**

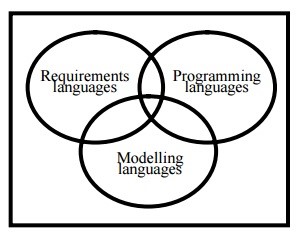
An architectural implementation framework is a piece of software that acts as a bridge between a particular architectural style and a set of implementation technologies. It provides key elements of the architectural styles in code, in a way that assists developers in implementing systems that conform to the prescriptions and constraints of the style.

Architectural implementation frameworks are a form of middleware. The difference between traditional middleware and architectural framework is the focus on architectural styles. Implementation frameworks are implemented specifically to support development in one or more architectural styles. Here the style is the primary artifact driving the implementation technology. Middleware is created based on services that are provided, generally without regard to the style of the application being developed.

**Architecture description languages:-**

Architecture description languages (ADLs) are formal languages that can be used to represent the architecture of a software-intensive system. As architecture becomes a dominating theme in large system development, methods for unambiguously specifying architecture will become indispensable.

By architecture, we mean the components that comprise a system, the behavioral specifications for those components, and the patterns and mechanisms for interactions among them. Note that a single system is usually composed of more than one type of component: modules, tasks, functions, etc. Architecture can choose the type of component most appropriate or informative to show, or it can include multiple views of the same system, each illustrating different componentry.

**[](https://www.todaysoftmag.com/images/articles/tsm55/a32.jpg)**

To date, architectures have largely been represented by informal circle-and-line drawings in which the nature of the components, their properties, the semantics of the connections, and the behavior of the system as a whole are poorly (if at all) defined. Even though such figures often give an intuitive picture of the system's construction, they usually fail to answer such questions as:

* What are the components? Do the modules exist only at design-time, or are they compiled together before run time? Are these tasks or processes threaded together from different modules, assembled at compile-time, and do they form run time units? Are they something as nebulous as "functional areas," as in data flow diagrams, or something else entirely?
* What do the components do? How do they behave? What other components do they rely on?
* What do the connections mean? Do they mean "sends data to," "sends control to," "calls," "is a part of," some combination of these, or something else? What are the mechanisms used to fulfill these relations?
* ADLs result from a linguistic approach to the formal representation of architectures, and as such they address the shortcomings of informal representations. Sophisticated ADLs allow for early analysis and feasibility testing of the design decisions.

ADLs trace their roots to module interconnection languages of the 1970s. Today, ADLs are in a maturing phase, but several exist. Current examples include Rapide, UniCon, ArTek, Wright and Meta-H.

**Architecture and ADLs**

Architecture plays several roles in project development, all of them important, and all of them facilitated by a formal representation of the architecture, such as with an ADL. A formal architecture representation is more likely to be maintained and followed than an informal one, because a formal architecture can more readily be consulted and treated as authoritative, and can more easily be transferred to other projects as a core asset. Roles include:

* **Basis for communication:** Project team members, managers, and customers all turn to the architecture as the basis for understanding the system, its development, and how it works during execution.
* **Project blueprint:** The choice of architectural components is institutionalized in the developing organization's team structure, work assignments, management units, schedule and work breakdown structures, integration plans, test plans, and maintenance processes. Once it is made, an architectural decision has an extremely long lifetime and survives even outside of the software that it describes.
* **Blueprint for product line development.** Architecture may be re-used on other systems for which it is appropriate. If managed carefully, an entire product family may be produced using a single architecture. In this case, the importance of an appropriate architecture is magnified across all the projects it will serve.
* **Embodiment of earliest design decisions:** The architecture represents the first mapping from requirements to computational components. The selection of components and connections, as well as the allocation of functionality to each component, is a codification of the earliest design decisions about a project. All downstream design decisions must be consistent with the architectural choices. As such, architectural decisions are the hardest to change, and have the most far-reaching consequences.
* **First approach to achieving quality attributes: A**rchitecture can either allow or preclude the achievement of most of a system's targeted quality attributes. Modifiability, for example, depends extensively on the system's modularization, which reflects the encapsulation strategies. Component reusability depends on how strongly coupled components are with other components in the system. Performance depends largely on the volume and complexity of the inter-component communication and coordination, especially if the components are physically-distributed processes. Thus, architecture embodies decisions about quality priorities and tradeoffs, and represents the earliest opportunity for evaluating those decisions and tradeoffs.

Some ADLs provide an opportunity for architecture-level analysis, such as automatic simulation generation, schedulability analysis, and the like. However, even in the absence of automated analysis capabilities, other evaluative strategies can be applied to the architecture. Thus, these early design decisions and quality attribute tradeoffs can be tested before they are too expensive to change.

**Architecture Description Language (ADL)** is defined as "a language (graphical, textual, or both) for describing a software system in terms of its architectural elements and the relationship among them".

**In** other words, ADL is a language enabling formalization, description, specification, modeling and reasoning on software architectures. Each of these features should be fulfilled by a language that is proclaimed to be ADL. A good ADL must provide abstractions that are adequate for modeling a large system. Each ADL embodies a particular approach to the specification and evolution of architecture.

**Struts:**

**Struts** are used to create a web applications based on Servlet and JSP. Struts depend on the MVC (Model View Controller) framework. Struts application is a genuine web application. Struts are thoroughly useful in building J2EE (Java 2 Platform, Enterprise Edition) applications because struts takes advantage of J2EE design patterns. Struts follow these J2EE design patterns including MVC.

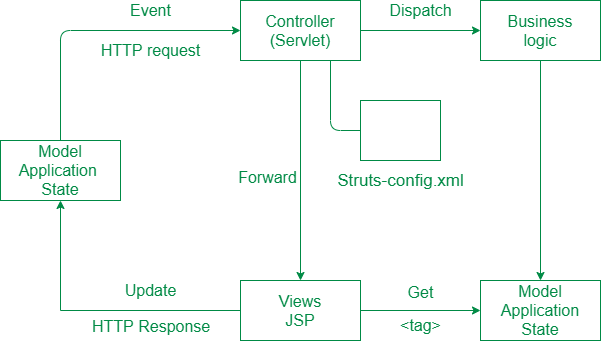
In struts, the composite view manages the layout of its sub-views and can implement a template, making persistent look and feel easier to achieve and customize across the entire application. A composite view is made up by using other reusable sub views such that a small change happens in a sub-view is automatically updated in every composite view.

Struts consists of a set of own custom tag libraries. Struts are based on MVC framework which is pattern oriented and includes JSP custom tag libraries. Struts also support utility classes.

**Features of Struts:** Struts has the following features:

* Struts encourage good design practices and modeling because the framework is designed with “time-proven” design patterns.
* Struts are almost simple, so easy to learn and use.
* It supports many convenient features such as input validation and internationalization.
* It takes much of the complexity out as instead of building your own MVC framework, you can use struts.
* Struts is very well integrated with J2EE.
* Struts has large user community.
* It is flexible and extensible; it is easy for the existing web applications to adapt the struts framework.
* Struts provide good tag libraries.
* It allows capturing input from data into javabean objects called Action forms.
* It also hand over standard error handling both programmatically and declaratively.

**Working of Struts:**



In the initialization phase, the controllers rectify a configuration file and used it to deploy other control

Layer objects. Struts configuration is form by these objects combined together. The struts configuration defines among other things the action mappings for an application.  
Struts controller Servlets considers the action mappings and routes the HTTP requests to other components in the framework. Request is first delivered to an action and then to JSP. The mapping helps the controller to change HTTP requests into application actions. The action objects can handle the request from and responds to the client (generally a web browser). Action objects have access to the applications controller servlet and also access to the Servlets methods. When delivering the control, an action objects can indirectly forward one or more share objects, including javabeans by establish them in the typical situation shared by java Servlets.

<https://www.codejava.net/frameworks/struts/introduction-to-struts-2-framework>

**Hibernate architecture**

The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

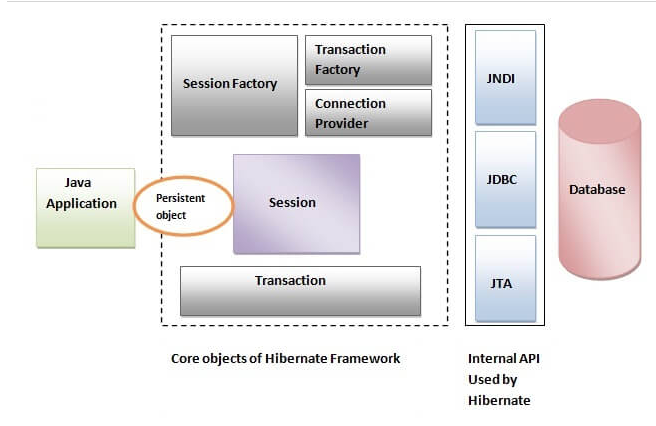
The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand API layer
* Database layer

Let's see the diagram of hibernate architecture:



This is the high level architecture of Hibernate with mapping file and configuration file.



Hibernate framework uses many objects such as session factory, session, transaction etc. alongwith existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface).

**Architecture**

To understand its architecture this comprises of:

* A Java application code consists of all the classes, variables and objects that define the business logic of the application. These classes communicate with the Hibernate.
* Then comes the Hibernate and its core principles, using which we could persist (store or save and retrieve) the objects of our business layer classes by communicating with the database server layer.
* It uses the Java core API, Java Database Connectivity (JDBC), Java Transaction API(JTA), Java Naming and Directory Interface(JNDI) to communicate with the database in order to persist the state of an object by performing read, create, update, delete(CRUD) operations.

It is basically used to communicate with the database. Here we just create some entities, which is nothing but a java class that will create a database table for us in the database.

**Framework**

Here we come across a question that why we use hibernate framework of java if we already have the core java principles. So here is the answer to that :

* Used best JDBC concepts such as:

1. Row set instead of ResultSet
2. Data Source (connection pooling) instead of DriverManager
3. Batch operations instead of individual operation
4. Prepared Statement instead of Statement

* Project development cost is reduced to a large factor.
* No memory leaks
* Managing associations such as one-to-many, etc., are simple and easy as compared to working with collection properties like java.util.Set, List, Map.
* Being an ORM framework, it will get all its advantages:
* Supports Second level cache
* The object becomes a record and vice versa.
* Additionally supports Object Query Language (HQL)
* Supports multiple primary key generators

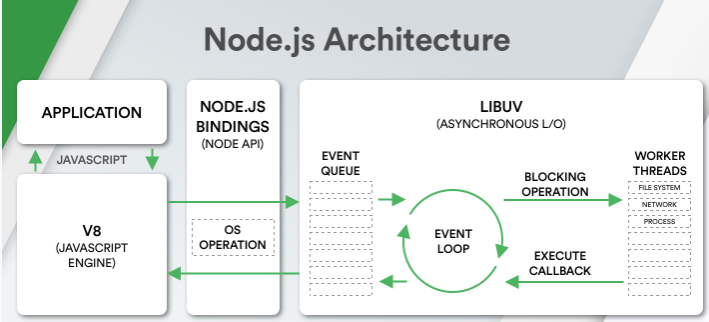
**Features**

So it was all about the java hibernate framework and its architecture; let’s talk about the features of the latest

Hibernate stable released on 23 Feb 2019. Developed by Red Hat software, it is a virtual machine platform with lots of features:

* It supports JMX (Java Management Extensions) and JCA (Java Cryptography Architecture).
* It provides a feature of J2EE integration.
* At system initialization time, it generates SQL.
* Optionally provide internal connection pooling and prepared statement caching.
* It supports optimistic locking with versioning.
* It provides outer join fetching.
* It introduces Lazy initialization.
* It provides session-level cache and optional second-level cache.
* It provides Dual-layer Cache Architecture.
* Automatic generation of the primary key.
* It supports the tough concept of composite keys.
* It supports the detached object concept.
* It introduces the automatic Dirty Checking concept.
* It provides transparent persistence without byte code processing.
* It provides Object/Relational mappings. Here are different O/R mapping strategies as multiple-objects to single-row mapping,
* Polymorphic associations, bi-directional association, association filtering. It also provides XML mapping documents.
* It provides different object-oriented query languages.
* Minimal object-oriented Hibernate query language(HQL), native SQL queries
* High object-oriented concept of criteria.

**Node.js**

****

Node.js is a cross-platform environment and library for running JavaScript applications which is used to create networking and server-side applications.

## ****What is JavaScript?****

[Javascript](https://www.javatpoint.com/javascript-tutorial) is a **lightweight, object-oriented scripting** language that is used to build **dynamic HTML pages** with interactive effects on a webpage. JavaScript is also commonly used in **game development** and **mobile app development**. It is an interpreted scripting language, and the code can only be executed and run in a web browser. We can use Node.js to execute and run the code outside of the browser. It's also known as a **browser's language**, and it can be used for both **client-side** and **server-side** development. **Brendan Eich** of **Netscape** created it, and it was first published in **1995**. The language was originally known as **LiveScript** before being renamed JavaScript. JavaScript's syntax is heavily influenced by the [programming language **C**](https://www.javatpoint.com/c-programming-language-tutorial). The extension of JavaScript filename is **.js**.

## What is Node.js?

[Node.js](https://www.javatpoint.com/nodejs-tutorial) is a **cross-platform, open-source** JavaScript runtime environment that enables JavaScript to be run on the server. Node.js enables JavaScript code to run outside of the browser. Node.js comes with a large number of modules and is mostly used in web creation. It may run on various platforms including, **Windows, Linux, Mac OS**, etc. It provides a cross-platform runtime environment with **event-driven, non-blocking (asynchronous) I/O** for creating highly scalable **server-side** JavaScript applications.

Node.js was **developed** and **introduced** by **Ryan Dahl** in **2009**. Node.js can be used to create a variety of applications, including **web apps, real-time chat apps, command-line apps**, and REST **API servers**, among others. However, it is mainly used to build network programs such as web servers. The standard filename extension of Node.js is **.js**.

It is open source and free to use. It can be downloaded from this link <https://nodejs.org/en/>

Many of the basic modules of Node.js are written in JavaScript. Node.js is mostly used to run real-time server applications.

**The definition given by its official documentation is as follows:**

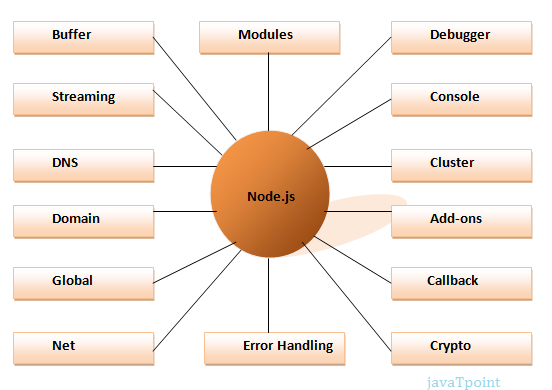
? Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices?

Node.js also provides a rich library of various JavaScript modules to simplify the development of web applications.

1. Node.js = Runtime Environment + JavaScript Library

**Different parts of Node.js**

The following diagram specifies some important parts of Node.js:



## Features of Node.js

Following is a list of some important features of Node.js that makes it the first choice of software architects.

1. **Extremely fast:**Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. **I/O is Asynchronous and Event Driven:**All APIs of Node.js library are asynchronous i.e. non-blocking. So a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call. It is also a reason that it is very fast.
3. **Single threaded:**Node.js follows a single threaded model with event looping.
4. **Highly Scalable:**Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
5. **No buffering:**Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.
6. **Open source:**Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.
7. **License:**Node.js is released under the MIT license.

**Advantages and Disadvantages of Node.js**

Various types of advantages and disadvantages of Node.js are as follows:

**Advantages**

* js is an MIT-licensed open-source framework.
* It is a lightweight framework that includes minimum modules. Other modules may be included as per the requirement of an application.
* It's a cross-platform framework that works on Windows, MAC, and Linux.
* It is used the JavaScript language to develop server-side applications.
* js applications never buffered any type of data in applications. Node.js applications simply output data in chunks.
* The Node.js library's APIs are all asynchronous or non-blocking. It simply means that a Node.js based server never waits to return data from an API. After calling an API, the server passes on to the next one, and a Node.js notification mechanism assists the server in receiving a response from the previous API call.
* js is a quick programming execution library built on the V8 JavaScript Engine in Google Chrome.

**Disadvantages**

* One of the big disadvantages of Node.js is its lack of consistency. The API changes regularly, which increases the developers' problems because they'll have to make changes to their current code base to maintain compatibility.
* It doesn't support multi-threading programming, and it is not with the development of heavy computing applications.
* In contrast to other programming languages, JavaScript lacks a well-equipped and functional library framework. As a result, users are forced to need a common library to execute various tasks, including the process of the pictures, **XML parsing, Object-Relational Mapping (ORM)**, and **handling database operations**, etc. It makes it difficult to developers to also implement basic programming tasks with Node.j

**Angular JS**

Angular JS is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone.

Angular Js is developed by Google.

It is an excellent framework for building single phase applications and line of business applications.

**Advantage of AngularJS**

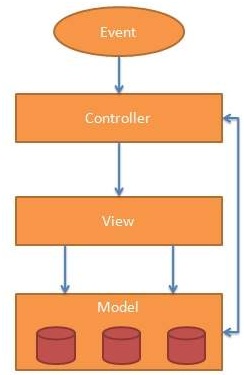
There are a lot of JavaScript frameworks for building web applications. So, it is a genuine question, why to use Angular JS.

**Following are the advantages of AngularJS over other JavaScript frameworks:**

* **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
* **Two way data binding:** AngularJS creates a two way data-binding between the select element and the orderProp model. orderProp is then used as the input for the orderBy filter.
* **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
* **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
* Directives, filters, modules, routes etc.

**AngularJS MVC Architecture**

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.



The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

**AngularJS First Example**

Following is a simple "Hello Word" example made with AngularJS. It specifies the Model, View, Controller part of an AngularJS app.

<!DOCTYPE html**>**

**<html** lang="en"**>**

**<head>**

**<script** src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js"**></script>**

**</head>**

**<body** ng-app="myapp"**>**

**<div** ng-controller="HelloController" **>**

**<h2>**Hello {{helloTo.title}} !**</h2>**

**</div>**

**<script>**

angular.module("myapp", [])

    .controller("HelloController", function($scope) {

        $scope.helloTo = {};

        $scope.helloTo.title = "World, AngularJS";

    } );

**</script>**

**</body>**

**</html>**

# ****AngularJS****

## ****What is JavaScript?****

[Javascript](https://www.javatpoint.com/javascript-tutorial) is a **lightweight, object-oriented scripting** language that is used to build **dynamic HTML pages** with interactive effects on a webpage. JavaScript is also commonly used in **game development** and **mobile app development**. It is an interpreted scripting language, and the code can only be executed and run in a web browser. We can use Node.js to execute and run the code outside of the browser. It's also known as a **browser's language**, and it can be used for both **client-side** and **server-side** development. **Brendan Eich** of **Netscape** created it, and it was first published in **1995**. The language was originally known as **LiveScript** before being renamed JavaScript. JavaScript's syntax is heavily influenced by the [programming language **C**](https://www.javatpoint.com/c-programming-language-tutorial). The extension of JavaScript filename is **.js**.

## What is AngularJS?

**AngularJS** is an open-source Model-View-Controller framework that is similar to the [JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html) framework. AngularJS is probably one of the most popular modern-day web frameworks available today. This framework is used for developing mostly Single Page applications. This framework has been developed by a group of developers from Google itself.

Because of the sheer support of Google and ideas from a wide community forum, the framework is always kept up to date. Also, it always incorporates the latest development trends in the market

**Angular JS** is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone.

## Features of AngularJS

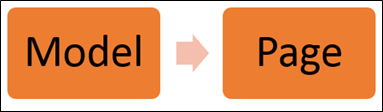
Angular has the following key features which makes it one of the powerful frameworks in the market:

1. **MVC** – The framework is built on the famous concept of MVC (Model-View-Controller). This is a design pattern used in all modern day web applications. This pattern is based on splitting the business logic layer, the data layer, and presentation layer into separate sections. The division into different sections is done so that each one could be managed more easily.
2. **Data Model Binding** – You don’t need to write special code to bind data to the HTML controls. This can be done by Angular by just adding a few snippets of code.
3. **Writing less code** – When carrying out DOM manipulation a lot of JavaScript was required to be written to design any application. But with Angular, you will be amazed with the lesser amount of code you need to write for DOM manipulation.
4. **Unit**[Testing](https://www.guru99.com/software-testing.html)ready – The designers at Google not only developed Angular but also developed a testing framework called “Karma” which helps in designing unit tests for AngularJS applications.

## AngularJS Advantages

Here are the advantages of AngularJS:

* Since it’s an open source framework, you can expect the number of errors or issues to be minimal.
* Two-way binding – Angular.js keeps the data and presentation layer in sync. Now you don’t need to write additional JavaScript code to keep the data in your HTML code and your data later in sync. Angular.js will automatically do this for you. You just need to specify which control is bound to which part of your model.



**Advantages of AngularJS**

* Routing – Angular can take care of routing which means moving from one view to another. This is the key fundamental of single page applications; wherein you can move to different functionalities in your web application based on user interaction but still stay on the same page.
* Angular supports testing, both [Unit Testing](https://www.guru99.com/unit-testing-guide.html), and [Integration Testing](https://www.guru99.com/integration-testing.html).
* It extends HTML by providing its own elements called directives. At a high level, directives are markers on a DOM element (such as an attribute, element name, and comment or CSS class) that tell AngularJS’s HTML compiler to attach a specified behavior to that DOM element. These directives help in extending the functionality of existing HTML elements to give more power to your web application.

**Angular Js is developed by Google.**

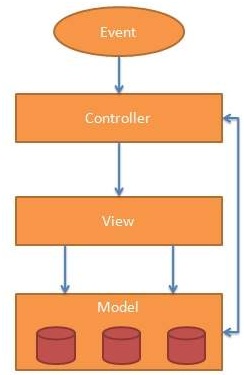
It is an excellent framework for building single phase applications and line of business applications.

**Following are the advantages of AngularJS over other JavaScript frameworks:**

* **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
* **Two way data binding:** AngularJS creates a two way data-binding between the select element and the order Prop model. Order Prop is then used as the input for the order By filter.
* **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
* **Model View Controller:** In Angular JS, it is very easy to develop application in a clean **MVC** way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
* Directives, filters, modules, routes etc.

# AngularJS MVC Architecture

MVC stands for **Model View Controller**. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.



The MVC pattern is made up of the following three parts:

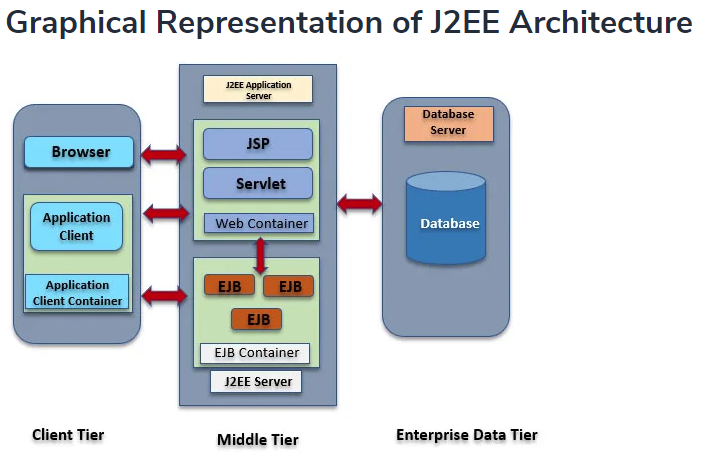
1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

**J2EE Architecture**

**J2EE** can be expanded as **Java 2 Enterprise Edition** that offers a development environment for enterprise application creation and implementation. J2EE Architecture is made up of **three tier**, such as the **client** tier that is used as an interactive medium for the end user or the client & consists of web clients and application clients, the **middle** tier that is used for defining logical functioning units & consist of web components and EJB components, and the **enterprise** data tier that is used for storage purposes in the form of relational database & consists of containers, components and services.

**J2EE Uses Three Tiers:**

* **Client Tier:** Client tier consists of user programs that interact with the user for request and response.
* **Middle Tier:** Middle tier usually contains enterprise beans and [web services](https://www.educba.com/architecture-of-web-services/) that distribute business logic for the applications.
* **Enterprise Data Tier:** Enterprise data is stored in a [relational database](https://www.educba.com/relational-database/). This tier contains containers, components and services.



Usually in J2EE architecture consist of four tiers Client Tier, Web Tier, Enterprise JavaBean (EJB) Tier, and Enterprise Information Tier. The middle tier consists of Web Tier + EJB tier.

#### 1. Client Tier

The client tier consists of programs or applications interact with the user. Usually, they are located in a different machine from the server. Client tier prompts the user inputs into user requests then forwarded to the J2EE server then processed result returned back to the client. A client can be a web browser, standalone application or server that runs on a different machine.

Clients can be classified as a Web Client and Application Client.

##### **Web Client**

Web client consists of dynamic web pages of various mark-up languages that are generated by web components running in **web tier or web browser** which renders pages received from the server. Web clients are also called as thin clients that usually do not perform things like query database, execute business rules. When using thin client-heavy operations are off loaded to enterprise beans executing in the J2EE server.

**Applets:**Web pages received from web tier embedded an [Applet these run](https://www.educba.com/applets-in-java/) on a web browser. Web components are APIs for creating a web client program. Web components enable the user to design cleaner and more modular applications. They provide a way to separate application programming.

Usually in J2EE architecture consist of four tiers Client Tier, Web Tier, Enterprise JavaBean Tier, and Enterprise Information Tier. The middle tier consists of Web Tier + EJB tier.

https://www.educba.com/angular-2-architecture/

# JEE or J2EE Design Patterns

J2EE design patterns are built for the developing the Enterprise Web-based Applications.

J2EE stands for Java 2 Enterprise Edition. The functionality of J2EE is developing and deploying multi-tier web-based enterprise applications. The J2EE platform is the combination of a set of services, application programming interfaces (APIs), and protocols. The J2EE platform adds the capabilities required to provide a complete, stable, secure, and fast Java platform at the enterprise level.

A J2EE module is a software unit that consists of one or more J2EE components for the same container type with one deployment descriptor of that type. Modules can be easily deployed or assembled into J2EE applications.

**J2EE defines four types of modules:**

* Application Client Module
* WEB Module
* Enterprise JavaBeans Module
* Resource Adapter Module

**Application client module** contains the following:

* Class files
* Client deployment descriptor

It is packaged as JAR files with a .jar extension.

**The web module contains** the following:

* JSP (Java Server Pages) files
* Class files for Servlets
* Web deployment descriptor
* GIF (Graphics Interchange Format) and HTML (Hypertext Markup Language) files

These modules are packaged as JAR files with a .war (Web Archive) extension.

The **Enterprise JavaBeans (EJB) module** contains the following:

* Class files for enterprise beans
* An EJB deployment descriptor

These modules are packaged as JAR files with a .jar extension.

The **resource adapter module** contains the following:

* Java interfaces
* Classes
* Native libraries
* Other documentation
* Resource Adapter deployment descriptor

These modules are packaged as JAR files with a .rar (Resource Adapter Archive) extension.

A **J2EE component** is assembled into a J2EE application with its related classes and files. It can also communicate with other components. The J2EE defines the following main components:

* Application clients components.
* Java Servlet and JavaServer Pages technology components.
* Business Components (Enterprise JavaBeans).
* Resource adaptor components.

**types of J2EE clients**

* Applets
* Application clients
* Java Web Start-enabled clients
* Wireless clients

In J2EE , there are mainly three types of design patterns, which are further divided into their sub-parts:

### 1. Presentation Layer Design Pattern

1. [Intercepting Filter Pattern](https://www.javatpoint.com/intercepting-filter-pattern)
2. [Front Controller Pattern](https://www.javatpoint.com/front-controller-pattern)
3. View Helper Pattern
4. Composite View Pattern

### 2. Business Layer Design Pattern

1. Business Delegate Pattern
2. Service Locator Pattern
3. Session Facade Pattern
4. Transfer Object Pattern

### 3. Integration Layer Design Pattern

1. Data Access Object Pattern
2. Web Service Broker Pattern

**Servlet**

**Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

**API**

An **API** is an intermediate software agent that allows dependent applications to communicate with each other. APIs provide a set of protocols, routines, and developer tools enabling software developers to extract and share information and let applications interact in an accessible manner.

## What is a Servlet?

Servlet can be described in many ways, depending on the context.

* Servlet is a technology which is used to create a web application.
* Servlet is an API that provides many interfaces and classes including documentation.
* Servlet is an interface that must be implemented for creating any Servlet.
* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.



**What is a web application?**

A web application is an application accessible from the web. A web application is composed of web components like Servlet, JSP, Filter, etc. and other elements such as HTML, CSS, and JavaScript. The web components typically execute in Web Server and respond to the HTTP request.

### CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. For each request, it starts a new process.



### Disadvantages of CGI

There are many problems in CGI technology:

1. If the number of clients increases, it takes more time for sending the response.
2. For each request, it starts a process, and the web server is limited to start processes.
3. It uses platform dependent language e.g. [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), [perl](https://www.javatpoint.com/perl-tutorial).

**Advantages of Servlet**



There are many advantages of Servlet over CGI. The web container creates threads for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) manages Servlets, so we don't need to worry about the memory leak, [garbage collection](https://www.javatpoint.com/Garbage-Collection), etc.
4. **Secure:** because it uses java language.

**JSP(Java Server Page)**

**JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

**Advantages of JSP over Servlet**

There are many advantages of JSP over the Servlet. They are as follows:

#### 1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

#### 4) Less code than Servlet

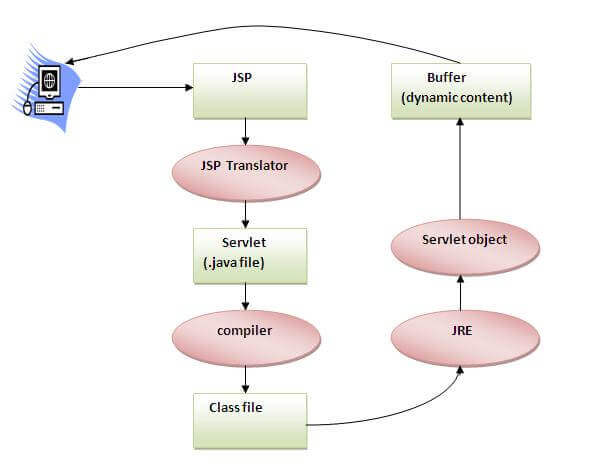
In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

**The Lifecycle of a JSP Page**

The JSP pages follow these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (the classloader loads class file)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( the container invokes jspInit() method).
* Request processing ( the container invokes \_jspService() method).
* Destroy ( the container invokes jspDestroy() method).

#### Note: jspInit(), \_jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

**Creating a simple JSP Page**

To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

**index.jsp**

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page. We will learn scriptlet tag later.

<html>

<body>

<% out.print(2\*5); %>

</body>

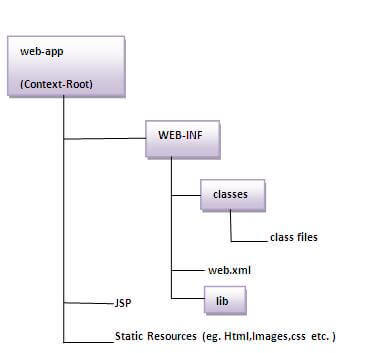
</html

It will print **10** on the browser.

**How to run a simple JSP Page?**

Follow the following steps to execute this JSP page:

* Start the server
* Put the JSP file in a folder and deploy on the server
* Visit the browser by the URL http://localhost:portno/contextRoot/jspfile, for example, http://localhost:8888/myapplication/index.jsp
* Do I need to follow the directory structure to run a simple JSP?
* No, there is no need of directory structure if you don't have class files or TLD files. For example, put JSP files in a folder directly and deploy that folder. It will be running fine. However, if you are using Bean class, Servlet or TLD file, the directory structure is required.
* The Directory structure of JSP
* The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.



# What is EJB

EJB is an acronym for *enterprise java bean*. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

To run EJB application, you need an *application server* (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

1. life cycle management,
2. security,
3. transaction management, and
4. object pooling.

EJB application is deployed on the server, so it is called server side component also.

EJB is like COM (*Component Object Model*) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

## When use Enterprise Java Bean?

1. **Application needs Remote Access**. In other words, it is distributed.
2. **Application needs to be scalable**. EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic**. EJB application is separated from presentation and persistent layer.

## Types of Enterprise Java Bean

There are 3 types of enterprise bean in java.

#### Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

#### Message Driven Bean

Like Session Bean, it contains the business logic but it is invoked by passing message.

#### Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

## Difference between RMI and EJB

Both RMI and EJB, provides services to access an object running in another JVM (known as remote object) from another JVM. The differences between RMI and EJB are given below:

|  |  |
| --- | --- |
| **RMI** | **EJB** |
| In RMI, middleware services such as security, transaction management, object pooling etc. need to be done by the java programmer. | In EJB, middleware services are provided by EJB Container automatically. |
| RMI is not a server-side component. It is not required to be deployed on the server. | EJB is a server-side component, it is required to be deployed on the server. |
| RMI is built on the top of socket programming. | EJB technology is built on the top of RMI. |

## EJB and Webservice

In EJB, bean component and bean client both must be written in java language.

If bean client need to be written in other language such as **.net**, **php** etc, we need to go with **webservices** (SOAP or REST). So EJB with web service will be better option.

## Disadvantages of EJB

1. Requires application server
2. Requires only java client. For other language client, you need to go for webservice.
3. Complex to understand and develop ejb applications.

**RMI** (Remote Method Invocation)

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

**stub**

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

**skeleton**

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



## Understanding requirements for the distributed applications

If any application performs these tasks, it can be distributed application.

.The application need to locate the remote method

1. It need to provide the communication with the remote objects, and
2. The application need to load the class definitions for the objects.

The RMI application have all these features, so it is called the distributed application.

## Java RMI Example

The is given the 6 steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

**JDBC:-**

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

* JDBC-ODBC Bridge Driver,
* Native Driver,
* Network Protocol Driver, and
* Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

* Driver interface
* Connection interface
* Statement interface
* PreparedStatement interface
* CallableStatement interface
* ResultSet interface
* ResultSetMetaData interface
* DatabaseMetaData interface
* RowSet interface

A list of popular *classes* of JDBC API are given below:

* DriverManager class
* Blob class
* Clob class
* Types class

**Why Should We Use JDBC**

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

## What is API

API (Application programming interface) is a document that contains a description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc.