

Spring Boot Project Setup - MongoDB

This document describes how to create a Spring Boot project with a MongoDB connection.

Spring Boot Extension

Install (if necessary) the Spring Boot Extension for VS Code

<https://code.visualstudio.com/docs/java/java-spring-boot>

Prerequisites

To develop a Spring Boot application in Visual Studio Code, you need to install the following:

- [Java Development Kit \(JDK\)](#)
- [Extension Pack for Java](#)
- [Spring Boot Extension Pack](#)

Install the Extension Pack for Java

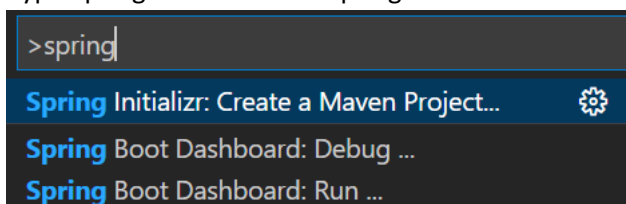
Install the Spring Boot Extension Pack

Spring Boot Project Setup

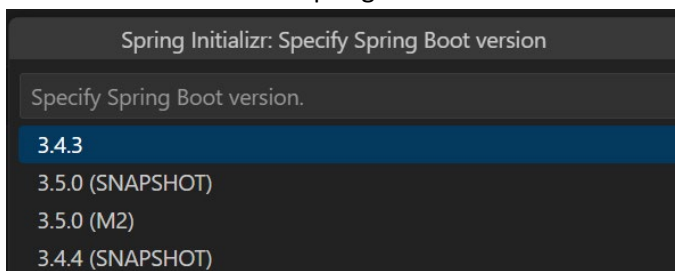
- a) Launch VS Code. Open a new window (File → New Window) if a project is already open. Open the Command Palette (Menu View, Command Palette) or with the corresponding shortcut:

- Windows: Ctrl+Shift+P,
- Mac: ⌘⇧P

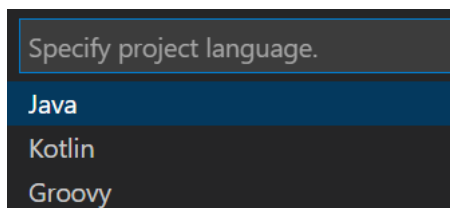
- b) Type Spring and select the Spring Initializer for Maven projects.



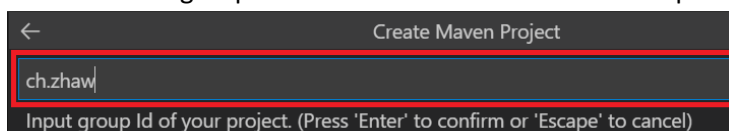
- c) Choose the latest stable Spring Boot version.



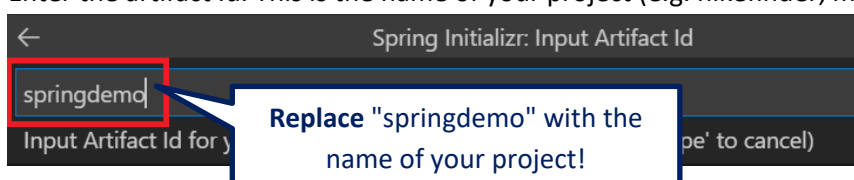
Choose Java.



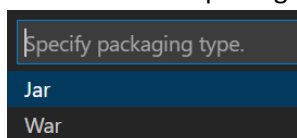
And enter the group id. This is used for the name of the package.



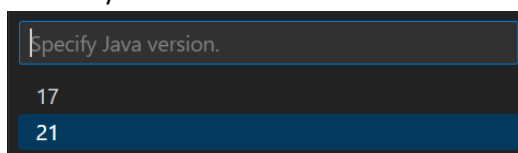
Enter the artifact Id. This is the name of your project (e.g. hikefinder, myApp).



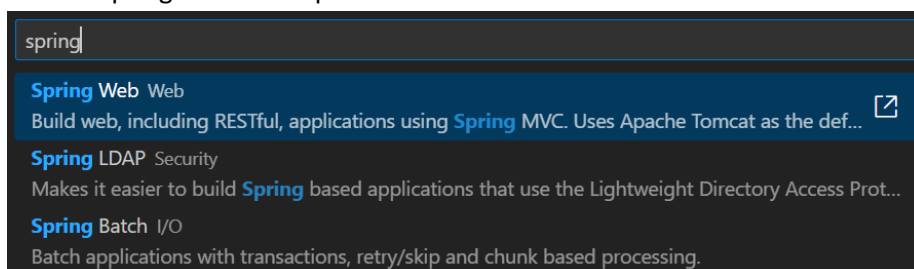
Select Jar as the package type.



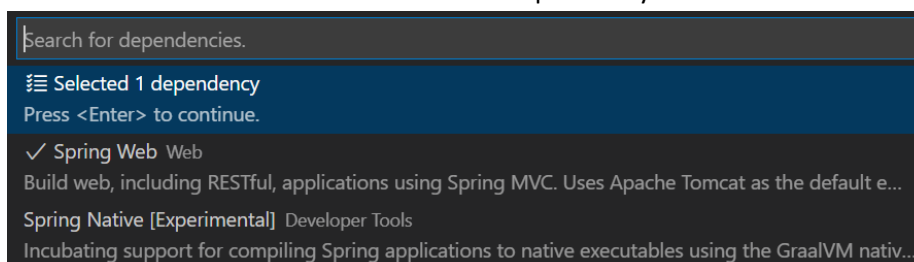
Choose your Java version. We recommend using version 21. This is the latest LTS version.



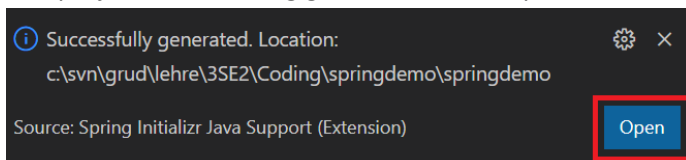
Select "Spring Web" at dependencies and click on it.



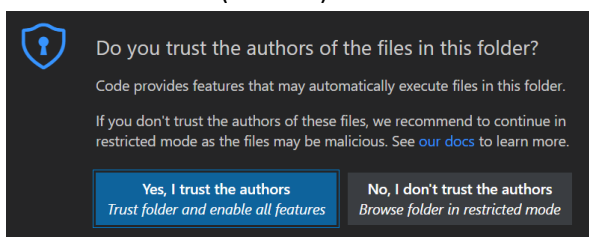
Press Enter to continue with the selected dependency.



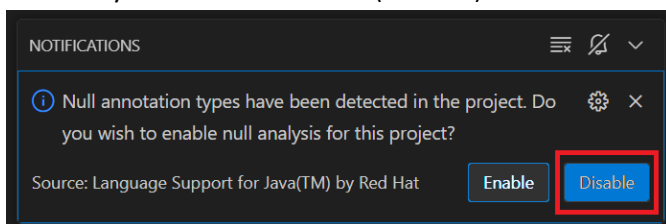
- d) Next, you need to choose the folder where you want the project to be saved. **VS Code will create a new subfolder in the selected folder with the name you entered as the artifact ID.** The selected folder must be **local**. Don't use a folder that's in the cloud (OneDrive or similar). Do not use umlauts or spaces in the name of the folder. Also parent folders should also not contain spaces and umlauts.
- e) The project is now being generated. Click Open when the corresponding dialog appears.



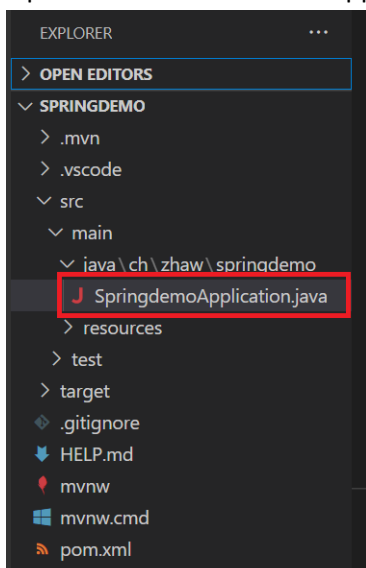
Trust the authors (if asked).



Null analysis can be turned off (if asked).



- f) Open the main class of the application.



- g) Wait for the build to complete (status bar). Start the application with the run button above the main method as soon as it appears. In the terminal, you should now see that the application has been successfully launched.

```

src > main > java > ch > zhaw > springdemo > J SpringdemoApplication.java > Language Support for Java(TM) by Red Hat > {} ch.zhaw.springdemo
1 package ch.zhaw.springdemo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SpringdemoApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(SpringdemoApplication.class, args);
11     }
12 }

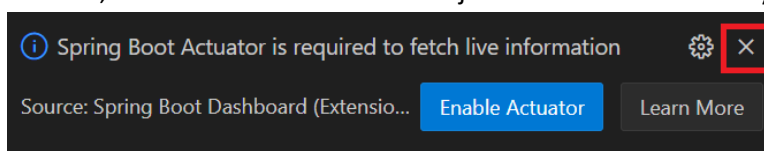
```

```

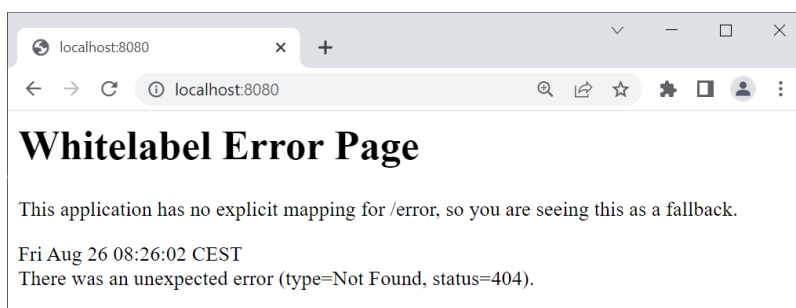
TERMINAL
2022-08-25 11:03:47.940 INFO 17720 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.65]
2022-08-25 11:03:48.121 INFO 17720 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-08-25 11:03:48.121 INFO 17720 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2253 ms
2022-08-25 11:03:49.281 INFO 17720 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-08-25 11:03:49.312 INFO 17720 --- [main] c.zhaw.springdemo.SpringdemoApplication : Started SpringdemoApplication in 4.255 seconds (JVM running for 4.996)

```

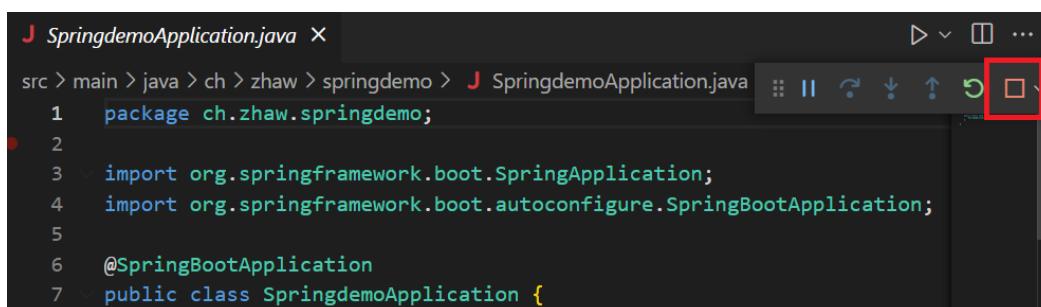
If asked, don't enable the Actuator. It just needs an unnecessary amount of resources.



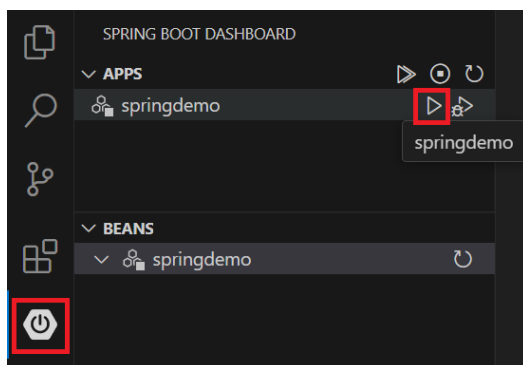
- h) The application is now running. When opening <http://localhost:8080/>, the "Whitelabel Error Page" should be displayed.



- i) The application can be stopped with the stop button.



- j) Alternatively, the application can also be started and stopped with the Spring Boot Dashboard.



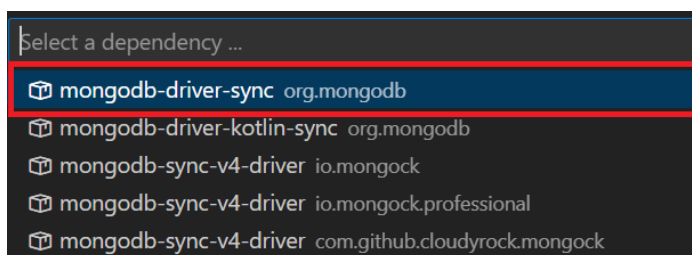
- k) Finally, add the following dependency to the pom.xml (in section dependencies). This dependency will automatically restart the application when changes are made.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
  <optional>true</optional>
</dependency>
```

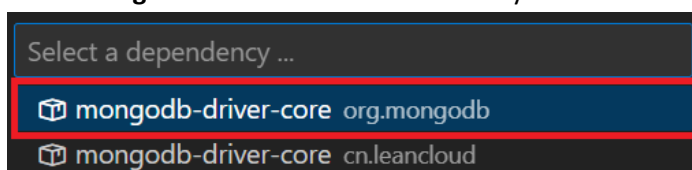
Don't forget to save the pom file, synchronize the classpath and manually stop and start the application again. From now on, the application should restart itself automatically when changes are made.

Link to MongoDB

- a) Open the Command Palette (Windows: Ctrl+Shift+P, Mac: ⌘⇧P) and run the command «Maven: Add a dependency..». Search for «**mongodb-driver-sync**». Choose the following variant:



Add «**mongodb-driver-core**» the same way.



Then save the POM file again and let the project be synchronized.

- b) Create a new folder named «controller» in src\main\java\ch\zhaw\[projectname].
- c) Create a new file within this folder named «MongoDB.java». Replace the file content with the code from below but keep the **original package** name. In addition, enter the **DB-Name** and the **connection string** (same as in previous Java projects).

```
package ch.zhaw.springdemo.controller;

import java.util.ArrayList;
import java.util.Arrays;
import org.bson.Document;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.mongodb.ConnectionString;
import com.mongodb.MongoClientSettings;
import com.mongodb.ServerApi;
import com.mongodb.ServerApiVersion;
import com.mongodb.client.AggregateIterable;
import com.mongodb.client.ListCollectionNamesIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

@RestController
public class MongoDB {

    private static MongoDatabase myDB = null;
    private static String databaseName = "...";
    private static String connectionString = "...";

    public MongoDB() {
        try {
            ServerApi serverApi = ServerApi.builder()
                .version(ServerApiVersion.V1)
                .build();
            MongoClientSettings mongodbSettings = MongoClientSettings.builder()
                .applyConnectionString(new ConnectionString(connectionString))
                .serverApi(serverApi)
                .build();
            MongoClient mongoClient = MongoClients.create(mongodbSettings);
            myDB = mongoClient.getDatabase(databaseName);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @GetMapping("/testmongodb")
    public String index() {
        ListCollectionNamesIterable col = myDB.listCollectionNames();
        ArrayList<String> collist = col.into(new ArrayList<String>());
        return collist.toString();
    }
}
```

- d) When opening <http://localhost:8080/testmongodb> now, you should get a list of all collection names in the selected DB. You may have to restart the app manually.