**Spring Framework**

The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications. One of the most popular Java Enterprise Edition (Java EE) frameworks, Spring helps developers create high performing applications using plain old Java objects (POJOs).

A framework is a large body of predefined code to which developers can add code to solve a problem in a specific domain. There are many popular Java frameworks including Java Server Faces (JSF), Maven, Hibernate, Struts, and Spring.

Diagram

Description automatically generated

Refer <https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/overview.html> for more details

**Why Spring?**

Java programs are complex and feature many heavyweight components. Heavyweight means the components are dependent on the underlying operating system (OS) for their appearance and properties.

Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight -- efficient at utilizing system resources -- and has a lot of support.

Spring removes tedious configuration work so that developers can focus on writing business logic. Spring handles the infrastructure so developers can focus on the application.

**How Spring works**

A web application (layered architecture) commonly includes three layers:

**Presentation/view layer (UI)** - This is the outermost layer which handles the presentation of content and interaction with the user.

**Business logic layer -** The central layer that deals with the logic of a program.

**Data access layer** - The deep layer that deals with data retrieval from sources.

Each layer is dependent on the other for an application to work. In other words, the presentation layer talks to the business logic layer, which talks to the data access layer. Dependency is what each layer needs to perform its function. A typical application has thousands of classes and many dependencies.

Without a Spring Framework, application code tends to be tightly coupled (interdependent), which is not considered good coding practice. Loose coupling is ideal because loosely coupled components are independent, meaning changes in one will not affect the operation of others.

Spring’s core logic is **dependency injection**. Dependency injection is a programming pattern that allows developers to build more decoupled architectures. Dependency injection means that Spring understands the different Java annotations that a developer puts on top of classes. Spring knows that the developer wants to create an instance of a class and that Spring should manage it. Spring also understands the dependency and makes sure that all instances created have properly populated dependencies.

For the Spring Framework to instantiate objects and populate the dependencies, a programmer simply tells Spring which objects to manage and what the dependencies are for each class. A developer does so by using annotations like:

@**component** - Lets Spring know which classes to manage (create). Marks the beans (objects) as managed components, which means that Spring will autodetect these classes for dependency injection.

@**autowired** - Tells Spring how to handle the instantiation of the class (so it starts looking for that dependency among components/classes to find a match). This spares developers from wiring with code and allows Spring to find what needs to be injected where.

**Important terms**

**Autowiring** - The process by which Spring identifies dependencies and matches and populates them.

**Bean** - A Spring bean is an object that is instantiated, created, and managed by the IoC container. Beans are the backbone of an application.

**Dependency injection** - A programming design pattern that makes code loosely coupled, meaning that any change in the application of one, will not affect the other.

**Inversion of control (IoC)** - Taking control away from the class and giving it to the Spring Framework.

**Inversion of control container** - This is the core of the Spring Framework where objects are created, wired together, configured, and managed throughout their life cycle.

**What is Maven?**

Maven is a popular open-source build tool developed by the Apache Group to build, publish, and deploy several projects at once for better project management. The tool provides allows developers to build and document the lifecycle framework.

Maven is written in Java and is used to build projects written in C#, Scala, Ruby, etc. Based on the Project Object Model (POM), this tool has made the lives of Java developers easier while developing reports, checks build and testing automation setups.

Maven focuses on the simplification and standardization of the building process, taking care of the following:

Builds

Documentation

Dependencies

Reports

SCMs

Distribution

Releases

Mailing list

**What is Maven: Objective**

Maven’s purpose is to provide developers with:

* A comprehensive, maintainable, reusable, and simple model for projects.
* A set of tools and plug-ins that can interact with the declarative model.

**What is Maven: Features**

Maven is loaded with many valuable and useful features, which goes a long way towards explaining its popularity. Here are some of Maven’s more noteworthy features:

* A huge, continuously growing repository of user libraries
* The ability to set up projects easily, using best practices
* Dependency management, featuring automatic updating
* Backwards compatible with previous versions
* Strong error and integrity reporting
* Automatic parent versioning
* Ensures consistent usage across all projects
* It’s extensible, and you can easily write plug-ins using scripting languages or Java

**Project Object Model (POM)**

Maven is so useful thanks to the Project Object Model (POM), which is an [XML file](https://www.simplilearn.com/tutorials/programming-tutorial/what-is-xml) that has all the information regarding project and configuration details. The POM has the description of the project, details regarding the versioning, and configuration management of the project.

The XML file is located in the project home directory. When you execute a task, Maven searches for the POM in the current directory.

**The Need for Maven**

Maven is chiefly used for Java-based projects, helping to download dependencies, which refers to the libraries or JAR files. The tool helps get the right JAR files for each project as there may be different versions of separate packages.

After Maven, downloading dependencies doesn’t require visiting the official websites of different software. You can visit [mvnrepository](https://mvnrepository.com/" \o "mvnrepository" \t "_blank) to find libraries in different languages. The tool also helps to create the right project structure in struts, servlets, etc., which is essential for execution.

**How to Use Maven**

When you’re ready to start [using Maven](https://www.simplilearn.com/tutorials/maven-tutorial/how-to-install-maven), keep these three things in mind:

1. Configure Maven in Java, using Project Object Model (POM) found in a pom.xml file.
2. All Maven-related configuration settings are found in the POM. You can edit and configure plug-ins in the <plugins> tag of a pom.xml file.
3. Maven provides default settings for configurations, so you don’t have to add every configuration into the pom.xml file.

**Advantages of Maven**

* Helps manage all the processes, such as building, documentation, releasing, and distribution in project management
* Simplifies the process of project building
* Increases the performance of the project and the building process
* The task of downloading Jar files and other dependencies is done automatically
* Provides easy access to all the required information
* Makes it easy for the developer to build a project in different environments without worrying about the dependencies, processes, etc.
* In Maven, it’s easy to add new dependencies by writing the dependency code in the pom file

**Conversely, Maven has a few drawbacks.**

* Maven requires installation in the working system and the Maven plug-in for the IDE
* If the Maven code for an existing dependency is unavailable, you cannot add that dependency using Maven itself
* Some[sources](https://onlineitguru.com/blog/explain-the-advantages-and-disadvantages-of-maven) claim that Maven is slow

**Spring Boot**

Spring Boot is an open-source micro framework maintained by a company called Pivotal. It provides Java developers with a platform to get started with an auto configurable production-grade Spring application. With it, developers can get started quickly without losing time on preparing and configuring their Spring application.

This post will guide you through understanding what Spring Boot is and what it’s used for.

**What’s the Difference Between Spring and Spring Boot?**

Spring Boot is built on top of the Spring framework, and it comes with many dependencies that can be plugged into the Spring application. Some examples are Spring Kafka, Spring LDAP, Spring Web Services, and Spring Security. However, developers have to configure each building brick themselves using a lot of XML configuration files or annotations.

Now, how do they differ?

Well, the Spring framework focuses on providing flexibility through its dependency injection feature. It helps to inject the required dependencies quickly but also to develop your application in a loosely coupled fashion. Some other benefits include:

* A lightweight framework.
* Helps with loose coupling dependencies and testability. The modular architecture allows you to pick the parts you need and isolate them.
* Has support for both XML and annotation configuration.
* Provides abstraction on ORM software to develop the ORM persistence logic.
* Compatible with many middleware services.
* Supports the JDBC framework, which improves productivity and reduces errors.

Spring Boot, on the other hand, is focused on shortening the code length and providing you with an easy way to run your Spring application.

**Notable Features of Spring Boot**

**Autoconfiguration**: Developers can automatically configure their Spring application. However, Spring Boot is also capable of changing the configuration based on the dependencies you list. For example, when you list “MySQL” as a dependency, it will configure your Spring application with the “MySQL connector” included. And if you want to add a custom configuration, you can create a class that overrides the default configuration for your “MySQL connector”.

**Standalone**: There’s no need to deploy your application to a web server. You simply enter the run command to start the application.

**Opinionated**: On the official page, we find that Spring Boot decides for you which defaults to use for the configuration. Also, it decides which packages to install for the dependencies you require. For example, if you include the Spring Boot starter “pom” for “JPA”, it will autoconfigure an in-memory database, a hibernate entity manager, and a simple data source. This is an example of an opinionated default configuration that you can override. While some developers might feel this is too opinionated, Spring Boot’s opinionated setup helps developers to get started quickly on their projects.

**Additional benefits include:**

* Reduces development time and increases the overall productivity of the development team.
* Helps you autoconfigure all components for a production-grade Spring application.
* Makes it easier for developers to create and test Java-based applications by providing a default setup for unit and integration tests.
* Avoids writing lots of boilerplate code, annotations, and XML configuration.
* Comes with embedded HTTP servers like [Tomcat or Jetty](https://stackify.com/tomcat-vs-jetty-vs-glassfish-vs-wildfly/) to test web applications.
* Adds many plugins that developers can use to work with embedded and in-memory databases easily. Spring allows you to easily connect with database and queue services like Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ, and [many more](https://spring.io/guides).
* Allows admin support—meaning you can manage via remote access to the application.