MCS-220: Web Technologies
Guess Paper-1

Q. What is a Web Application and what are Web Server and Client?

Ans. A web application is computer software that can be accessed using any web browser. Usually, the frontend of a web application is created using the scripting languages such as HTML, CSS, and JavaScript, supported by almost all web browsers. In contrast, the backend is created by any of the programming languages such as Java, Python, Php, etc., and databases. Unlike the mobile application, there is no specific tool for developing web applications; we can use any of the supported IDE for developing the web application.

The word Web Development is made up of two words, that is:

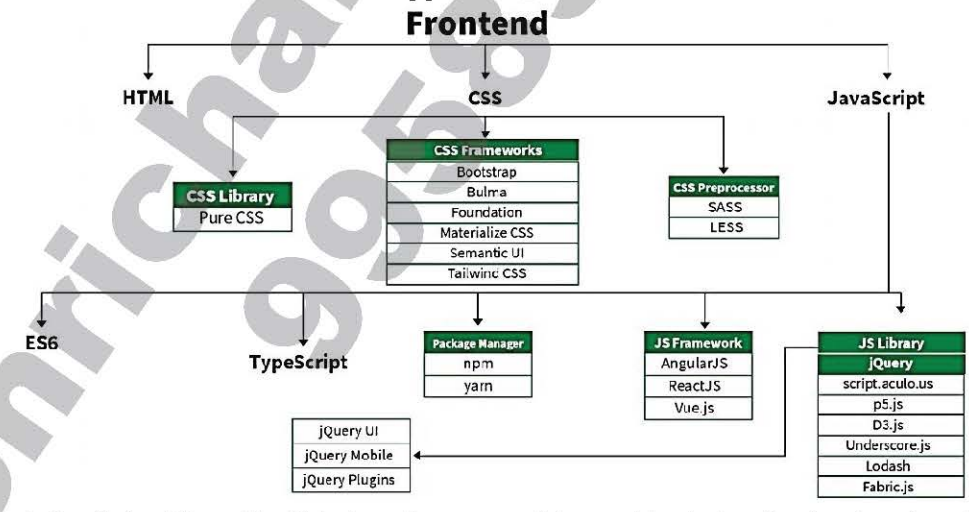
Web: It refers to websites, web pages or anything that works over the internet.

Development: Building the application from scratch.

Web Development can be classified into two ways:

- Frontend Development
- Backend Development

Frontend Development: The part of a website that the user interacts directly is termed as front end. It is also referred to as the 'client side' of the application.



Html: Html stands for HyperText Markup Language. It is used to design the front end portion of web pages using markup language. It acts as a skeleton for a website since it is used to make the structure of a website.

CSS: Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. It is used to style our website.

JavaScript: JavaScript is a scripting language used to provide a dynamic behavior to our website.

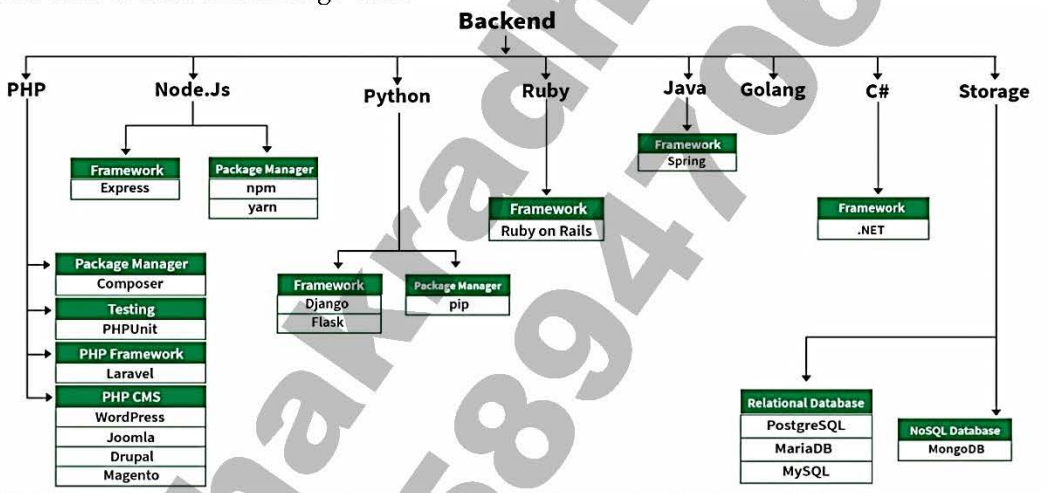
Bootstrap: Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular CSS framework for developing responsive, mobile-first websites. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones).

- Bootstrap 4
- Bootstrap 5

Frontend Frameworks and Libraries:

- AngularJS
- React.js
- VueJS
- jQuery
- Bootstrap
- Material UI
- Tailwind CSS
- jQuery UI
- Some other libraries and frameworks are: Handlebar.js Backbone.js, Ember.js etc.

Backend Development: Backend is the server side of a website. It is the part of the website that users cannot see and interact. It is the portion of software that does not come in direct contact with the users. It is used to store and arrange data.



PHP: PHP is a server-side scripting language designed specifically for web development.

Java: Java is one of the most popular and widely used programming languages. It is highly scalable.

Python: Python is a programming language that lets you work quickly and integrate systems more efficiently.

Node.js: Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside a browser.

Back End Frameworks: The list of back end frameworks are: Express, Django, Rails, Laravel, Spring, etc.

If you develop a web application (independent of the programming language you are using), you typically put your web application on a dedicated server (and not your local computer). The web application runs on the server and people can access it there. The server is either a real machine (with CPU, memory, hard disk, etc.) or a virtual server which is basically a machine which is separated by software into smaller machines.

Instead of running your application directly on a dedicated server, you could also run it in a cloud environment. This cloud environment provides the necessary server for your application. An example for this is the Google App Engine which allows hosting web applications written in different programming languages.

It is possible to use your local computer as a server, but usually you want to have a fixed server which runs 24 hours per day, 7 days per week so that web clients can always reach your server under a pre-defined address.

The web server is a process that handles the client's request and responds. It processes the request made by the client by using the related protocols. The main function of the webserver is to store the request and respond to them with web pages. It is a medium between client and server. For example, Apache is a leading web server.

A client is software that allows users to request and assist them in communicating with the server. The web browsers are the clients in a web application; some leading clients are Google Chrome, Firefox, Safari, Internet Explorer, etc.

Q. Difference between Web Browser and Web Server?

Ans. For International Network communication, we require a web browser and web servers. Web browser and servers play an important role to establish the connection. The client sends requests for web document or service. The message goes from the web browser to the web server is known as an HTTP request. When the web server receives the request, it searches its stores to find the appropriate page. If the web server is able to locate the page, it parcels up to the HTML contained within (using some transport layer protocol), addresses these parcels to the browser (using HTTP), and transmit them back across the network.

If the web server is unable to find the requested page, it sends a page containing an error message (i.e. Error 404 – page not found) and it parcels up to the dispatches that page to the browser. This message received by the web browser by the server is called the HTTP response.

The main differences between the Web browser and web servers are:

Web Browser	Web Server
Web Browser is an Application program that displays a World wide web document. It usually uses the internet service to access the document.	Web server is a program or the computer that provide services to other programs called client.
The Web browser requests the server for the web documents and services.	The Web server accepts, approve and respond to the request made by the web browser for a web document or services.
The web browser act as an interface between the server and the client and displays a web document to the client.	The web server is software or a system which maintains the web applications, generate response and accept clients data.
The web browser sends an HTTP request and gets an HTTP response.	The web server gets HTTP requests and sends HTTP responses.
Doesn't exist any processing model for the web browser.	There exist three types of processing models for web server i.e Process-based, Thread based and Hybrid.
Web browser stores the cookies for different websites.	Web servers provide an area to store and organize the pages of the website.
The web browser is installed on the client computer.	The web server can be a remote machine placed at the other side of your network or even on the other end of the globe, or it is your very own personal computer at home.

Q. Justify Is Java Interpreted or Compiled?

Ans. The Java programming language was developed in the early 1990s by Sun Microsystem. Java is an object-oriented, simple, efficient, robust, and general-purpose programming language. It is primarily used for web-based enterprise applications. It was initially designed for embedded network applications running on different platforms.

When we start learning Java programming, one question arises: whether Java is interpreted or Compiled, or both. Also, this question may be asked by the interviewee in your Java-related interviews. So the answer to this question is Java is both Interpreted and compiled. However, it isn't clear whether Java is compiled or interpreted. It neither generates machine code after compiling a source file nor interpreted the source file to execute code instructions line by line. To answer this question, we need to understand how Java is a platform-independent language? Which means we can write Java code on a platform and can run on other platforms such as hardware operating machine, without making any changes? So, understanding how Java achieves platform independence will provide a complete answer to this question.

Java is completely portable; the same Java code will run identically on different platforms, regardless of hardware compatibility or operating systems.

The Java source code first compiled into a binary byte code using Java compiler, then this byte code runs on the JVM (Java Virtual Machine), which is software based interpreter. So Java is considered as both interpreted and compiled.

The compiled byte code allows JVM to be small and efficient, and fast performing. Also, this byte code provides portability to the Java programming language. It allows executing this code to any JVM that is properly implemented on a machine, regardless of hardware and software components & configurations of the machine. Almost all web browsers contain JVM to execute the Java applet code.

Let's understand it with a simple Java program:

Simple Java Program and It's Working

Create a simple .java file using your favourite text editor:

```
alien@alien-Inspiron-3542:~$ nano Basic.java
```

Put the below code in it:

```
class Basic{
   public static void main(String args[]){
    System.out.println("Hello JavaTpoint");
   }
}
```

And save the file.

Now, check our directory where we have saved the .java file.

```
alien@alien-Inspiron-3542:~$ ls
 Basic.java                              jjjj.jpg
 bin                                     Music
 Desktop                                 opt
 Documents                               Pictures
 Downloads                               Public
 eclipse-workspace                       snap
'gimp selection tools.mp4'               Templates
 google-chrome-stable_current_amd64.deb  Videos
 home.html                               XnViewMP-linux-x64.deb
'Java program'
```

We can execute a Java application by following two steps.

- Compile the Java program
- Execute the application

Compile the Java application: To compile the Java program, execute the below command:

javac Basic.java

The above command will compile the Java program and create a .class file of the Java program that contains the byte code of the Java application.

```
alien@alien-Inspiron-3542:~$ javac Basic.java
alien@alien-Inspiron-3542:~$ ls
 Basic.class                             'Java program'
 Basic.java                              jjjj.jpg
 bin                                     Music
 Desktop                                 opt
 Documents                               Pictures
 Downloads                               Public
 eclipse-workspace                       snap
'gimp selection tools.mp4'               Templates
```

We can see from the above output that after compiling the Java program, a .class file is created. This file contains the byte code of the Java program. We can execute this byte code to any of the machines implementing JVM.

Execute the Application: To execute this application, execute the below command:

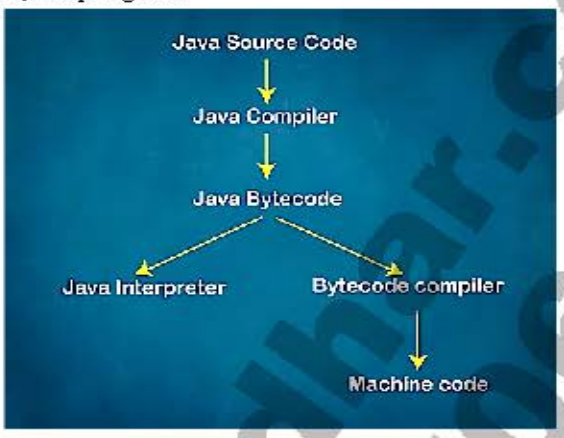
java Basic

The above command will execute the byte code and perform the functionality of the program:

```
alien@alien-Inspiron-3542:~$ java Basic
Hello JavaTpoint
```

From the above output, we have printed a simple statement.

Step by Step execution of the Java program:



Write Java code and save the file with .java

Now, this file will be compiled using the Java compiler, which is javac.

The Java Compiler will compile the Java file and create a .class file having byte code (which is not actually a machine code, unlike the C compiler)

This generated byte code is a non-executable code, and now it needs an interpreter to convert it into machine code. Here the JVM handles it.

Now, JVM will execute this byte code to execute Java byte code on a machine.

Now, our program will perform the functionality and gives the desired output.

Conclusion: Java is considered as both interpreted and compiled. It uses a Java compiler (javac) and JVM (which is actually a software-based interpreter) to execute a Java application on a machine completely.

Q. Comment on Java Big Data Frameworks.

Ans. Big Data is a huge volume of data collection, which is growing exponentially with time. Traditional database management tools cannot handle big data. So, a large size of data is managed and processed using big data tools. There are several Big Data tools available to manage a huge amount of data efficiently.

Nowadays, technologies are evolving very quickly and replacing the previous ones. But, Java technology is available for more than two decades and still a favorable choice for developers. It is used by millions of developers around the Globe and running efficiently on billions of devices. The main reason behind its stability is its updations; Java

is updating itself with time. If we look up the Java version history, we can see that almost every year, a new version of Java is released with some updates and enhancements.

If we look closely at the evolution of technologies, operating systems, and databases over the years, many things are changed. Nowadays, tech developers mainly focusing on the space with big data and IoT.

When it comes to big data, Java is still the backbone for many big data frameworks. Java is naturally good for big data as some major core modules of popular Big data tools are written in Java. Also, one of the major advantages of using big data tools in Java is that some leading big data tools are open-source for Java developers.

Future of Java in Big Data: If we make a statement that "Java is the future of big data", it is worth it. Let's discuss why?

As we have discussed earlier in this topic, the core parts of leading big data tools are written in Java. So, the root of big data is deeply ingrained in Java. Several open-source Java-based communities are contributing to making open-source big data tools.

Nowadays, we can see exponential growth in data. Analyzing such a large amount of data will be continued to increase over the period time as well. One major way to analyze this data is batch data processing, which is mostly done by using open-source tools such as Hadoop and Spark. Both are Java-based tools.

Hadoop is a big name in the big data spectrum. It is one of the prominent tools; for this tool, Java is the language. Hence, it is easy for Java developers to learn it. In fact, for Java developers learning Java-based big data tools is just like learning a new API in Java.

Just like Hadoop, the Pig is also another easy option to learn for Java developers.

Let's discuss some of Java features that allow it to easily deal with big data: Big Data Tools are Accessible for Java Most of the big data tools are accessible for Java so implementation of big data will be the cheapest possible tech stack & flexible.

Java is Type-Safe: It is crucial for data scientists to deal with a large amount of data with the huge set of information being processed. Java is type-safe, so it allows spending less time on unit testing and codebases maintenance.

Java is Scalable: Java is outstanding in terms of scalability. It supports a wide toolkit, a huge community, and cross-platform compatibility, which makes it a perfect choice for designing complex big data infrastructures.

Java is Portable: Java is portable, can be run on any hardware and software platform. This also makes it a good choice for big data.

Java has Garbage Collection.

Java facilitates garbage collection and automatic memory distribution, which also be useful for big data processing.

Java is Secure

Security is one of the main reasons behind the popularity of Java.

Thus, we can say Java has a shining future in the big data processing.

Let's discuss some popular big data frameworks for Java:

1. **Apache Hadoop:** Apache Hadoop is a well-known name in big data management tools. It is an open-source framework provided by Apache Foundation. It efficiently stores and analyzes the huge volume of data. Hadoop is written in Java.

The Apache Hadoop software library allows distributed processing of large data sets across clusters of computers. It is a leading big data tool designed to scale up from single servers to thousands of distributed machines.

Some key features of Hadoop are as following:

- It provides improved authentication while dealing with the HTTP proxy server.
- It provides a compatible Filesystem effort.
- It provides support for POSIX-style file system extended attributes.
- It offers a robust ecosystem to meet the analytical developer's needs.
- It provides flexibility in the data processing.
- It provides faster data processing.
- It is a low cast tool as compared to other big data tools.

2. **Apache Spark:** Apache Spark is similar to the Hadoop Map Reduce framework, but it is getting much more popular than Map Reduce in the big data processing. It is a cluster computing framework that can be run on thousands of machines. Further, it can be run on

distributed computers for analyzing the massive datasets across these devices. The Spark works on the concept of RDD (Resilient Distributed Dataset).

Park performs large ETL: (extract, transform, and load) operations easily. Furthermore, it performs predictive analytics and reporting applications operations over large datasets. Apache Spark performs the following operations:

- It loads data into the RDD (Resilient Distributed Dataset).
- It transforms data to make it compatible for handling operations.
- It caches the reusable data across different sessions.
- It performs some predefined or custom operations on the data.

The Spark is written in Scala programming language, which inherently is written in Java. Hence, indirectly Java is the base of the Apache Spark stack and is fully supported by all its products. Its stack has an extensive Java API. Thus, The Apache is an easily adopted big data framework for the Java developer.

The following are some of Spark's API that can be easily understood and used by the Java developer:
The core RDD frameworks, along with their functions

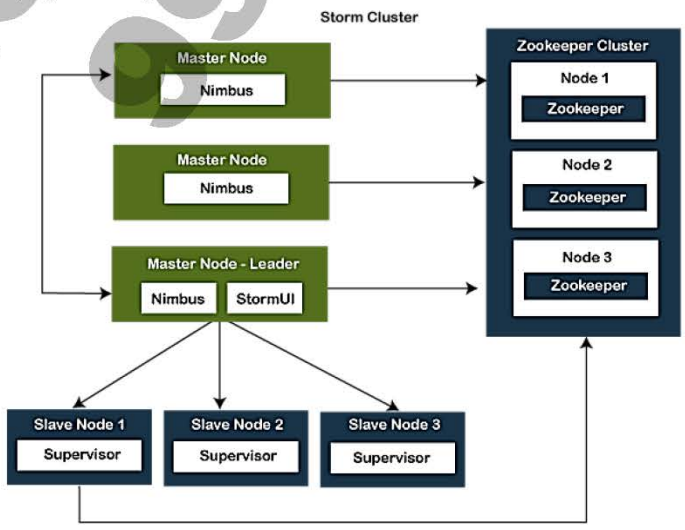
- Spark SQL code
- Spark Streaming code
- Spark MLlib algorithm
- Spark GraphX library

(3) Apache Storm: The storm is a free and open-source distributed real-time computational system for big data provided by the Apache foundation. It is an efficient tool for big data that makes it is easy to reliably process unbounded streams of data with real-time processing. It is an easy tool that can be used with any programming language.

It facilitates real-time data processing, machine learning, continuous computation, ETL, distributed RPC, and more. It is a fast tool that performs over a million tuples processing per second per node. It is a distributed, scalable, fault-tolerant tool and easy to set up tool. It integrates with the queueing and available database technologies.

Its architecture has two main components:

- Master node (Nimbus)
- Worker nodes (Supervisor)

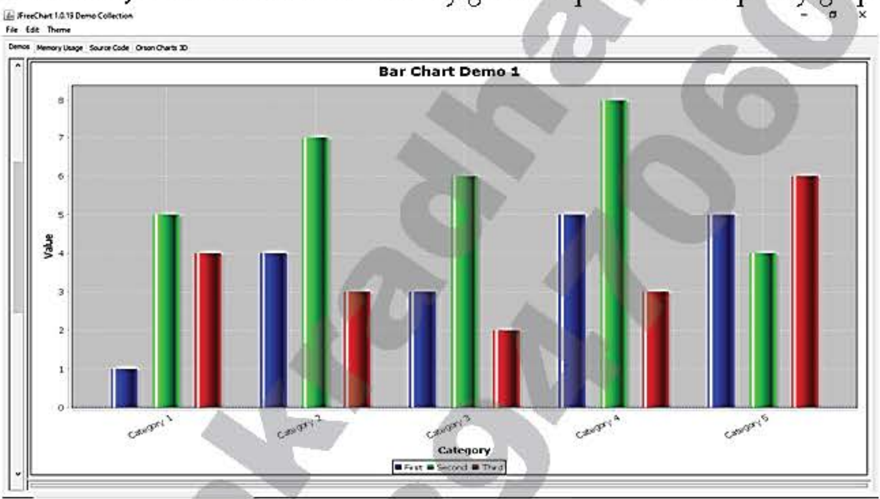


Some key features of Apache Storm are as following:

- User-friendly
- Free and open source
- Ideal for small to large scale implementations
- Highly fault-tolerant
- Reliable
- Superfast
- Real-time processing
- Scalable
- Distributed
- Facilitates dynamic load balancing and optimization using operational intelligence

(4) Java J Free Chart: Data visualization is also an integral task in big data analysis. As big data deals with a huge amount of datasets, it is also necessary to represent and look out for the raw data. When data is represented in a chart, it becomes easy to analyze data.

JFreechart is one of the leading tools available for the visualization of data. It is an open-source tool with built-in libraries in Java to allow a user to easily generate professional-quality graphs and charts.



We can create different types of visuals such as pie charts, bar charts (with an optional 3D effect), scatter plots, Gantt charts, line charts, time series charts, and more using JFreeChart.

JFreeChart library provides plug-in support in various IDE's such as Eclipse, Netbeans, and more. It provides several choices to add charts in our application.

(5) Apache Mahout: Apache Mahout is also an open-source big data tool, which provides a Java ML library. It is an Apache Software Foundation product designed for Machine Learning. It enables machines learning with queries without being overly programmed. It supports scalable machine learning algorithms, extracts recommendations and relationships from data sets in a convenient way.

The Mahout runs on Hadoop, using the MapReduce paradigm. With its data science tools, Mahout supports the following features:

Recommendations

- Clustering
- Classifications
- Collaborative filtering
- Frequent Itemset Mining

Mahout's algorithms run on Hadoop. Thus, it performs well in a distributed environment. Furthermore, it provides inbuilt Map Reduce implementations of several ML algorithms.

(6) Deeplearning4j: Deeplearning4j is also an important big data tool. It is a Java library that is useful for building different types of neural networks. It can also be integrated with Apache Spark on the big data stack can even run on GPUs. It provides several Java libraries with lots of built-in algorithms for deep learning and casting in Java. Additionally, it also has large communities and documentation.
Some useful features of Deeplearning4j are as following:

- Distributed GPUs and CPUs
- Java, Python, and Python APIs
- Supports micro-service architecture
- Scalable on Hadoop
- GPU support for to scale up on AWS

(7) HPCC: HPCC is also the most widely used big data tool. It is developed by LexisNexis Risk Solution. HPCC system delivers on a single platform and provides an end-to-end data lake management solution. The HPCC provides an easy way to develop the data application. It is a simple, fast, accurate, and cost-effective tool. It is primarily developed for high-speed data engineering.

Some useful features of HPCC are as following:

- It is a highly efficient big data tool that accomplishes big data tasks with minimal code.
- It provides faster big data processing with high redundancy and availability.
- It can be used for complex data processing on a cluster of data
- It provides a graphical IDE for simplifying the development, testing, and debugging.
- It provides optimized code for parallel processing.
- It has enhanced scalability and performance.
- The ECL code compiles into optimized C++, and it can also extend using C++ libraries.

(8) Qubole: The Qubole Data is an open-source autonomous Big Data management Tool. It is a self-managed and optimized tool that allows the data team to focus on the business outcome.

Some useful features of Qubole are as following:

- It provides a single Platform for every use case
- It is an Open-source big data tool.
- It supports engines optimized for the Cloud.
- It provides comprehensive Security, Governance, and Compliance.
- It provides actionable Alerts, Insights, and Recommendations to optimize reliability, performance, and costs.
- It automatically enacts policies to avoid performing repetitive manual actions.

(9) Couch DB: The CouchD Bis also a vital tool for handling big data. It is used to store data in JSON documents that are accessible from the web or query using JavaScript. It provides distributed scaling with fault-tolerant storage. It defines Couch Replication Protocol to access data.

Some useful features of CouchDB are as following:

CouchDb works like other databases; it is a single node database.

- It allows running a single logical database server on multiple servers.
- It uses ubiquitous HTTP protocol and JSON data format.
- It provides easy replication of a database across multiple server instances.
- It has an easy interface for the document, provides insertion, updates, retrieval, and deletion operations.
- It provides a JSON-based document format that can be translatable across different languages.

(10) Apache Cassandra: The Cassandra:database is a widely used big data tool. It provides effective management of large amounts of data.

Some useful features of Apache Cassandra are as following:

- I provide support for replicating across multiple data centers.
- It automatically replicates data to multiple nodes for fault-tolerance
- It is the most suitable tool for applications that can't afford to lose data, even when an entire data center is down.
- It provides support for third-party contracts and services.

Q. Describe briefly J2EE Development Frameworks?

Ans. Java 2 Enterprise Edition has excelled at standardizing many important middleware concepts. For example, J2EE provides a standard interface for distributed transaction management, directory services, and messaging. In addition, Java 2 Standard Edition (J2SE), which underpins J2EE, provides a largely successful standard for Java interaction with relational databases. However, as the "J2EE's Lack of Application Programming Support" sidebar explains, the platform has failed to deliver a satisfactory application programming model. Sun Microsystems and the large application server vendors have traditionally responded to this problem by advocating development tools as a way to hide J2EE's complexity.

However, tools for managing J2EE artifacts aren't nearly as good as tools for working with the Java language, with their sophisticated refactoring capabilities, and J2EE tool support is generally inferior to that of the Microsoft .NET platform. Many J2EE tools are themselves complex, as is the code they generate. Many in the open source community, especially smaller vendors, have chosen the alternative of developing frameworks designed to simplify the experience of building J2EE applications. Popular frameworks such as Struts, Hibernate, and the Spring Framework play an important role in many of today's J2EE development projects.

Why use a framework? A software framework is a set of classes that make up a reusable design for an application or, more commonly, one tier of an application. Whereas application code calls a class library to perform services, a framework calls application code and thus manages the flow of control. This is often referred to as the Hollywood principle: "Don't call us, we'll call you."

The application developer writes code that the framework will then call at runtime.

Designing a framework for use in a wide variety of unknown contexts is challenging. However, the framework approach is well adapted to the complexities of J2EE development because it can provide a simple, easy-to-use model for application programmers.

Using a well-designed open source framework offers many advantages:

- With a good framework, developers write only the code they need to write; they don't get bogged down working directly with low-level infrastructure APIs. This is the key value proposition.

- A well-designed framework can provide structure and consistency to an application. The structure will be clear to additional developers joining the project.
- An easy-to-follow framework can promote best practice through examples and documentation.
- Successful open source frameworks are better tested than in house code.
- Frameworks usually become popular only if they have something to offer.

In-house frameworks are often mandated, while a J2EE project is likely to adopt an open source framework only if it delivers clear benefits.

J2EE itself defines several frameworks. For example, an Enterprise JavaBeans (EJB) container or Servlet engine relies on the Hollywood principle, with the J2EE runtime instantiating and invoking managed objects. Open source Web application frameworks such as Struts add their own framework over the standard Servlet framework. The emphasis here is on frameworks above J2EE, which provide a simpler programming model or other benefits. Open Source Frameworks Emerge Most large J2EE projects have traditionally used in-house frameworks to hide the platform's complexity. Only recently has a consensus emerged about the generic problems that require a generic solution, and around particular frameworks that provide good generic solutions. There is now a clear trend for frameworks to "standardize" more of the infrastructure that formerly was developed on a per-project basis. One reason for J2EE frameworks' sudden popularity is the platform's increased maturity. Developers now recognize areas in which the standard APIs are deficient and know from experience how difficult it is to write a good framework to fill the gap. In addition, many high-quality frameworks are now available that offer outstanding documentation and the support of a focused development team, without imposing licensing fees. Struts the trend toward open source frameworks began with Web applications. In 1999-2000, developers realized the deficiencies of the Java Server Pages "Model 1" approach, in which JSP templates handled incoming requests as well as static template data. This meant that JSPs often contained both business logic and complex HTML or other markup. With no standard framework in place or J2EE specification support, developers responded with their own Front Controller implementations. These moved business logic to Java classes, thereby eliminating the need to maintain such hybrid artifacts.

The Front Controller pattern is often referred to as Web MVC after the classic Model View Controller architecture pattern that is common to GUI development in object-oriented languages. (The name is somewhat misleading in that Web MVC views must pull information from the model, whereas in classic MVC, the model pushes events to views.) Initial Front Controller implementations varied greatly in quality. The Apache Software Foundation's release of Struts (http://struts.apache.org) in 2001-2002 changed all this. While not an ideal Web MVC framework, Struts worked well enough to quickly become the de facto standard. Struts demonstrated all the benefits of open source frameworks such as ease of recruiting personnel familiar with the structure it imposed. By late 2002, it was the natural choice for most J2EE Web applications, and every serious J2EE Web developer was familiar with it. The near-universal adoption of Struts commoditized an important chunk of the J2EE architectural stack. Even conservative organizations accepted its use in a prominent part of their software infrastructure and agreed to the Apache license's terms. Hibernate The next domino to fall was persistence. J2EE "out of the box" provided two means for accessing persistent stores—most often, relational databases: JDBC, the J2SE standard API for relational database management system access; and entity beans, an EJB component type dedicated to modeling a persistent entity. JDBC's error-prone programming model inhibited object-oriented design by forcing developers to work with relational concepts in Java code. Entity beans, despite hype from Sun and major J2EE vendors, likewise proved to be cumbersome: Initially, the technology was severely underspecified, not even taking into account management of relationships between persistent objects; it made applications difficult to test; and it offered an inadequate query language.

By 2003, developers largely ignored entity beans despite enhancements in EJB 2.0 and 2.1. Early efforts. Solutions to the persistence problem came in the form of object-relational mapping (ORM), which provides transparent persistence for plain old Java objects (POJO), a concept described in the

sidebar, "The Noninvasive Framework: Power to the POJO." Though not unique to Java, ORM is especially popular in the Java community—compared, for example, to .NET developers, who seem to regard it with suspicion. Commercial ORM tools such as Oracle's TopLink (www.oracle.com/technology/products/ias/toplink) were well established by the late 1990s, but only a minority of projects used them because they were expensive, complex, and appeared to conflict with the Sun sanctioned entity bean standard.

Nevertheless, they usually achieved better results in practice than JDBC or entity beans, thus proving the case for POJO persistence. Java Data Objects, which appeared as a Java Community Process (www. jcp.org) specification in 2001, offered generic POJO persistence to any persistent store (although implementations typically provided their best support for relational databases). However, Sun's lukewarm attitude toward JDO, coupled with J2EE vendors' lack of interest in POJO persistence at that time, prevented the technology from achieving popularity. Hibernate arrives. Radical change came in 2002 for two reasons. First was widespread realization that entity beans had failed in practice, and that developers should ignore that part of the J2EE specifications. By retarding rather than advancing the progress of ORM in Java, entity beans remain a prime example of how poor specifications can stifle development of superior technologies.

The second factor was the arrival of Hibernate (www.hibernate.org), the first popular, fully featured open source ORM solution. Hibernate offered fewer features than TopLink but delivered a robust implementation of the most desirable ones, and its focused development team aggressively sought improvements. Hibernate wasn't particularly innovative, building on the extensive understanding of ORM, but it offered a more intuitive programming model than existing competitors and removed at one stroke the cost and ease-of-use barriers to ORM. Around the same time, new commercial products offered highly efficient implementations of the JDO specification that targeted relational databases, giving developers a rich choice. Meanwhile, Top Link remained a good option, with its license becoming friendlier to developers.

ORM triumphs. Together, all these factors converged to make ORM the norm rather than the exception by 2003- 2004. Although some projects still built their own persistence frameworks, the existence of Hibernate, TopLink, and leading JDO implementations made this extremely difficult undertaking unnecessary and indefensible. Another part of the application stack was now within the domain of popular frameworks, yet large gaps remained. For example, a typical Web application using Struts and Hibernate still lacked framework support for business logic. Although the J2EE specifications address some of these issues, primarily through EJB, they don't provide an adequate application programming model. Spring J2EE frameworks have inexorably moved into application frameworks, which aim to provide consistent programming in all tiers and thereby integrate the application stack. The Spring Framework (www.springframework. org) is the dominant product in this space, with adoption comparable to that of Hibernate. Spring essentially combines inversion of control (IoC) and aspect-oriented programming (AOP)—both described in the sidebar,

"The Noninvasive Framework: Power to the POJO"—with a service abstraction, to provide a programming model in which application code is implemented in POJOs that are largely decoupled from the J2EE environment (and thus reusable in various environments). Spring also provides an alternative to EJB in many applications—for example, delivering declarative transaction management to any POJO. The Spring approach has proven to deliver excellent results in many kinds of projects, from small Web applications to large enterprise applications. Other products in the same space include Hive Mind (http://jakarta. apache.org/hive mind), which is conceptually similar to Spring but has a somewhat different take on IoC, and Nano Container which combines the Pico Container IoC container with services. Collectively, these products are referred to as lightweight containers to distinguish them from traditional J2EE approaches. By decoupling a POJO model from J2EE APIs, which are hard to stub at test time, lightweight containers greatly simplify unit testing. It's possible to unit test in a plain J Unit environment, without any need to deploy code to an application server or to simulate an application server environment. Given the increased—and deserved—popularity of test-driven development, this has been a major factor in lightweight frameworks' popularity Growing recognition and use of J2EE development frameworks is measurably reducing cost in many projects,

as well as delivering better speed to market and higher maintainability. Today's best frameworks offer excellent quality, solid documentation, and numerous books and articles to support their use. Nevertheless, two areas in particular seem set for uncertainty in the J2EE space: the conflict between J2EE "standards" and open source innovation, and the growing importance of AOP. The open source versus standards conflict looms in two areas. In the presentation tier, Java Server Faces (JSF), backed by Sun and some of the largest vendors, competes with entrenched open source solutions such as Struts. In the middle tier, EJB 3.0 offers a dependency injection capability reminiscent of a subset of spring's capabilities, mixed with liberal use of J2SE 5.0 annotations. In both areas, innovation has traditionally come from open source rather than specifications. However, JSF is somewhat indebted to ASP.NET, while the open source Tapestry project (http:// jakarta.apache.org/tapestry)—a mature implementation of many of the same concepts—owes much to Apple's commercial Web Objects. Likewise, EJB 3.0 seems to be attempting to standardize dependency injection, though it's unclear what benefit this brings—especially if it results in the loss of important features, which seems inevitable. EJB 3.0 also attempts a new departure in entering the application programming space: an area in which the J2EE specifications haven't shone to date. Meanwhile, AOP's importance is steadily increasing within the J2EE community. While adoption isn't yet widespread, certain uses of AOP, such as declarative transaction management, are already popular. Solutions including Spring and dynaop (http://dynaop. dev.java.net), which offer what might be called "AOP with training wheels," help to increase awareness of AOP.

Fullblown AOP technologies such as AspectJ will likely experience wider adoption in the next few years as well. Significantly, the Java Community Process shows no sign of any move to standardize AOP, although JBoss (www.jboss.com)—which is overtly committed to working through the JCP with the EJB 3.0 specification—is vigorously pursuing proprietary AOP technology. T he next-generation J2EE specifications as a whole are embracing a simpler, POJO programming model, similar to that already offered by combinations such as Spring and Hibernate. J2EE developers are sure to benefit from recent trends, which are driven more by practical experience than by marketing hype. This is a welcome change from the platform's early days, when results often failed to live up to the promise held out by vendors.

MCS-220: Web Technologies
Guess Paper-2

Q. List Various Open Source J2EE Frameworks

Ans. Spring: Spring is a layered Java/J2EE application framework, based on code published in Expert One-on-One J2EE Design and Development

Dinamica Framework: This framework is based on the MVC architecture (model-view-controller), but in contrast with other MVC frameworks, most of its parts are highly reusable, meaning that your programming effort is minimized whenever possible, avoiding unnecessary programming. Many tasks are accomplished by simply configuring some XML files and templates. Presentation templates are completely separated from logic. Dinamica does not use any template language nor JSP pages technology, just plain text files with some special markers that will be replaced by the actual data.

Jeenius: Jeenius is a framework to simplify the creation of J2EE applications. It has a strong focus on building web-based applications.

Expresso: Expresso is an open standards-based, enterprise-strength J2EE architectural framework for developing database-driven web applications based on open standards.

jGuard: Build on top of JAAS, for J2EE web applications. his goal is to provide for web app developers, an easy way to manage authentication and authorizations.

Mdarad-toolbox: MDARAD is an open source modeling tool that generates web applications under simple servlet containers or, alternatively, in J2EE containers. It uses the dataïsl and™ (www.dataisland.org) platform independent modeling (PIM) specifications. The generated application is packaged and built in order to facilitate user customizations to the generated code and implements many of the J2EE core design patterns.

Some of the generated features are:

- Struts JSP Tiles and actions to manage web layer *
- Persistence Facades using Hibernate
- XML Serialization Services *
- ReST and SOAP Web Services *
- Lucene Full Text Search Facades *

Junit Tests: The MDARAD J2EE webapp project is divided in three components:

Mdarad-Framework: collection of interfaces, utilities and abstract classes used by the generated web application.

* **Andromda-Webapp:** the cartridge used to generate the application. *

Mdarad-Genapp: the development environment generator and manager. This is where you generate your project and control what will be generated or not. MDARAD also has a vision to allow ontology communications based on web semantics and ontology alignment.

Glass Fish: Glass Fish is the name for the open source development project for building a Java EE 5 application server. It is based on the source code for Sun Java System Application Server PE 9 donated by Sun Microsystems and Top Link persistence code donated by Oracle. This project provides a structured process for developing a high quality application server that makes new features available faster than ever before. It is the response to Java developers who want access to the source code and the ability to contribute to the development of Sun's next generation application server which is based on Glass Fish. This project is designed to encourage communication between Sun and Oracle engineers and the community and will enable all developers to participate in the application server development process.

Jdon: Jdon Framework is the combination of Domain Drive Development (Ruby on Rails that is simple and rapid but not java language) and Ioc/AOP components (Spring that is very agility but has complicated XML configuration). Jdon Framework combines with rapidity and agility for your web application architecture and seamlessly supports two service architecture types: Java Beans(POJO) or

EJB. It can easily migrate POJO to EJB, or integrate legacy EJB systems. Jdon Framework Features: 1. All components can be replaced, even the framework itself. 2. Support the default implementations for CRUD and Master-details display. 3. Improve performance of the framework and its applications by plug-in cache. 4. Provide the session function in micro container, application component can be stateful. Jdon Framework = Ioc(pico) + AOP(simple) + CRUD + Cache

OpenCore: Midleware abstraction layer providing foundation for rapid development and smooth integration of open subsystems. *

- Data model, persistence, business logic, web user interface and desktop application (thick client) framework usable as simple Java classes (POJOs) or within J2EE application server
- Persistence layer architecture and implementation ensuring optimistic locking preventing concurrent data modification and unique data id generation and timestamp maintenance directly by a database
- Default persistence layer implementation using pure JDBC for high performance, low overhead and easy portability.
- Database schema definition, maintenance and versioning
- Database connectivity, connection pooling, transaction management *
- Framework and patterns for data access and modification *
- Utilities for context propagation, asynchronous executions, encryption, transactional file manipulation *
- Web user interface framework, page inheritance, layout definition, security
- Desktop application (Thick client) framework with default implementation using SWT *
- Database support for DB2, HSQLDB, MaxDB, MS SQL Server, MySQL, PostgreSQL, Oracle, SAP DB, Sybase ASE *
- Connection pool support for Commons-DBCP, C3P0, Proxool, XAPool and J2EE data sources * Transaction manager support for JOTM, J2EE transaction managers and custom in-JVM transaction manager
- J2EE application server support for JBoss, JOnAS, WebLogic, WebSphere

AppFuse: AppFuse is an open source project/application that uses best-of-breed Java open source tools to help you develop web applications quickly and efficiently. Not only does it provide documentation about developing lightweight POJO-based applications, but it comes out of the box with features that many applications need, including features for authentication and authorization, user management, Remember Me, password hint, signup, SSL switching, e-mail, URL rewriting, skin ability/page decoration/templated-layout and file upload. This out-of-the-box functionality is one of the main features in AppFuse that separates it from the other \"CRUD Generation\" frameworks, including Ruby on Rails, Trails, and Grails.

J2EE Pattern Oriented Framework (Jt 1.2): Jt1.2 has been released. It is a lightweight pattern oriented framework for the rapid implementation of J2EE applications. Jt has been utilized in several large mission critical systems. Jt implements many well-known patterns including Data Access Objects (DAO). It features messages, attributes, logging/debugging capabilities, resource loading, etc. This framework supports several J2EE technologies including Java Server Pages (JSPs), JDBC, EJBs, JavaMail, XML and Web Services. Jt1.2 features support for J2EE Enterprise Java Beans (EJB) via Jt Adapters. EJB clients are able to gain transparent access to remote framework objects. This simplifies the development of EJB applications: no need to deal with the complexities of EJB application development (deployment descriptors, Home/Remote interfaces, etc). An implementation of the J2EE Service Locator pattern is also provided.

Additional features include:

- *It is a pattern oriented framework.
- It implements many well-known design patterns.

This includes Data Access Objects (DAO), adapters for several J2EE API, etc.

It implements a messaging pattern/API: Framework objects are able to interchange information and perform computations by sending, receiving and processing messages. The messaging paradigm

provides additional encapsulation and software simplicity. The messaging API implemented by the Jt **Framework is very simple:** very few calls are required to create, manipulate and destroy messages and objects. On the other hand, this API is very powerful.

- Lightweight/fast framework for rapid development.
- The framework can be easily extended by adding additional. It adapters and helpers.
- Support for the JDBC API via a JDBC adapter.
- Java Mail API support via the implementation of a Java Mail adapter
- Web Services support via the implementation of a Web Services adapter. It messaging API greatly simplifies the development of web services.
- Easy customization of framework applications. This is done via resource files: object attributes can be automatically loaded from a resource file.
- Java Server Pages (JSP) support. * Support for the XML API via XML helpers. Framework objects can be converted to/from the XML format.
- Built-in logging/debugging capabilities. Messages between framework objects are automatically logged. This simplifies the debugging and testing tasks.
- Built-in testing capabilities.
- The It Framework provides a consistent way of handling and logging application errors and exceptions. *
- Proven technology. It framework has been used for the development of several large enterprise applications.
- Cross-platform, implemented using Java TM technology.
- Runs on any J2EE 1.4 compatible application server. For additional information please refer to http://jt.dev.java.net

Atomikos Transactions Essentials: Atomikos Transactions Essentials™ is a premium transaction manager for automatic cancelation of problematic transactions, across disparate data sources or back-end systems. Unlike CICS™ (IBM, mainframe) or Tuxedo™ (Bea, workstations), this product is optimized for easy integration into Java-based solutions: it has a very small memory footprint and seamless compatibility with virtually every Java application. Integration into your application can be as simple as one line of code.

OpenEnterpriseX: OpenEnterpriseX is a free, open source, comprehensive and standards-based Java(TM)/J2EE(TM) development suite distribution for building enterprise applications. It is based on leading Open Source web servers, containers, frameworks, utilities, database and integrated development editor. The distribution will not in any way hinder your usage of any of the open source packages individually. Instead, it serves to help you use the packages easily.

JAG Java Application Generator: JAG is an application that creates complete, working J2EE applications. It is intended to alleviate much of the repetitive work involved in creating such applications, while providing a means of quality assurance that the applications created will be of consistent quality. The projects generated by JAG have the following features: * The generated applications build with Apache Ant * EJB and Hibernate features such as relations * J2EE 'BluePrint' patterns such as Session Facade, Service Locator, Business Delegate, Fast Lane Reader and Value Objects. * A pluggable service tier: ServiceLocator or Spring Framework. * A pluggable business/persistence tier: EJB2/3 or Hibernate 2/3 * Tapestry 4 presentation-layer for a component based web tier using java 5 annotations * Webservices using XFire or Axis * Acegi security login * JasperReports PDF reports * A presentation layer (web application) that takes advantage of the latest features of Struts * Extensive use of Java 5 Annotations or XDoclet in the generated application

jWebApp: jWebApp is a J2EE Servlet-based Model-View-Controller framework that allows you to use anything you like for the Model and View. jWebApp allows independence in model and view technologies. You can use any model/business layer technologies, any database-access technologies, any web-authoring technologies, and plain old HTML and HTML forms.

ActiveInsight: Active Insight is an open source ESP / CEP framework that offers real-time, value-based detection and reaction to events and patterns. It offers a distributed (cloud ready) Event Stream

Processing framework used for processing single and aggregated events (Complex Event Processing). The Active Insight framework offers a pattern detection engine and distributed caching modules designed to be embedded in 3rd party applications (ISV's) or deployed as a standalone application alongside other back-end systems.

Doff: Doff is a lightweight Open Source J2EE toolkit that provides mapping between actions (servlets) and URLs without writing XML or any configuration file. Doff uses Java 5 annotations and offers many features.

Seam Framework: Seam is an open source development platform for building rich Internet applications in Java. Seam integrates technologies such as Asynchronous JavaScript and XML (AJAX), JavaServer Faces (JSF), Java Persistence (JPA), Enterprise Java Beans (EJB 3.0) and Business Process Management (BPM) into a unified full-stack solution, complete with sophisticated tooling.

Jspresso: Jspresso is a new generation framework to easily build professional quality java distributed desktop applications. Jspresso dramatically reduces the development cycles needed to get your business application up and running while not sacrificing quality, robustness and performance. Jspresso is not another webapp framework. Jspresso based applications offer the exact same ergonomics as desktop applications while keeping an N-tier centralized architecture. Jspresso covers extensively the whole software architecture and relieves the developer from all the plumbing by solving most of the technical concerns. Jspresso philosophy extends the \"Convention over configuration\" paradigm with a descriptive strategy made of assembling built-in descriptors (java beans). You describe what you want to achieve and not how you want to achieve it.

Nabucco Framework: The NABUCCO Framework is a modern approach for developing Java EE software in a model-driven way using a component model. The NABUCCO Framework consists of a MDA approach with a separate DSL, as well as components based thereon, for the development of products and customized applications. Objectives of the Framework are: - Creation of reusable components - Complete multi-client capability of all components - Depiction of recurring functions (Best Practices) - Independence from tools and special technologies - Supported by UML as well as textual DSLs - Fast and simple development.

Q. Explain briefly Spring Framework?

Ans. spring makes programming Java quicker, easier, and safer for everybody. Spring's focus on speed, simplicity, and productivity has made it the world's most popular Java framework.

Spring is Everywhere: Spring's flexible libraries are trusted by developers all over the world. Spring delivers delightful experiences to millions of end-users every day—whether that's streaming TV, online shopping, or countless other innovative solutions. Spring also has contributions from all the big names in tech, including Alibaba, Amazon, Google, Microsoft, and more.

Spring is flexible: Spring's flexible and comprehensive set of extensions and third-party libraries let developers build almost any application imaginable. At its core, Spring Framework's Inversion of Control (IoC) and Dependency Injection (DI) features provide the foundation for a wide-ranging set of features and functionality. Whether you're building secure, reactive, cloud-based microservices for the web, or complex streaming data flows for the enterprise, Spring has the tools to help.

Spring is productive: Spring Boot transforms how you approach Java programming tasks, radically streamlining your experience. Spring Boot combines necessities such as an application context and an auto-configured, embedded web server to make micro service development a cinch. To go even faster, you can combine Spring Boot with Spring Cloud's rich set of supporting libraries, servers, patterns, and templates, to safely deploy entire micro services-based architectures into the cloud, in record time.

Spring is fast: Our engineers care deeply about performance. With Spring, you'll notice fast startup, fast shutdown, and optimized execution, by default. Increasingly, Spring projects also support the reactive (nonblocking) programming model for even greater efficiency. Developer productivity is Spring's superpower. Spring Boot helps developers build applications with ease and with far less toil

than other competing paradigms. Embedded web servers, auto-configuration, and "fat jars" help you get started quickly, and innovations like Live Reload in Spring Dev Tools mean developers can iterate faster than ever before. You can even start a new Spring project in seconds, with the Spring Initializr at start.spring.io.

Spring is secure: Spring has a proven track record of dealing with security issues quickly and responsibly. The Spring committers work with security professionals to patch and test any reported vulnerabilities. Third-party dependencies are also monitored closely, and regular updates are issued to help keep your data and applications as safe as possible. In addition, Spring Security makes it easier for you to integrate with industry-standard security schemes and deliver trustworthy solutions that are secure by default.

Spring is supportive: The Spring community is enormous, global, diverse, and spans folks of all ages and capabilities, from complete beginners to seasoned pros. No matter where you are on your journey, you can find the support and resources you need to get you to the next level: quick starts, guides & tutorials, videos, meetups, support, or even formal training and certification.

What can Spring do:

Microservices: Quickly deliver production grade features with independently evolvable micro services.

Reactive: Spring's asynchronous, no blocking architecture means you can get more from your computing resources.

Cloud: Your code, any cloud—we've got you covered. Connect and scale your services, whatever your platform.

Web apps: Frameworks for fast, secure, and responsive web applications connected to any data store.

Serverless: The ultimate flexibility. Scale up on demand and scale to zero when there's no demand.

Event Driven: Integrate with your enterprise. React to business events. Act on your streaming data in real time.

Batch: Automated tasks. Offline processing of data at a time to suit you.

Spring is widely used for creating scalable applications. For web applications Spring provides Spring MVC which is a widely used module of spring which is used to create scalable web applications.

But main disadvantage of spring projects is that configuration is really time-consuming and can be a bit overwhelming for the new developers. Making the application production-ready takes some time if you are new to the spring.

Solution to this is Spring Boot. Spring Boot is built on the top of the spring and contains all the features of spring. And is becoming favorite of developer's these days because of it's a rapid production-ready environment which enables the developers to directly focus on the logic instead of struggling with the configuration and set up.

Spring Boot is a micro service-based framework and making a production-ready application in it takes very less time.

Prerequisite for Spring Boot is the basic knowledge Spring framework.

For revising the concepts of spring framework read this article.

Evolution of Spring Boot: Spring Boot came into existence when in October 2012, a client, Mike Young strom made a Jira request asking for bootstrapping the spring framework so that it can be quickly started. And hence in early 2013, Spring Boot was made.

In April 2014, Spring Boot 1.0 was created followed by various versions.

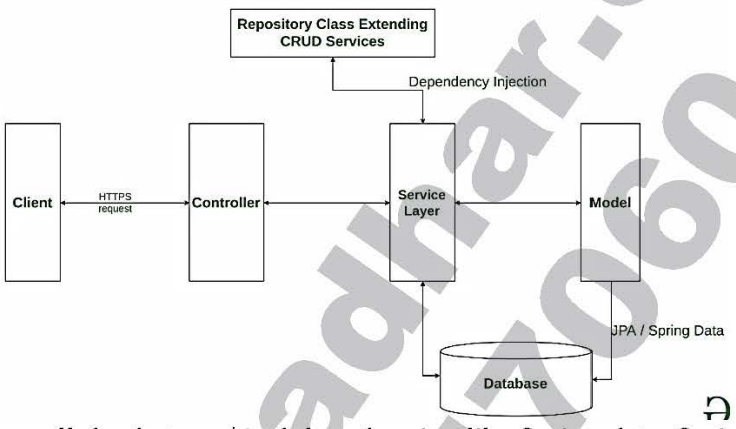
Spring Boot 1.1 on June 2014,

- 1.2 in March 2015,
- 1.3 in December 2016,
- 1.4 in January 2017 and
- Spring Boot 1.5 on February 2017.
- Spring Boot Architecture

To understand the architecture of Spring Boot, let us first see different layers and classes present in it.

Layers in Spring Boot: There are four main layers in Spring Boot:
Presentation Layer: As the name suggests, it consists of views(i.e. frontend part)
Data Access Layer: CRUD (create, retrieve, update, delete) operations on the database comes under this category.
Service Layer: This consist of service classes and uses services provided by data access layers.
Integration Layer: It consists of web different web services(any service available over the internet and uses XML messaging system).
Then we have utility classes, validator classes and view classes.
All the services provided by the classes are implemented in their corresponding classes and are retrieved by implementing the dependency on those interfaces.
Spring Boot flow architecture:

Spring Boot flow architecture



Since Spring boot uses all the features/modules of spring-like Spring data, Spring MVC etc. so the architecture is almost the same as spring MVC, except for the fact that there is no need of DAO and DAOImpl classes in Spring boot.

- Creating a data access layer needs just a repository class instead which is implementing CRUD operation containing class.
- A client makes the https request(PUT/GET)
- Then it goes to controller and the controller mapped with that route as that of request handles it, and calls the service logic if required.
- Business logic is performed in the service layer which might be performing the logic on the data from the database which is mapped through JPA with model/entity class
- Finally, a JSP page is returned in the response if no error occurred.

Setup Spring Boot:

1. Setup Java JDK from Oracle's official site.
2. Download and Setup STS(Spring Tools Suite).
3. Start a new spring starter project:

- Click on File -> New -> Spring starter project
- Fill the appropriate details and add dependency and finish.
- Edit the application properties.
- Run the main file as a Java application.

Q. Define Hibernate Architecture?
Ans. Hibernate: Hibernate is a framework which is used to develop persistence logic which is independent of Database software. In JDBC to develop persistence logic we deal with primitive types. Whereas Hibernate framework we use Objects to develop persistence logic which are independent of database software.
Hibernate Architecture:

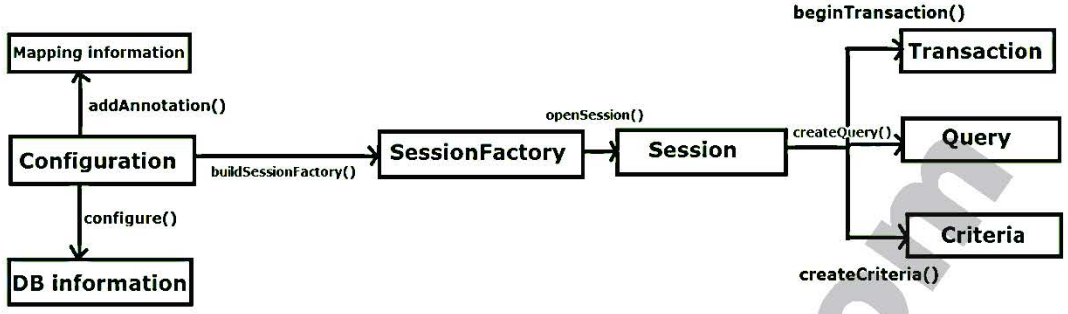


Fig: Hibernate Architecture

Configuration:

- Configuration is a class which is present in org.hibernate.cfg package. It activates Hibernate framework. It reads both configuration file and mapping files.

It activate Hibernate Framework

Configuration cfg=new Configuration();

It read both cfg file and mapping files

cfg.configure();

- It checks whether the config file is syntactically correct or not.
- If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.
- It checks whether the config file is syntactically correct or not.
- If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.

Session Factory: Session Factory is an Interface which is present in org.hibernate package and it is used to create Session Object.

It is immutable and thread-safe in nature.

Session:Session is an interface which is present in org.hibernate package. Session object is created based upon SessionFactory object i.e. factory.

- It opens the Connection/Session with Database software through Hibernate Framework.
- It is a light-weight object and it is not thread-safe.
- Session object is used to perform CRUD operations.
- Session session=factory.buildSession();

Transaction:

- Transaction object is used whenever we perform any operation and based upon that operation there is some change in database.
- Transaction object is used to give the instruction to the database to make the changes that happen because of operation as a permanent by using commit() method.
- Transaction tx=session.beginTransaction();

tx.commit();

Query:

- Query is an interface that present inside org.hibernate package.
- A Query instance is obtained by calling Session.createQuery().
- This interface exposes some extra functionality beyond that provided by Session.iterate() and Session.find():
- A particular page of the result set may be selected by calling setMaxResults(), setFirstResult().
- Named query parameters may be used.
- Query query=session.createQuery();

Criteria:

- Criteria is a simplified API for retrieving entities by composing Criterion objects.
- The Session is a factory for Criteria. Criterion instances are usually obtained via the factory methods on Restrictions.

- Criteria criteria=session.createCriteria();

Flow of working during operation in Hibernate Framework:

Suppose We want to insert an Object to the database. Here Object is nothing but persistence logic which we write on java program and create an object of that program. If we want to insert that object in the database or we want to retrieve the object from the database. Now the question is that how hibernates save the Object to the database or retrieve the object from the database. There are several layers through which Hibernate framework go to achieve the above task. Let us understated the layers/flow of Hibernate framework during performing operations:

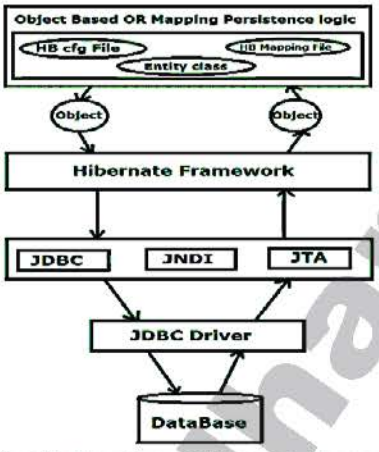


Fig: Working Flow of Hibernate framework to save/retrieve the data from the database in form of Object

Stage I: In first stage, we will write the persistence logic to perform some specific operations to the database with the help of Hibernate Configuration file and Hibernate mapping file. And after that we create an object of the particular class on which we wrote the persistence logic.

Stage II: In second stage, our class which contains the persistence logic will interact with the hibernate framework where hibernate framework gives some abstraction do perform some task. Now here the picture of java class is over. Now Hibernate is responsible to perform the persistence logic with the help of layers which is below of Hibernate framework or we can say that the layers which are the internal implementation of Hibernate.

Stage III: In third stage, our hibernate framework interact which JDBC, JNDI, JTA etc to go to the database to perform that persistence logic.

Stage IV & V: In fourth & fifth stage, hibernate is interact with Database with the help of JDBC driver. Now here hibernate perform that persistence logic which is nothing but CRUD operation. If our persistence logic is to retrieve a record then in the reverse order it will display on the console of our java program in terms of Object.

Q. What is spring boot?

Ans. Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.

Advantages: Spring Boot offers the following advantages to its developers:

- Easy to understand and develop spring applications
- Increases productivity
- Reduces the development time

Goals: Spring Boot is designed with the following goals:

- To avoid complex XML configuration in Spring
- To develop a production ready Spring applications in an easier way
- To reduce the development time and run the application independently
- Offer an easier way of getting started with the application.

Q. What is Object Relational Mapping (ORM)?

Ans. When we work with an object-oriented system, there is a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.

Consider the following Java Class with proper constructors and associated public function –

```
publicclassEmployee{
privateint id;
privateString first_name;
privateString last_name;
privateint salary;
publicEmployee(){}
publicEmployee(String fname,String lname,int salary){
this.first_name = fname;
this.last_name = lname;
this.salary = salary;
}
publicint getId(){
return id;
}
publicString getFirstName(){
return first_name;
}
publicString getLastName(){
return last_name;
}
publicint getSalary(){
return salary;
}
}
```

Consider the above objects are to be stored and retrieved into the following RDBMS table –

```
create table EMPLOYEE (
id INT NOT NULL auto_increment,
first_name VARCHAR(20)default NULL,
  last_name  VARCHAR(20)default NULL,
salary     INT  default NULL,
  PRIMARY KEY (id)
);
```

First problem, what if we need to modify the design of our database after having developed a few pages or our application? Second, loading and storing objects in a relational database exposes us to the following five mismatch problems –

Sr.No.	Mismatch & Description
1	**Granularity** Sometimes you will have an object model, which has more classes than the number of corresponding tables in the database.
2	**Inheritance** RDBMSs do not define anything similar to Inheritance, which is a natural paradigm in object-oriented programming languages.
3	**Identity** An RDBMS defines exactly one notion of 'sameness': the primary key. Java,

	however, defines both object identity (a==b) and object equality (a.equals(b)).
4	**Associations** Object-oriented languages represent associations using object references whereas an RDBMS represents an association as a foreign key column.
5	**Navigation** The ways you access objects in Java and in RDBMS are fundamentally different.

The Object-Relational Mapping (ORM) is the solution to handle all the above impedance mismatches.

MCS-220: Web Technologies
Guess Paper-3

Q. What are Hibernate mapping types?

Ans. The types declared and used in the mapping files are not Java data types; they are not SQL database types either. These types are called Hibernate mapping types, which can translate from Java to SQL data types and vice versa.

Primitive Types

Mapping type	Java type	ANSI SQL Type
Integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

Date and Time Types:

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Binary and Large Object Types

Mapping type	Java type	ANSI SQL Type
binary	byte[]	VARBINARY (or BLOB)
text	java.lang.String	CLOB
serializable	any Java class that implements java.io.Serializable	VARBINARY (or BLOB)
clob	java.sql.Clob	CLOB
blob	java.sql.Blob	BLOB

JDK-related Types

Mapping type	Java type	ANSI SQL Type
class	java.lang.Class	VARCHAR
locale	java.util.Locale	VARCHAR
timezone	java.util.TimeZone	VARCHAR
currency	java.util.Currency	VARCHAR

Q.What is web Security?

Ans. Security is critical to web services. However, neither XML-RPC nor SOAP specifications make any explicit security or authentication requirements.

There are three specific security issues with web services –

- Confidentiality
- Authentication
- Network Security

Confidentiality: If a client sends an XML request to a server, can we ensure that the communication remains confidential?

- Answer lies here –
- XML-RPC and SOAP run primarily on top of HTTP.
- HTTP has support for Secure Sockets Layer (SSL).
- Communication can be encrypted via SSL.
- SSL is a proven technology and widely deployed.

A single web service may consist of a chain of applications. For example, one large service might tie together the services of three other applications. In this case, SSL is not adequate; the messages need to be encrypted at each node along the service path, and each node represents a potential weak link in the chain. Currently, there is no agreed-upon solution to this issue, but one promising solution is the W3C XML Encryption Standard. This standard provides a framework for encrypting and decrypting entire XML documents or just portions of an XML document. You can check it at www.w3.org/Encryption

Authentication

If a client connects to a web service, how do we identify the user? Is the user authorized to use the service?

The following options can be considered but there is no clear consensus on a strong authentication scheme.

HTTP includes built-in support for Basic and Digest authentication, and services can therefore be protected in much the same manner as HTML documents are currently protected.

SOAP Digital Signature (SOAP-DSIG) leverages public key cryptography to digitally sign SOAP messages. It enables the client or server to validate the identity of the other party.

The Organization for the Advancement of Structured Information Standards (OASIS) is working on the Security Assertion Markup Language (SAML).

Network Security

There is currently no easy answer to this problem, and it has been the subject of much debate. For now, if you are truly intent on filtering out SOAP or XML-RPC messages, one possibility is to filter out all HTTP POST requests that set their content type to text/xml.

Another alternative is to filter the SOAP Action HTTP header attribute. Firewall vendors are also currently developing tools explicitly designed to filter web service traffic.

Q. Mention the features and Advantages of JSSE?

Ans. Data that travels across a network can easily be accessed by someone who is not the intended recipient. When the data includes private information, such as passwords and credit card numbers, steps must be taken to make the data unintelligible to unauthorized parties. It is also important to ensure that the data has not been modified, either intentionally or unintentionally, during transport. The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols were designed to help protect the privacy and integrity of data while it is being transferred across a network.

The Java Secure Socket Extension (JSSE) enables secure Internet communications. It provides a framework and an implementation for a Java version of the SSL and TLS protocols and includes functionality for data encryption, server authentication, message integrity, and optional client

authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol (such as HTTP, Telnet, or FTP) over TCP/IP. For an introduction to SSL, see Secure Sockets Layer (SSL) Protocol Overview.

By abstracting the complex underlying security algorithms and handshaking mechanisms, JSSE minimizes the risk of creating subtle but dangerous security vulnerabilities. Furthermore, it simplifies application development by serving as a building block that developers can integrate directly into their applications.

JSSE provides both an application programming interface (API) framework and an implementation of that API. The JSSE API supplements the core network and cryptographic services defined by the java.security and java.net packages by providing extended networking socket classes, trust managers, key managers, SSL contexts, and a socket factory framework for encapsulating socket creation behavior. Because the SSLSocket class is based on a blocking I/O model, the Java Development Kit (JDK) includes a nonblocking SSLEngine class to enable implementations to choose their own I/O methods.

The JSSE API supports the following security protocols:

TLS: version 1.0, 1.1, 1.2, and 1.3 (since JDK 8u261)

SSL (Secure Socket Layer): version 3.0

These security protocols encapsulate a normal bidirectional stream socket, and the JSSE API adds transparent support for authentication, encryption, and integrity protection.

JSSE is a security component of the Java SE platform, and is based on the same design principles found elsewhere in the Java Cryptography Architecture (JCA) framework. This framework for cryptography-related security components allows them to have implementation independence and, whenever possible, algorithm independence. JSSE uses the cryptographic service providers defined by the JCA framework.

Other security components in the Java SE platform include the Java Authentication and Authorization Service (JAAS) and the Java Security Tools. JSSE encompasses many of the same concepts and algorithms as those in JCA but automatically applies them underneath a simple stream socket API.

The JSSE API was designed to allow other SSL/TLS protocol and Public Key Infrastructure (PKI) implementations to be plugged in seamlessly. Developers can also provide alternative logic to determine if remote hosts should be trusted or what authentication key material should be sent to a remote host.

Features and Benefits: JSSE includes the following important features:

- Included as a standard component of the JDK
- Extensible, provider-based architecture
- Implemented in 100% pure Java
- Provides API support for TLS
- Provides implementations of SSL 3.0 and TLS versions 1.0, 1.1, 1.2, and 1.3 (since JDK 8u261)
- Includes classes that can be instantiated to create secure channels (SSLSocket, SSLServerSocket, and SSLEngine)
- Provides support for cipher suite negotiation, which is part of the SSL handshaking used to initiate or verify secure communications
- Provides support for client and server authentication, which is part of the normal SSL handshaking
- Provides support for HTTP encapsulated in the SSL protocol, which allows access to data such as web pages using HTTPS
- Provides server session management APIs to manage memory-resident SSL sessions
- Provides support for the certificate status request extension (OCSP stapling), which saves client certificate validation round-trips and resources
- Provides support for the Server Name Indication (SNI) extension, which extends the TLS protocols to indicate what server name the client is attempting to connect to during handshaking

- Provides support for endpoint identification during handshaking, which prevents man-in-the-middle attacks
- Provides support for cryptographic algorithm constraints, which provides fine-grained control over algorithms negotiated by JSSE

JSSE Standard API: The JSSE standard API, available in the javax.net and javax.net.ssl packages, provides:

- Secure sockets and server sockets.
- A non blocking engine for producing and consuming streams of TLS data (SSLEngine).
- Factories for creating sockets, server sockets, SSL sockets, and SSL server sockets. By using socket factories, you can encapsulate socket creation and configuration behavior.
- A class representing a secure socket context that acts as a factory for secure socket factories and engines.
- Key and trust manager interfaces (including X.509-specific key and trust managers), and factories that can be used for creating them.
- A class for secure HTTP URL connections (HTTPS).

SunJSSE Provider: Oracle's implementation of Java SE includes a JSSE provider named SunJSSE, which comes preinstalled and preregistered with the JCA. This provider supplies the following cryptographic services:

- An implementation of the security protocols SSL 3.0 and TLS 1.0, 1.1, 1.2, and 1.3 (since JDK 8u261).
- An implementation of the most common TLS cipher suites, which encompass a combination of authentication, key agreement, encryption, and integrity protection.
- An implementation of an X.509-based key manager that chooses appropriate authentication keys from a standard JCA keystore.
- An implementation of an X.509-based trust manager that implements rules for certificate chain path validation.
- An implementation of PKCS12 as JCA keystore type "pkcs12". Storing trust anchors in PKCS12 is not supported. Users should store trust anchors in the Java keystore (JKS) format and save private keys in PKCS12 format.

Q. What is spring Security?

Ans. Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

It is a sub-project of Spring framework which was started in 2003 by Ben Alex. Later on, in 2004, It was released under the Apache License as Spring Security 2.0.0.

It overcomes all the problems that come during creating non spring security applications and manage new server environment for the application.

This framework targets two major areas of application are authentication and authorization.Authenticationis the process of knowing and identifying the user that wants to access.

Authorization is the process to allow authority to perform actions in the application.

We can apply authorization to authorize web request, methods and access to individual domain.

Technologies that support Spring Security Integration

Spring Security framework supports wide range of authentication models. These models either provided by third parties or framework itself. Spring Security supports integration with all of these technologies.

- HTTP BASIC authentication headers
- HTTP Digest authentication headers
- HTTP X.509 client certificate exchange
- LDAP (Lighweight Directory Access Protocol)
- Form-based authentication
- OpenID authentication
- Automatic remember-me authentication

- Kerberos
- JOSSO (Java Open Source Single Sign-On)
- AppFuse
- AndroMDA
- Mule ESB
- DWR(Direct Web Request)

The beauty of this framework is its flexible authentication nature to integrate with any software solution. Sometimes, developers want to integrate it with a legacy system that does not follow any security standard, there Spring Security works nicely.

Advantages: Spring Security has numerous advantages. Some of that are given below.

- Comprehensive support for authentication and authorization.
- Protection against common tasks
- Servlet API integration
- Integration with Spring MVC
- Portability
- CSRF protection
- Java Configuration support

Spring Security History: In late 2003, a project **Acegi Security System for Spring** started with the intention to develop a Spring-based security system. So, a simple security system was implemented but not released officially. Developers used that code internally for their solutions and by 2004 about 20 developers were using that.

Initially, authentication module was not part of the project, around a year after, module was added and complete project was reconfiguring to support more technologies.

Q. Write steps to create custom login form?

Ans. Creating a login page: Within Spring Web MVC there are two steps to creating our login page:

Creating a controller

Creating a view

Configuring a login view controller

Within Spring Web MVC, the first step is to ensure that we have a controller that can point to our view. Since our project adds the **javaconfig/messages** project as a dependency and it contains a view controller for **/login** we do not need to create a controller within our application. For reference, you can see the configuration below:

```
// ...
@EnableWebMvc
@ComponentScan("org.springframework.security.samples.mvc")
publicclassWebMvcConfigurationextendsWebMvcConfigurerAdapter{
// ...
@Override
publicvoid addViewControllers(ViewControllerRegistry registry){
registry.addViewController("/login").setViewName("login");
registry.setOrder(Ordered.HIGHEST_PRECEDENCE);
}
}
```

Creating a login view

Our existing configuration means that all we need to do is create a **login.html** file with the following contents:

src/main/resources/views/login.html

```
<htmlxmlns:th="http://www.thymeleaf.org"xmlns:tiles="http://www.thymeleaf.org">
<head>
<titletiles:fragment="title">Messages : Create</title>
</head>
```

```
<body>
<divtiles:fragment="content">
<formname="f"th:action="@{/login}"method="post">
<fieldset>
<legend>Please Login</legend>
<divth:if="${param.error}"class="alert alert-error">
Invalid username and password.
</div>
<divth:if="${param.logout}"class="alert alert-success">
            You have been logged out.
</div>
<labelfor="username">Username</label>
<inputtype="text"id="username"name="username"/>
<labelfor="password">Password</label>
<inputtype="password"id="password"name="password"/>
<divclass="form-actions">
<buttontype="submit"class="btn">Log in</button>
</div>
</fieldset>
</form>
</div>
</body>
</html>
```

The URL we submit our username and password to is the same URL as our login form (i.e. **/login**), but a **POST** instead of a **GET**.
When authentication fails, the browser is redirected to **/login?error** so we can display an error message by detecting if the parameter **error** is non-null.
When we are successfully logged out, the browser is redirected to **/login?logout** so we can display an logout success message by detecting if the parameter **logout** is non-null.
The username should be present on the HTTP parameter username
The password should be present on the HTTP parameter password

Q. What is role based access Control?

Ans. Role-based access control (RBAC) is a method of restricting network access based on the roles of individual users within an enterprise.

RBAC ensures employees access only information they need to do their jobs and prevents them from accessing information that doesn't pertain to them.

An employee's role in an organization determines the permissions that individual is granted and ensures lower-level employees can't access sensitive information or perform high-level tasks.

In the role-based access control data model, roles are based on several factors, includingauthorization,responsibility and job competency. As such, companies can designate whether a user is an end user, an administrator or a specialist user. In addition, access to computer resources can be limited to specific tasks, such as the ability to view, create or modify files.

Limiting network access is important for organizations that have many workers, employ contractors or permit access to third parties, like customers and vendors, which makes it difficult to monitor network access effectively. Companies that depend on RBAC are better able to secure their sensitive data and critical applications.

Benefits of RBAC: There are multiple benefits to using RBAC, including:

- **Improving operational efficiency:** With RBAC, companies can decrease the need for paperwork and password changes when they hire new employees or switch the roles of existing employees. RBAC lets organizations quickly add and change roles, as well as implement them across platforms, operating systems (OSes) and applications. It also cuts

down on the potential for error when assigning user permissions. Additionally, with RBAC, companies can more easily integrate third-party users into their networks by giving them predefined roles.

- **Enhancing compliance:** Every organization must comply with local, state and federal regulations. Companies generally prefer to implement RBAC systems to meet the regulatory and statutory requirements for confidentiality and privacy because executives and IT departments can more effectively manage how the data is accessed and used. This is particularly important for financial institutions and healthcare companies that manage sensitive data.
- **Giving administrators increased visibility:** RBAC gives network administrators and managers more visibility and oversight into the business, while also guaranteeing authorized users and guests on the system are only given access to what they need to do their jobs.
- **Reducing costs.** By not allowing user access to certain processes and applications, companies may conserve or more cost-effectively use resources, such as network bandwidth, memory and storage.
- **Decreasing risk of breaches and data leakage:** Implementing RBAC means restricting access to sensitive information, thus reducing the potential for data breaches or data leakage.

Best practices for role-based access control implementations: There are a number of best practices organizations should follow for implementing RBAC, including:

- Determine the resources for which companies need to control access, if they're not already listed -- for instance, customer databases, email systems and contact management systems.
- Analyze the workforce and establish roles that have the same access needs. However, don't create too many roles because that would defeat the purpose of role-based access control and create user-based access control instead. Some RBAC examples include a basic role that includes the access every employee needs, such as to email and the corporate intranet. Another role could be that of a customer service representative who would have read/write access to the customer database. Another role could be that of a customer database admin with full control of the customer database.
- After creating a list of roles and their access rights, align the employees to those roles, and set their access.
- Evaluate how roles can be changed, as well as how to close accounts for those leaving the company and how to register new employees.
- Ensure RBAC is integrated across all systems throughout the company.
- Conduct training so employees understand the principles of RBAC.
- Periodicallyconduct audits of the roles, so the employees who are assigned to them and the access that's permitted for each role. If a role is found to have unnecessary access to a certain system, change the role and modify the access level.