devonfw getting started guide

Updated at 2019-11-22 10:11:37 UTC

Table of Contents

Part I: Getting Started with devonfw

1. Resource Overview

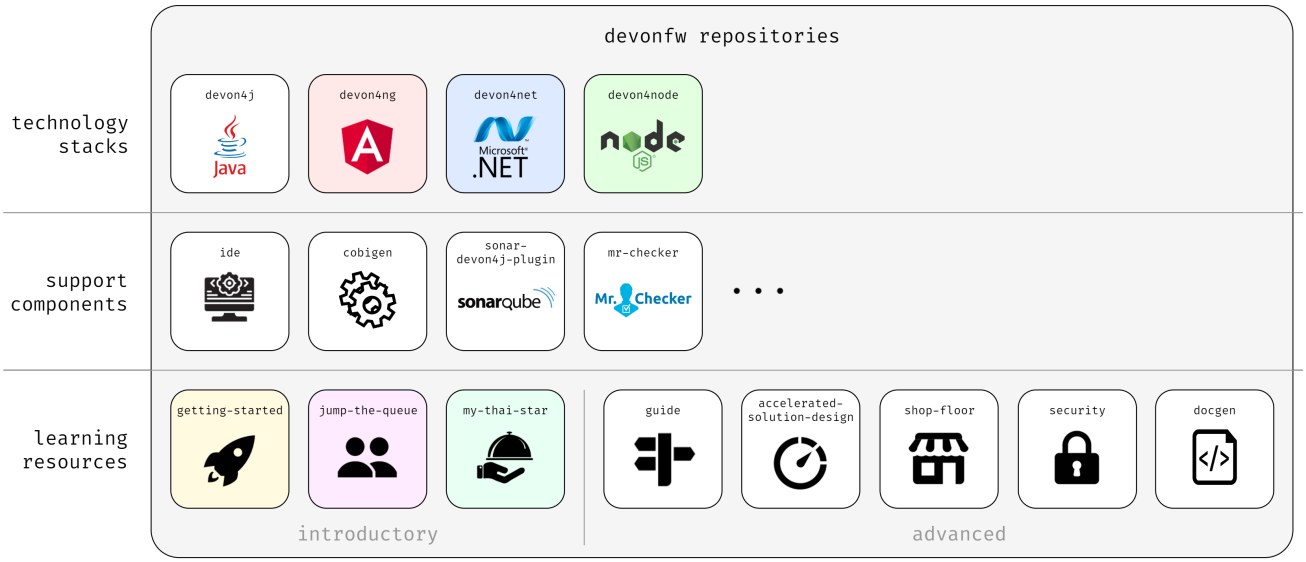
Listed below are the major **devonfw** resources. They're explained in more detail in the following section.

- Repositories
- Distribution
- Guide
- Community
- Contributing
- Homepage

You don't need to follow every link in this section right away. These links mainly exist to provide more in-depth information for a second read-through or if you want to create a bookmark collection.

1.1. Repositories

The GitHub repositories within the devonfw organization contain the source code, documentation and wikis of individual devonfw projects. Projects in early development and prototypes are located in the devonfw forge. They usually remain there until they are ready for a broader release or for use in production.



An overview of the devonfw organization repositories.

1.2. Distribution

The devonfw distribution contains scripts that will help you manage your projects as well as the software packages used by devonfw. It is provided in the form of a `.tar.gz` archive (we recommend 7-Zip to unpack this archive on Windows).

1.3. Guide

The devonfw guide is a PDF document with over 900 (!) pages. It represents the totality of devonfw's documentation. It is always up-to-date (due to being automatically generated) and included in each release of the distribution.

1.4. Community

The devon community is active on Microsoft Teams and Yammer. If you run into any problems with devonfw, have general questions, feedback or suggestions, please feel free to post them in the respective sections there.

1.5. Contributing

If you have gotten to grips with devonfw and want to contribute to the framework, please refer to the contributing guide first. Even finding and fixing errors in the documentation — wheather that be spelling, grammatical or logical errors — can be a huge help for our open source effort.

1.6. Homepage

The official homepage represents a major public presence for devonfw and is meant to attract new framework users as well as customers for Capgemini. It also hosts video tutorials for new developers.

2. Framework Introduction

To gain a better understanding of what devonfw is all about and why it was initially developed, please read the framework introduction in the devonfw guide:

What is devonfw?

In the following section, we will only offer a brief overview over the main stacks and tools included in devonfw and provide you with links for further reading.

2.1. devon4j

The *backend* of most devonfw applications is built with Java. To speed up the development process, the devon4j stack provides pre-selected frameworks and tools that ensure a secure backend-design which conforms to current standards.

To learn more about devon4j, please refer to the devon4j wiki.

2.2. devon4ng

The *frontend* of most devonfw applications is based on the Angular framework. To speed up frontend development, devon4ng provides several Angular application templates as a starting point for new projects. Furthermore it contains code samples that demonstrate important frontend features, such as routing, theming and internationalization.

To learn more about devon4ng, please refer to the devon4ng wiki.

2.3. CobiGen

"The *Code-based Incremental Generator* CobiGen is build as an extensible framework for incremental code generation." New devonfw users will most likely interact with the CobiGen Eclipse plug-in to automatically generate Java classes based on certain database structures and their respective entity classes (as demonstrated in the JumpTheQueue tutorial).

To learn more about CobiGen, please refer to the CobiGen wiki.

3. The devon IDE

First we have to clarify what we are talking about, when we mention the **devon IDE**:

The devon IDE is *not* one monolithic program that is installed with a traditional executable; rather it's a collection of scripts which are invoked via command line to automate several, repetetive development tasks. These scripts then interact with other tools (e.g. Maven, Jenkins), frameworks (Spring, Angular) or third party IDEs (Eclipse, VS Code, IntelliJ IDEA) to streamline the development workflow.
As such, the package size of the IDE is initally very small, the setup is simple, the included software is portable and the whole environment is unobtrusive and light on system resources.

3.1. Setup

Please refer to the setup guide in the devon IDE documentation to learn where to download and how to setup the development environment.

3.2. Usage

Please refer to the usage section in the devon IDE documentation to learn about how to manage workspaces and create launch scripts.

3.3. Commands

Please refer to the commandlets section in the devon IDE documentation to learn more about the `devon` command and available sub-commands.

3.4. Further Reading

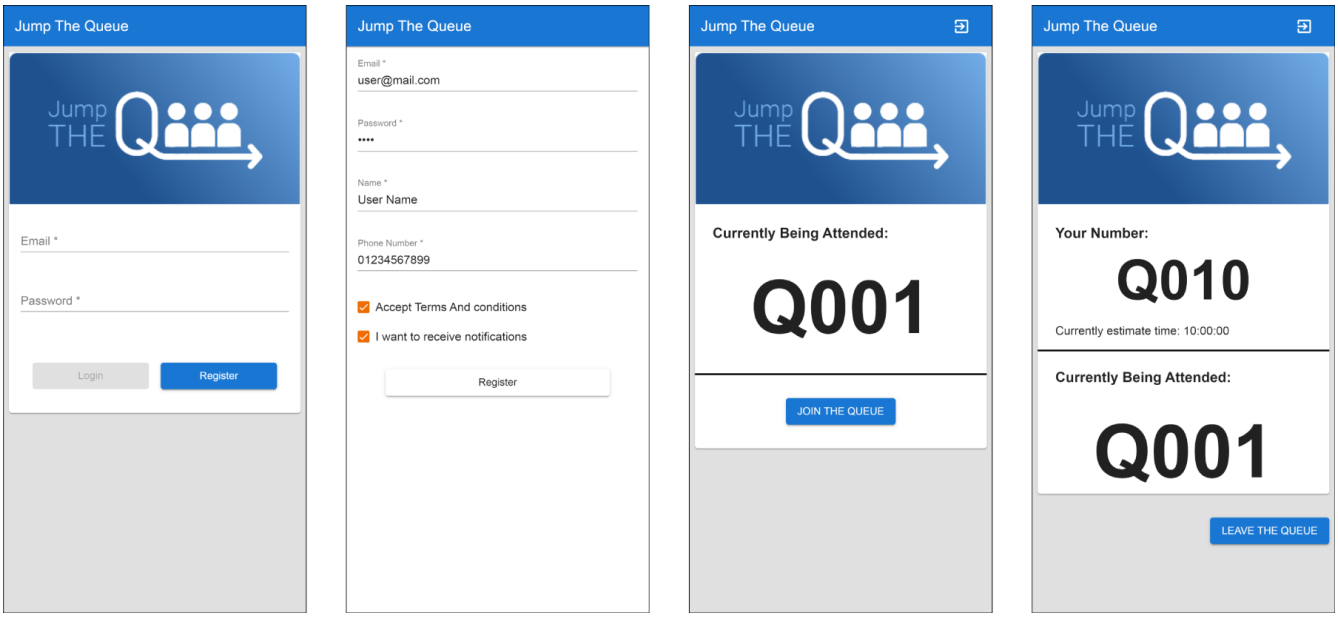
To gain more insight into the motivation behind the devon IDE, you can read the feature section of the documentation.

4. JumpTheQueue Tutorial App

JumpTheQueue is a small application, which you can build yourself by following a step-by-step tutorial. This will teach you more about the components, layers and workflow of devonfw.

The tutorial is located in the documentation of the JumpTheQueue repository. Your entry point will be the the wiki home page.

The tutorial assumes you have successfully completed the IDE setup previously.



Screenshots of the JumpTheQueue App.

5. MyThaiStar Sample App

MyThaiStar is a more complex sample app, that demonstrates the full capabilities of the devonfw stack and framework.

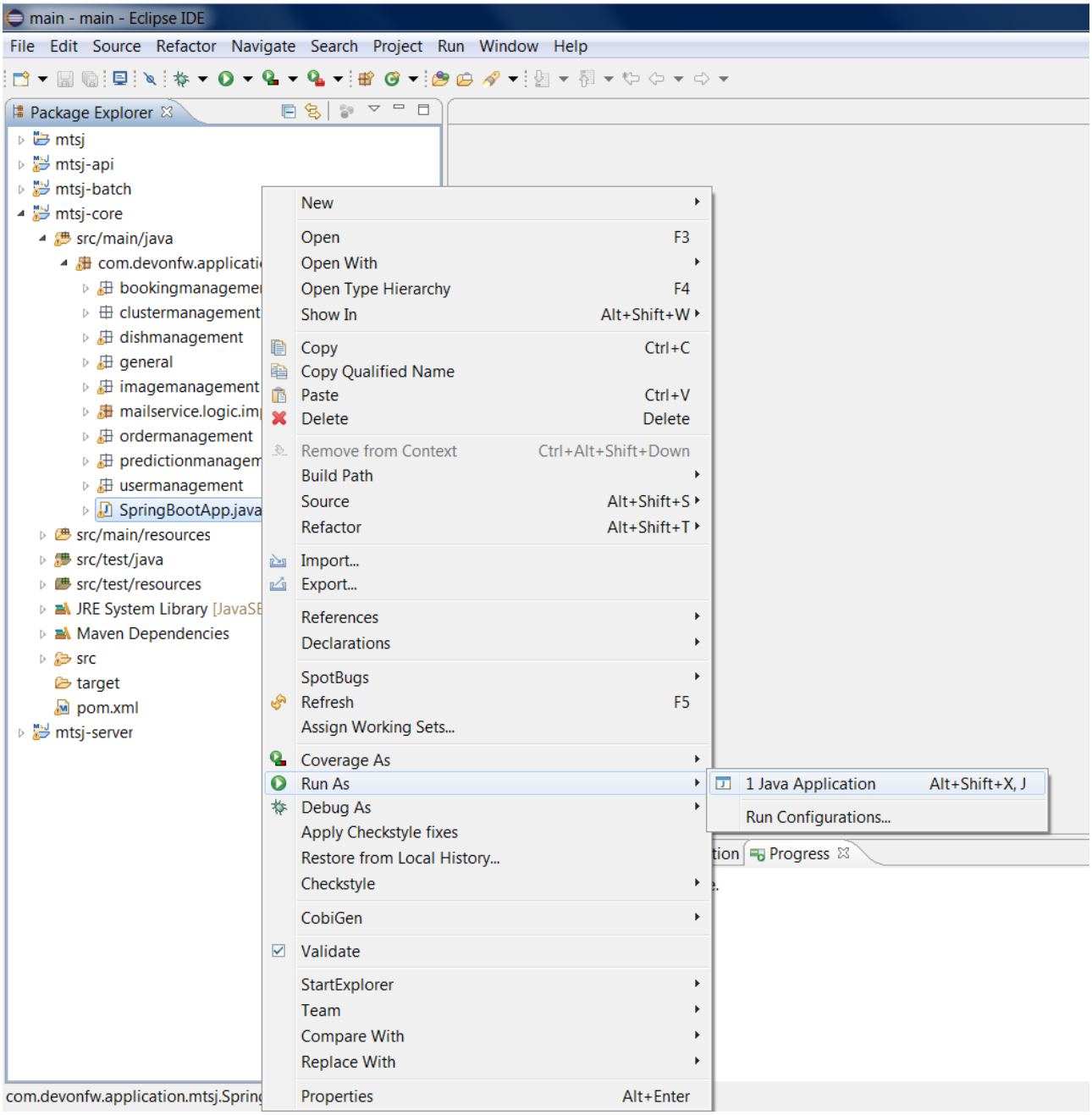
You should take a look at the project structure and familiarize yourself with it, since most devonfw projects follow this exemplary implementation. On this page we will describe how to launch the app on your system, so you can explore the different functionalities it offers.

We assume you have successfully completed the IDE setup previously.

1. In the root directory of the devonfw distribution, right click and select "**Open Devon CMD shell here**" from the Windows Explorer context menu. Then navigate to the main workspace and checkout the MyThaiStar Git repository like this:

```
cd workspaces/main
git clone https://github.com/devonfw/my-thai-star.git
```

2. Perform: `cd my-thai-star`
3. Execute: `devon eclipse ws-up`
4. Execute: `devon eclipse create-script`
5. Go to the root folder of the distribution and run `eclipse-main.bat`
6. In Eclipse navigate to `File > Import > Maven > Existing Maven Projects`, then import the cloned project from your workspace by clicking the "Browse" button and selecting `/workspaces/my-thai-star/java/mtsj/`.
7. Run the backend by right-clicking `SpringBootApp.java` and selecting `Run as > Java Application` in the context menu. The backend will start up and create log entries in the Eclipse Console tab.



8. Return to your command shell and perform: `cd angular`
9. Execute: `npm install`
10. Execute: `ng serve`
11. Once started, the frontend will be available at localhost:4200/restaurant. Login with the username and password "**waiter**" and take a look at the various functionalities provided by MyThaiStar.

6. Further Information

6.1. VS Code Extension Pack

The devonfw Platform Extension Pack is a collection of useful extensions for Visual Studio Code that support the development of devonfw applications. You can get it here.

Please be aware that the Extension Pack is quite 'extensive' and will take a while to download and install. It might also slow down VS Code depending on your system. To avoid this you can take a look at the included extensions and install them individually if needed.

6.2. SonarQube Plug-in (Architecture Validation)

The devon4j plug-in for SonarQube can be used to validate if a certain project follows the devonfw architecture blueprint. Please refer to the setup guide in the respective repository, if you want to start using this plug-in.

6.3. MrChecker Framework (E2E Testing)

The MrChecker framework for end-to-end testing is supplied in the form of a Maven project which you can extend with your own test cases. It supports the classic JUnit, as well as the Cucumber test structure. Please visit the MrChecker wiki to learn more.