Hello Scout JS Full Stack

Version 11.0

Table of Contents

Looking for something else? Visit https://eclipsescout.github.io for all Scout related documentation.

Introduction

In this tutorial we will create a Scout JS application consisting of a JavaScript frontend connected using REST to a Java backend that stores the data in a database.

If you don't know what Scout JS is yet, please read the Get Started Guide first.

We will create the application using IntelliJ. The generated application can display persons loaded from a database. These persons can be edited or deleted and new persons can be added.

Prerequisites

This section guides you through the installation of the tools required to start the `Hello Scout JS Full Stack` application.

Node.js

First, Node.js needs to be installed as Scout uses it to build web assets. So if you don't have it yet, visit the Node.js download site, choose the package for your platform and install it on your local machine.

Make sure the Node.js installation is on the PATH. You can verify it by using your command line:

```
c:\> node --version
v12.21.0
```

Add pnpm

Scout uses pnpm 5 as package manager. Therefore, install it into your Node installation by using your command line:

```
npm install -g pnpm
```

and verify that it was installed successfully with:

```
c:\> pnpm -v
5.18.2
```

IntelliJ

If you have no IntelliJ yet, you can download it from the JetBrains download site. We recommend selecting the Ultimate edition to have the JavaScript support included in the IDE. There is a 30-day trial if you have no licence. For this tutorial you can also use the free Community Edition, but it requires some extra steps indicated.

Install or extract the package and run it using `bin/idea64.exe` in the installation folder. Follow the instructions until the `Welcome to IntelliJ IDEA` screen is shown.

On the left side switch to `Plugins`, search for `Eclipse Scout` and press the green `Install` button. In case a `Third-Party Plugins Privacy Notice` is shown, press `Accept`. The Scout plugin does not collect or process any personal data. Afterwards, the plugin is being downloaded from the JetBrains Marketplace and installed locally. As soon as this is completed, press the `Restart IDE` button.

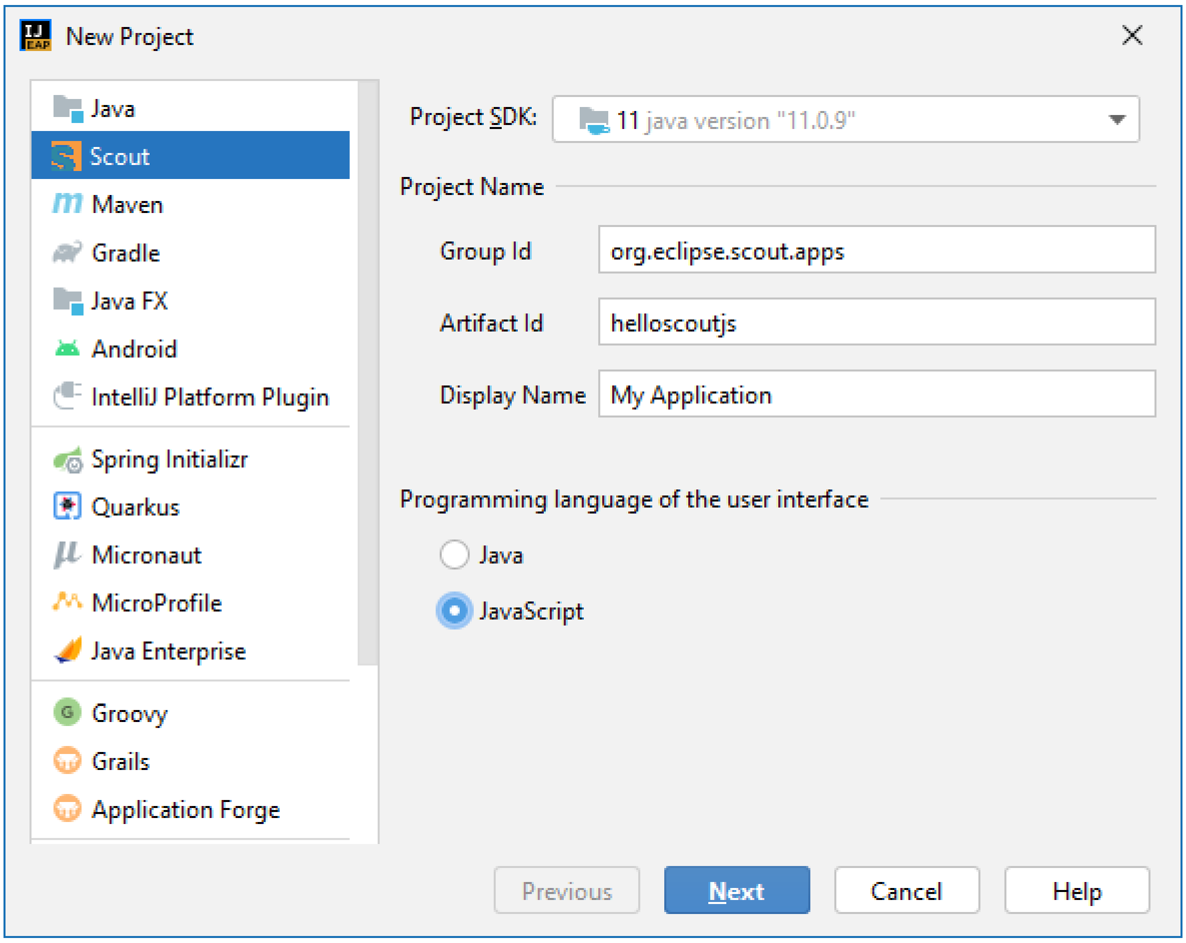
The same can also be achieved by navigating to `File | Settings | Plugins` in case you already have an existing IntelliJ project running.

Congratulations! You have successfully set up IntelliJ IDEA for Scout development.

Create the Project

Start your IntelliJ (if not already running) and in the `Welcome to IntelliJ IDEA` screen click on `New Project`. The `New Project` wizard starts. The same can also be achieved from the menu `File | New | Project…` if an existing project is open already.

On the left side select the `Scout` type. You have to enter a `Group Id`, `Artifact Id` and a `Display Name` for your Scout project as shown in Figure 1. As the created project will make use of Apache Maven, please refer to the Maven naming conventions to choose `Group Id` and `Artifact Id` for your project. The `Display Name` is used as the application name presented to the user (e.g. in the browser title bar).



Project SDK: 11 java version "11.0.9"

Project Name

Group Id: org.eclipse.scout.apps

Artifact Id: helloscoutjs

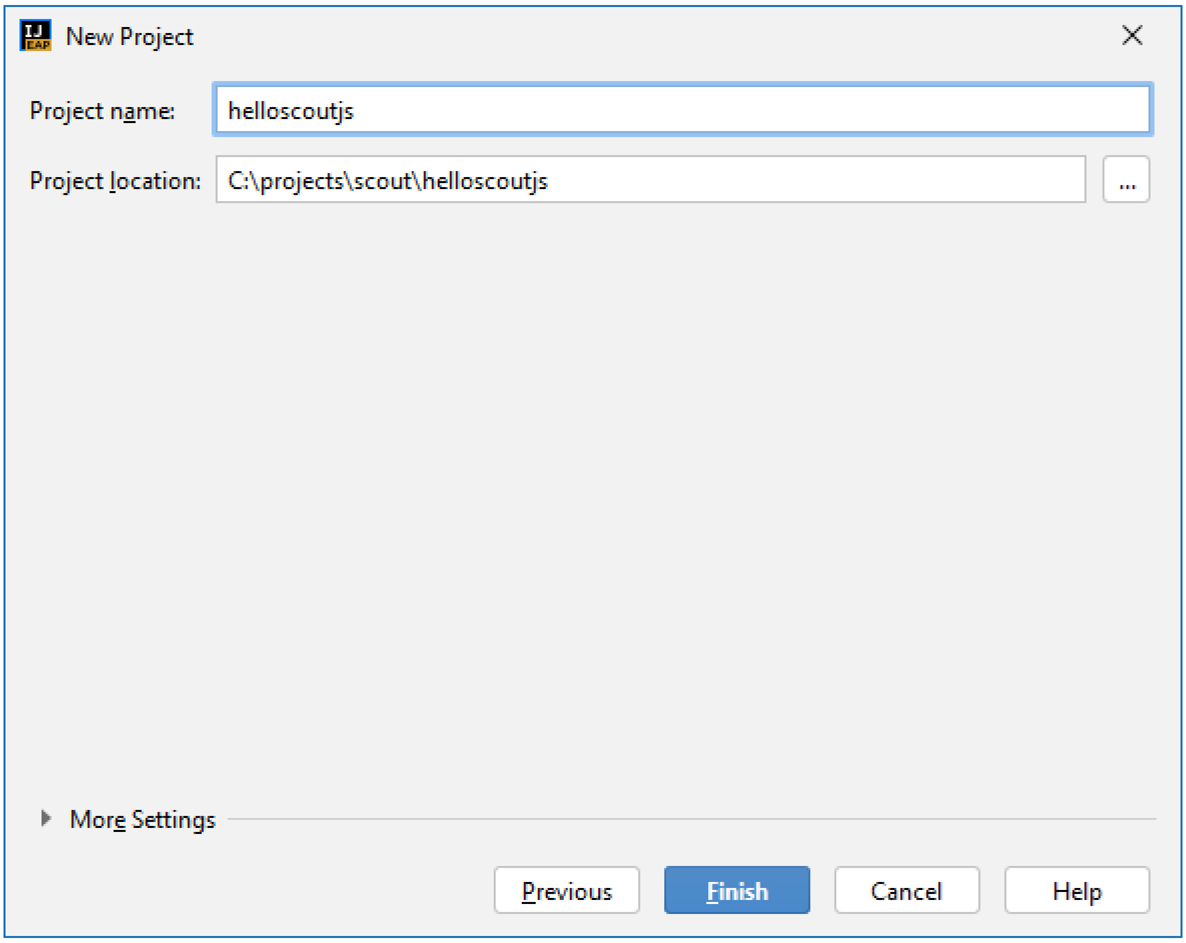
Display Name: My Application

Programming language of the user interface

Java [✓ JavaScript]

Figure 1. The new Scout project wizard.

For the `Hello Scout JS Full Stack` application use `helloscoutjs` as the artifact id and ensure the user interface programming language is set to `JavaScript` as sown in Figure 1. You can keep the other default values. Then click the **[Next]** button.


New Project

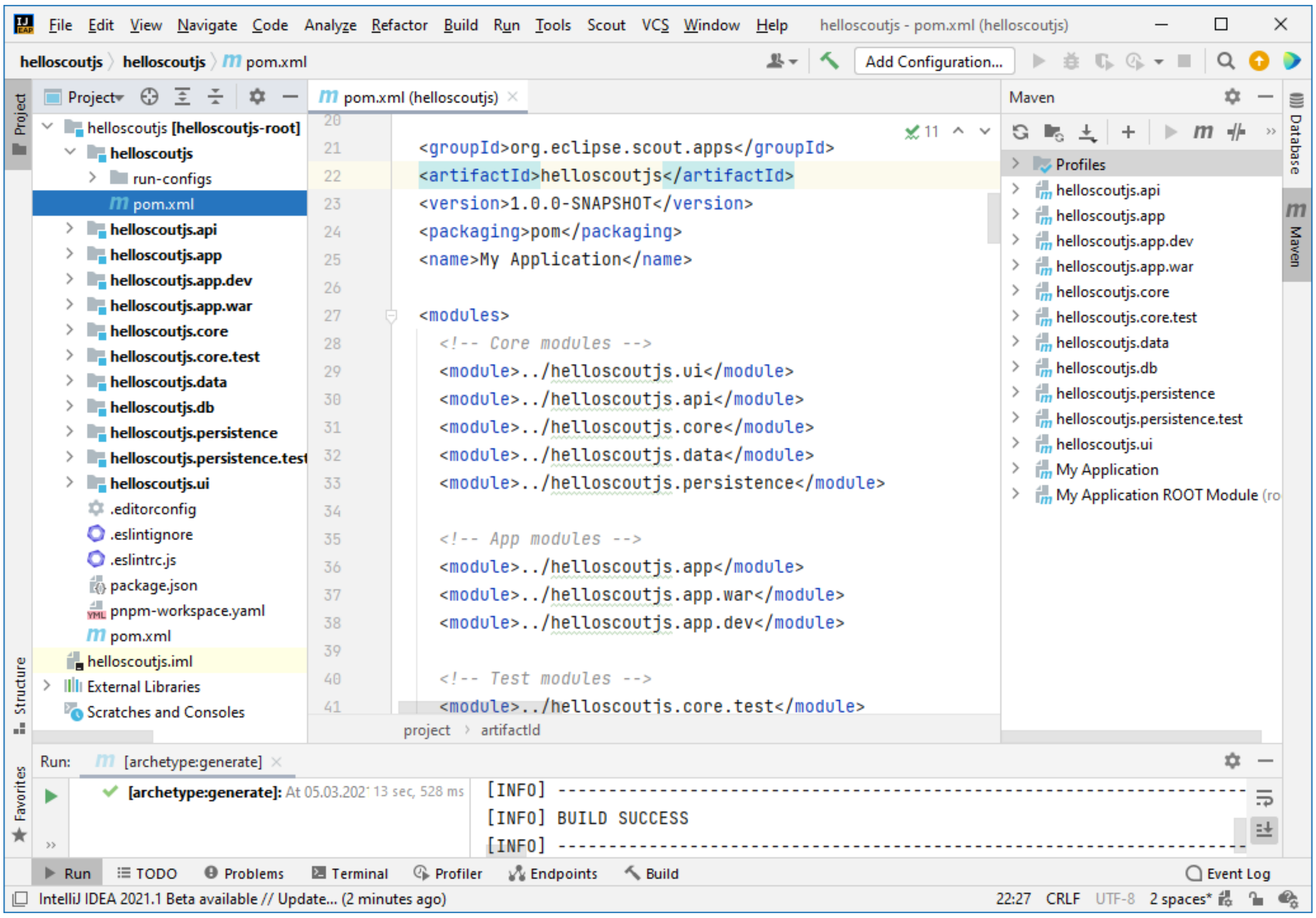
Project name: helloscoutjs

Project location: C:\projects\scout\helloscoutjs

Figure 2. Specify name and location of the new project.

On the second page please specify project name and location and press **[Finish]**. The Scout plugin then creates the initial project content for you (you will see some Maven build output). Wait until all tasks have completed.

Afterwards, you will find the created Scout modules in the Project view as shown in Figure 3.



```
  <groupId>org.eclipse.scout.apps</groupId>
  <artifactId>helloscoutjs</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>My Application</name>

  <modules>
    <!-- Core modules -->
    <module>../helloscoutjs.ui</module>
    <module>../helloscoutjs.api</module>
    <module>../helloscoutjs.core</module>
    <module>../helloscoutjs.data</module>
    <module>../helloscoutjs.persistence</module>

    <!-- App modules -->
    <module>../helloscoutjs.app</module>
    <module>../helloscoutjs.app.war</module>
    <module>../helloscoutjs.app.dev</module>

    <!-- Test modules -->
    <module>../helloscoutjs.core.test</module>
```

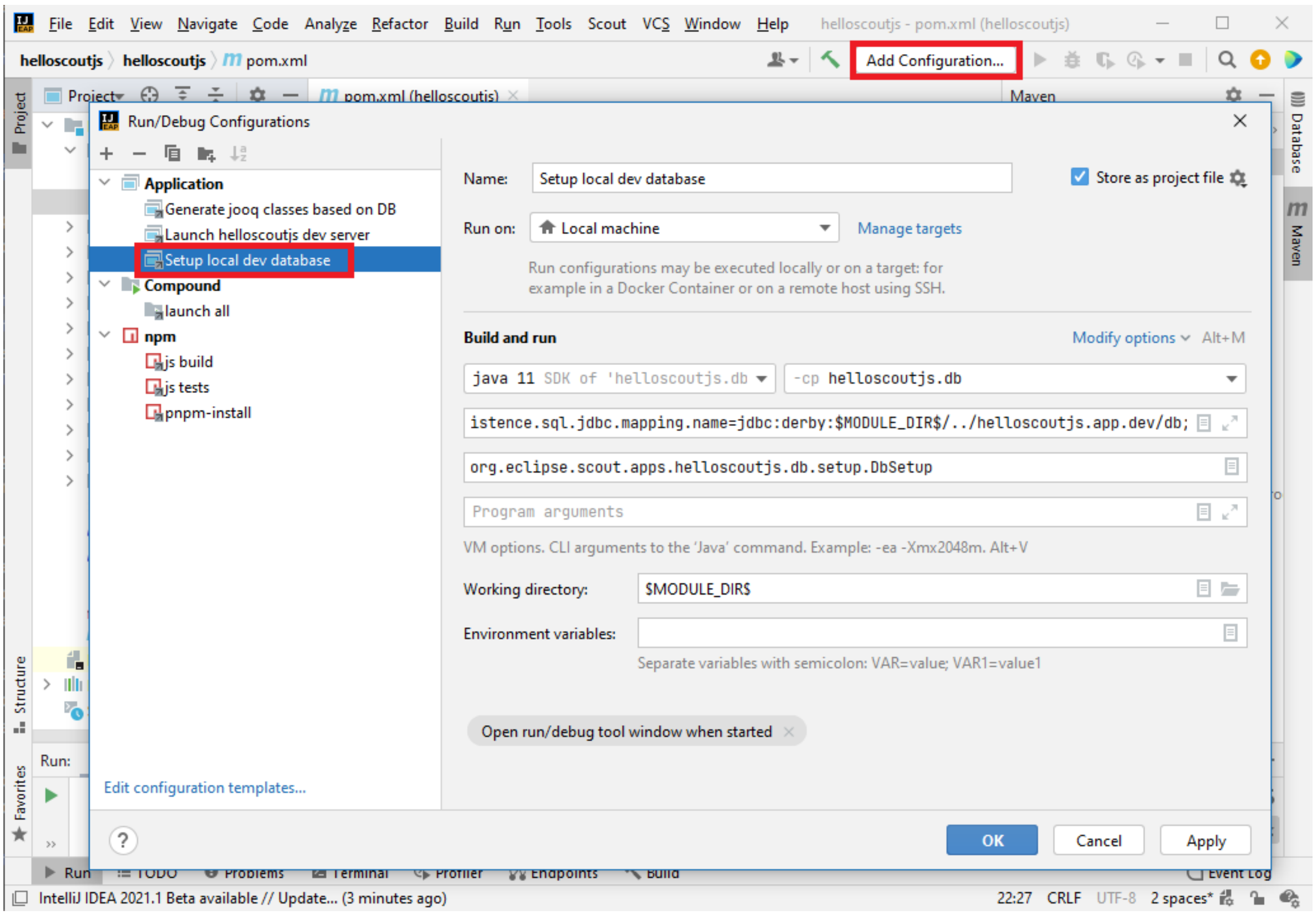
```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Figure 3. The initial set of Maven modules created for the Hello Scout application.

If the modules are not automatically created as indicated in Figure 3, right click on the root `pom.xml` and click `Add as Maven Project`.

The `Hello Scout JS Full Stack` application's backend accesses a local Derby database using jOOQ. For this to work, you first have to setup the local database. This is done be executing the database setup application.

For this click the *Add Configuration…* menu on the top as shown in Figure 4. In the dialog, expand the `Application` type on the left side, select the prepared run configuration `Setup local dev database` and confirm with **[Ok]**. Then click on the green triangle symbol directly right of the *Add Configuration…* menu. This will launch the db setup application which creates a new Derby database in the `helloscoutjs.app.dev/db` folder.


File Edit View Navigate Code Analyze Refactor Build Run Tools Scout VCS Window Help
helloscoutjs - pom.xml (helloscoutjs)
helloscoutjs 〉 helloscoutjs 〉 pom.xml
Add Configuration...
Run/Debug Configurations
Application
Generate jooq classes based on DB
Launch helloscoutjs dev server
Setup local dev database
Compound
launch all
npm
js build
js tests
pnpm-install
Edit configuration templates...
Manage targets
Modify options Alt+M
OK
Cancel
Apply
IntelliJ IDEA 2021.1 Beta available // Update... (3 minutes ago)

Name: Setup local dev database

✓ Store as project file

Run on: Local machine

Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Build and run

```
java 11 SDK of 'helloscoutjs.db'    -cp helloscoutjs.db
istence.sql.jdbc.mapping.name=jdbc:derby:$MODULE_DIR$/../helloscoutjs.app.dev/db;
org.eclipse.scout.apps.helloscoutjs.db.setup.DbSetup
Program arguments
```

VM options. CLI arguments to the 'Java' command. Example: -ea -Xmx2048m. Alt+V

Working directory: $MODULE_DIR$

Environment variables:

Separate variables with semicolon: VAR=value; VAR1=value1

Open run/debug tool window when started

Figure 4. Selecting the db setup run configuration

Now you are ready to start the Scout JS application. This includes downloading the necessary JavaScript dependencies, executing the Java and JavaScript builds and launching the Scout development server. It serves the JavaScript assets to the browser and acts as backend for the REST calls coming from the user interface. You could start each step separately, but for the sake of convenience, there is a compound run configuration available which performs all these tasks (IntelliJ Ultimate only, see below for instructions if using the Community Edition).

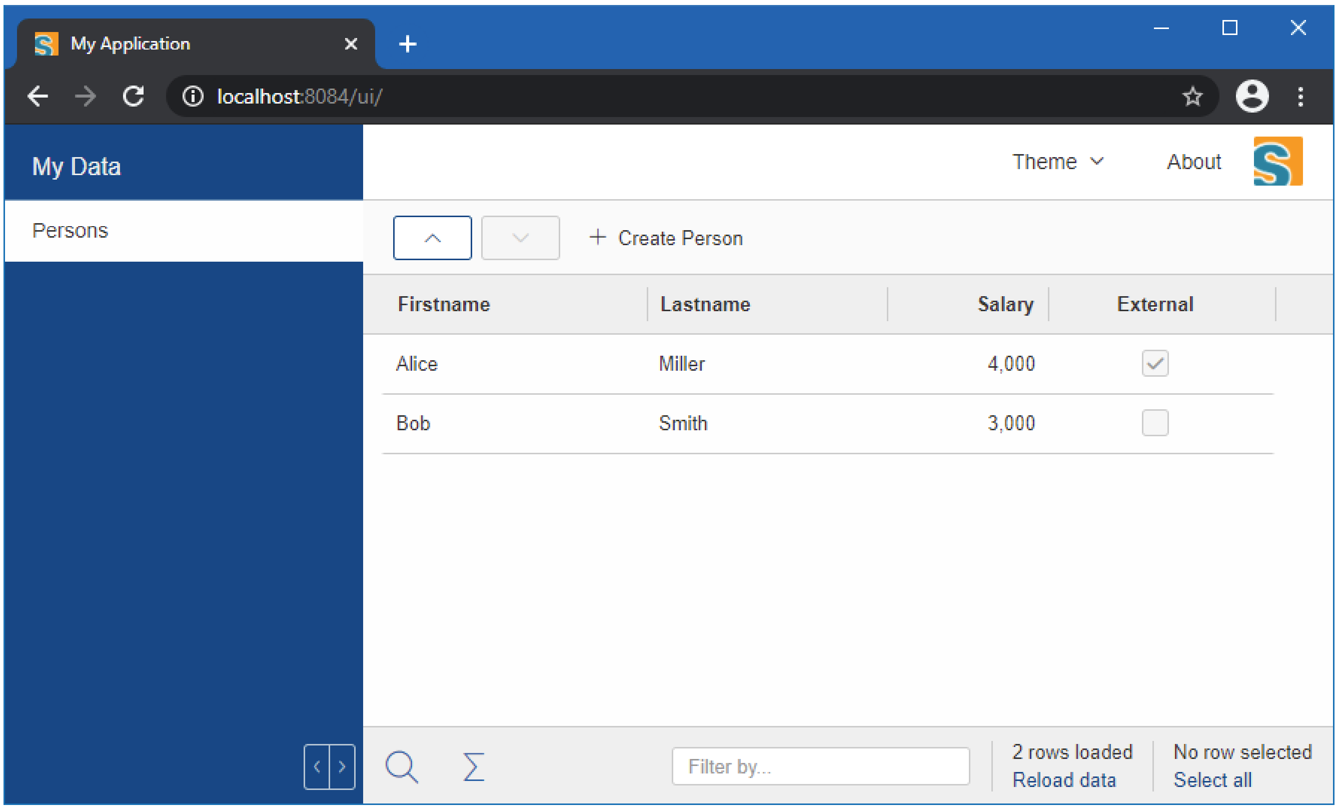
To use it switch the run configuration by clicking the run configuration menu on the top again. Now there is no need to open the dialog as the available run configurations are shown in a dropdown menu. Select the `launch all` compound and run it using the green triangle button.

The JavaScript build fails in case the installed Node.js was not found or is too old. In that case, follow the instructions in the section *Prerequisites* and check the IntelliJ settings in `File | Settings | Languages & Frameworks | Node.js and NPM`.

The `launch all` compound uses JavaScript run configurations which are only available in IntelliJ Ultimate. If using the Community Edition, follow these instructions instead: Execute `npm run pnpm-install` on the command line in the root of your project (next to the `pnpm-workspace.yaml`) to install all JavaScript dependencies. Then execute `npm run build:dev:watch` in the `helloscoutjs.app` module to start the JavaScript build and watcher. The watcher keeps on running and will continuously update the JavaScript assets as you change your JavaScript source files (hot-code-replace). Finally start the run configuration `Launch helloscoutjs dev server`.

Once the JS build has been completed (this may take a while for the first time as some dependencies need to be downloaded) and the server has been started, the `Hello Scout JS Full Stack` application can be accessed by navigating to http://localhost:8084/ in your favorite web browser.

The `Hello Scout JS Full Stack` application is then presented as shown in Figure 5.



Firstname	Lastname	Salary	External
Alice	Miller	4,000	✓
Bob	Smith	3,000	☐

Figure 5. The Hello Scout JS Full Stack application in the browser.

What's Next?

To learn more about Scout JS, we recommend having a look at the other Scout JS Tutorials and the Scout JS Technical Guide.

To see more example code of Scout JS, you should have a look at the Scout JS Widgets application and its source code.

In case you should get stuck somewhere and need help, contact us on the Scout Forum or on Stack Overflow.

Do you want to improve this document? Have a look at the sources on GitHub.