

1.2 Setting Up a Spring Boot Project

[Spring initializr](#) is the website from where we can download our starter application and use that to build java applications.

The screenshot shows the Spring Initializr website interface. A red arrow points from the 'spring initializr' search result on Google to the main page. Another red arrow points from the 'Build Tools' section to the Java dependency selection. A third red arrow points from the 'Project Metadata' section to the 'Name' field, which is highlighted with a purple oval. A fourth red arrow points from the 'Dependencies' section to the 'Spring Web' dependency, which is also highlighted with a purple oval. A bracket on the right side indicates that specific dependencies can be selected. A large red arrow points from the 'GENERATE' button to a callout box containing instructions for generating the project.

Google search results for "spring initializr". The first result is the official Spring Initializr website. The page shows project configuration options like Maven, Java version (3.5.3), and Spring Boot version (3.4.7). It also displays dependencies (Spring Web) and metadata fields (Group, Artifact, Name, Description, Package name, Packaging). A 'GENERATE' button is at the bottom.

Build Tools

Dependencies

Project Metadata

Dependencies

Just click on this generate button and it will generate a zip file for you. Now go ahead and uncompress this file and open it inside your intellij IDE.

Spring Web

Spring Reactive Web

Spring for GraphQL

Rest Repositories

Spring Session for Spring Data MongoDB

The screenshot shows an IDE interface with the following components:

- Project View:** Shows the directory structure of the project. The `src/main/java` folder contains a package named `com.sibasundar.week1Introduction.introductionToSpringBoot`, which contains a class named `IntroductionToSpringBootApplication`. This class is highlighted in the code editor.
- Code Editor:** Displays the `IntroductionToSpringBootApplication.java` file. The code includes the package declaration, imports, and the main method annotated with `@SpringBootApplication`.
- Run Tab:** Shows the run configuration for the application, with the name `IntroductionToSpringBootApplication`.
- Console Tab:** Displays the application logs. A log entry from Tomcat indicates that it has initialized with port `8080` (`http`).
- Status Bar:** Shows the current path as `introductionToSpringBoot > src > main > java > com > sibasundar > week1Introduction > introductionToSpringBoot > IntroductionToSpringBootApplication.java`, the time as `13:21`, and the encoding as `UTF-8`.

As you can see, **Apache Tomcat Server** has successfully started on **port 8080**. Spring Boot has automatically configured the embedded **Servlet Container** for us, eliminating the need for manual setup. Now, we can focus entirely on writing the **business logic** without worrying about infrastructure configuration.

Test folder:

Spring Initializr automatically includes the **test** folder as part of **Maven's** standard directory structure for writing **unit** and **integration tests** to validate your **Spring Boot** application's **correctness**, **reliability**, and **stability**.

mvnw, mvnw.cmd

- These are **Maven Wrapper** scripts:
 - They allow anyone to build your project **without** installing **Maven** globally.
 - `mvnw` for Linux/macOS, `mvnw.cmd` for Windows.

pom.xml

- This is the **Maven Project Object Model** file.
- It defines your project's **dependencies**, **plugins**, **Java version**, **build settings**, etc.