

# School Timetabling (Java, Spring Boot, Maven or Gradle)

Assign lessons to timeslots and rooms to produce a better schedule for teachers and students.

The screenshot shows a web application for school timetabling. At the top, there is a 'Solve' button and a 'Score: ?' indicator. Below these are three tabs: 'By room' (selected), 'By teacher', and 'By student group'. The main area is a table with 'Timeslot' on the left and 'Room A', 'Room B', and 'Room C' as columns. The timeslots are listed for Monday and Tuesday, ranging from 08:30 to 15:30. Below the table, there is a section titled 'Unassigned lessons' containing six colored boxes representing lessons: Math (green, by A. Turing, 9th grade, 0), Math (green, by A. Turing, 9th grade, 1), Physics (yellow, by M. Curie, 9th grade, 2), Chemistry (blue, by M. Curie, 9th grade, 3), Biology (orange, by C. Darwin, 9th grade, 4), and History (purple, by I. Jones, 9th grade, 5).

- [Run the application](#)

## Run the application

1. Git clone the timefold-quickstarts repo and navigate to this directory:

```
$ git clone https://github.com/TimefoldAI/timefold-quickstarts.git
...
$ cd timefold-quickstarts/technology/java-spring-boot
```

2. Start the application with Maven:

```
$ mvn spring-boot:run
```

or with Gradle:

```
$ gradle bootRun
```

3. Visit <http://localhost:8080> in your browser.
4. Click on the **Solve** button.

## Run the packaged application

When you're ready to deploy the application, package the project to run as a conventional jar file.

1. Build it with Maven:

```
$ mvn package
```

or with Gradle:

```
$ gradle clean build
```

2. Run the Maven output:

```
$ java -jar target/spring-boot-school-timetabling-1.0-SNAPSHOT.jar
```

or the Gradle output:

```
$ java -jar build/libs/spring-boot-school-timetabling-1.0-SNAPSHOT.jar
```

**NOTE** To run it on port 8081 instead, add `-Dserver.port=8081`.

3. Visit <http://localhost:8080> in your browser.
4. Click on the **Solve** button.

## Create a native image

**IMPORTANT** The solver runs considerably slower in a native image.

If you want faster startup times or need to deploy to an environment without a JVM, you can build a native image.

## Build using Docker

1. Build a Docker image with Maven:

```
$ mvn -Pnative spring-boot:build-image
```

or with Gradle:

```
$ gradle bootBuildImage
```

2. Start the built Docker image using **docker run**:

```
$ docker run --rm -p 8080:8080 docker.io/library/spring-boot-school-timetabling:1.0-SNAPSHOT
```

3. Visit <http://localhost:8080> in your browser.
4. Click on the **Solve** button.

## Build using locally installed GraalVM

1. Build it with Maven:

```
$ mvn -Pnative native:compile
```

or with Gradle:

```
$ gradle nativeCompile
```

2. Run the Maven output:

```
$ ./target/spring-boot-school-timetabling
```

or the Gradle output:

```
$ ./build/native/nativeCompile/java-spring-boot
```

**NOTE** To run it on port 8081 instead, add **-Dserver.port=8081**.

3. Visit <http://localhost:8080> in your browser.
4. Click on the **Solve** button.

## More information

Visit [timefold.ai](https://timefold.ai).