Devonfw

Backlog item & work package definition

Issue	Devcon GUI
Target	Devcon
Milestone	V2.1.0
Type/reference	Backlog item - <<Needs Teamforge ID>>
Estimation	6 days
Timebox	Sprint 1, 2
Initial date	08-08-2016
Revision	0.2

Work package | Task definition

Devcon is included In release v.2.0.1 of Devonfw. Devcon is a cross-platform command line tool running on the JVM that provides many automated tasks around the full life-cycle of Devon applications, from installing the basic working environment and generating a new project, to running a test server and deploying an application to production.

Devcon is the easiest way to use Devon. With a focus on project automation, easy command execution and declarative configuration, it gets out of your way and lets you focus on your code.

For details, see the User Guide: (https://github.com/devonfw/devon/wiki/devcon-user-guide

Devcon is a command-line tool. This task consist of providing the program with a simple Graphical User Interface (GUI) which can be used as an alternative to the command-line interface.

This task entails all activities required to realize the following deliverables according to defined functional requirement.

Deliverables

The devcon binary, devcon.jar, extended in such way that it can be started with a GUI.

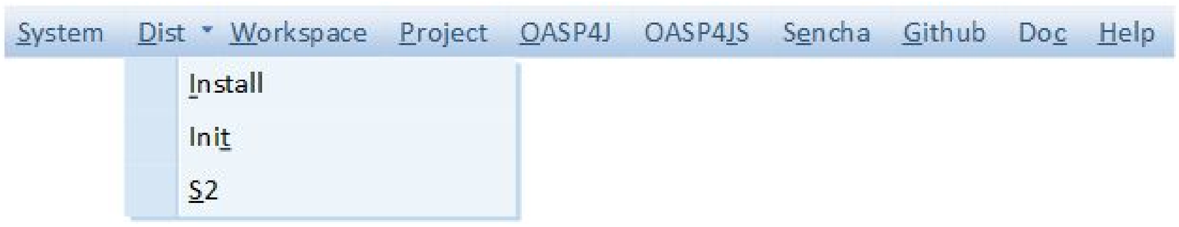
Functional specification

The devcon binary, devcon.jar, SHOULD have another parameter, -g (for “gui”), which will start up a graphical user interface (GUI). Any other parameter should be ignored once -g is given.

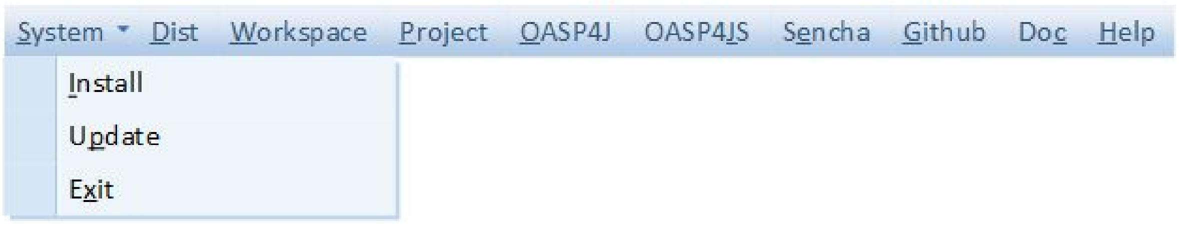
This should be an empty form with a main menu generated from the information as stored in the CommandRegistry component. Modules MUST be arranged horizontally on top of the form with their respective commands in the submenu.

These commands, unlike the current implementation, where modules/commands are sorted alphabetically, MAY be displayed according to an “order” attribute on the CmdModuleRegistry & Command annotations. If commands or parameters do not have a “sort” attribute, then the alphabetically sorted list of these SHOULD be appended to the “sorted-by-attribute” ones.

Example:



There is one exception to the above. The first menu (module) MUST have an “Exit” option added to the bottom of its sub-menu. NOTE: this DOES NOT necessarily have to be the “System” menu as depicted below. It MUST be the first menu. This option “E_x_it” MUST terminate the application. This can be done outside of the context of the CommandModuleRegistry (it MUST not affect the console version of devcon). Example:



Once a command has been chosen from the menu, the main form should display:

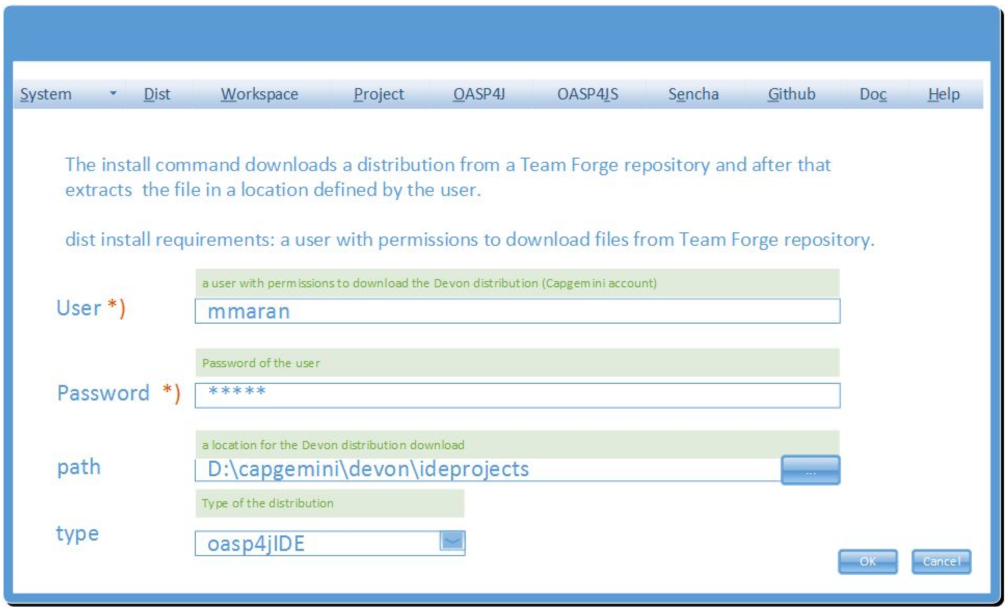
- The command description/help text on top of the form (with a optional scrollbar when there's too much text)
- the user controls as defined by the parameters set on the command.
- With optional help descriptions on the controls

These controls MAY be displayed according to an “order” attribute on the Parameter annotation. If a Parameters does not have a “sort” attribute, then the alphabetically sorted list of these SHOULD be appended to the “sorted-by-attribute” ones.

There are a number of particular control types. They need to be specified with the “input-type” annotation. The underlying table lists all of them:

input-type	Description; will result in
generic	Plain text field
path	Text field with “file selector” button (showing a file selector dialog box when pressed)
password	Text field with password mask (“***”) so the password cannot be read from the screen
pulldown	List of values (must be configurable through the annotation! SHOULD NOT be set programmatically)

This will result in a screen like you can see here below.



The install command downloads a distribution from a Team Forge repository and after that extracts the file in a location defined by the user.

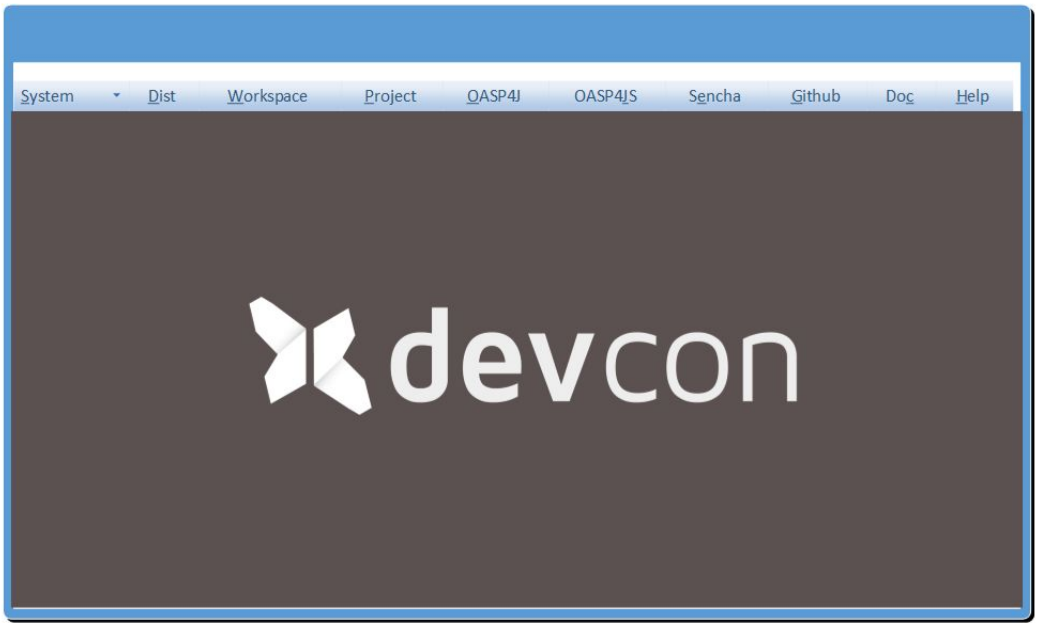
dist install requirements: a user with permissions to download files from Team Forge repository.

	a user with permissions to download the Devon distribution (Capgemini account)
User *)	mmaran
	Password of the user
Password *)	*****
	a location for the Devon distribution download
path	D:\capgemini\devon\ideprojects
	Type of the distribution
type	oasp4jIDE

Mandatory fields MUST carry a distinct visual token that they are optional. An asterisk (“*”) behind the field name should suffice. Alternatively, another colour scheme might do as well.

Design

The colours as used in the application are defined in the file Logo_Devcon-colours.jpg .
The background of the main form when there is NO command selected, should be the file Logo_Devcon-background.png.



(ask the project lead for the files)

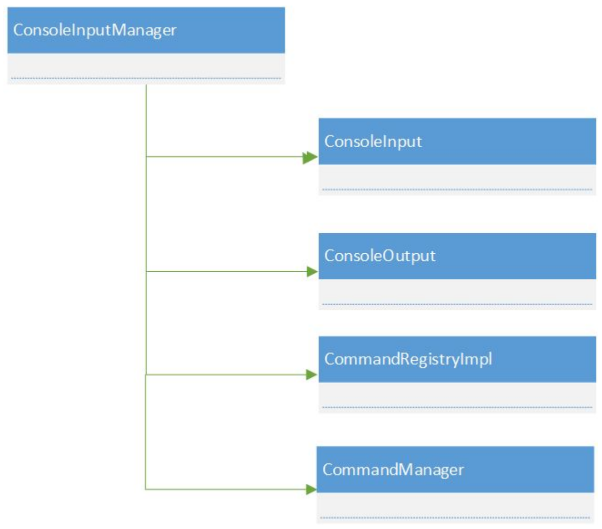
Requirements

- The GUI SHOULD be implemented through **JavaFX**
- It SHOULD work on JRE/JDK 1.7 & 1.8
- The menus MUST be dynamically generated in the same way as the command line interface (apart from where noted in the functional specification)
- The menu and form controls MUST be ordered through an extension of the CmdModuleRegistry, Command & Parameter annotations. This option SHOULD be extended to the console interface as well.
- The menu items SHOULD have the option to define shortcut keys (to be shown in the menu) through an extension of the Command annotation (this SHOULD NOT be replicated in the console interface as this would serve no purpose)
- The input controls SHOULD be able to offer specific functionality. Through an extension of the Parameter annotation it SHOULD be possible to define the input type, like date etc (defined in the functional specification)

- The current API within a command MUST NOT be subject to change because of any change to the specification.

Technical design

The current Console application is formed around this informally specified class diagram (not valid UML).

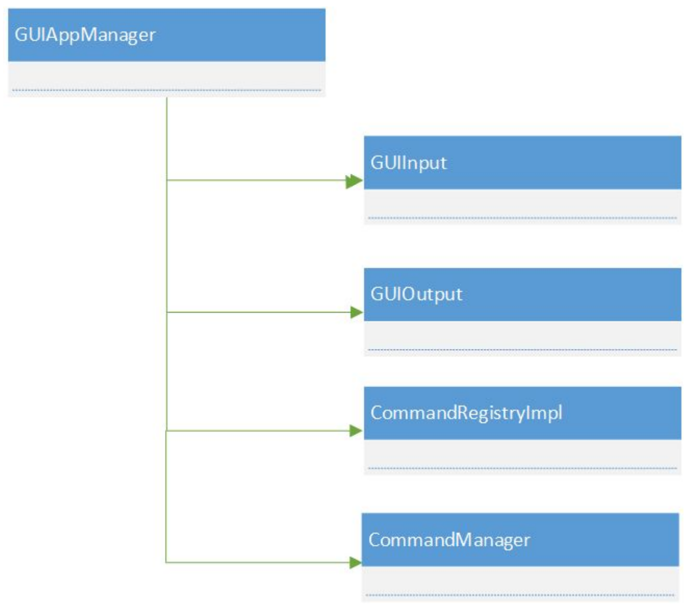


From the Devcon main class, the class ConsoleInputManager is instantiated. The is the real “main” class of the console application. It is the component responsible for parsing and processing the command line parameters. After parsing and validation of the command line parameters, control is passed to the COmmandManager.

The CommandManager is the central orchestrating unit within Devcon. It is responsible for executing commands. These commands are obtained from the CommandRegistry. This is the central repository where all CommandModules with their respective Commands are loaded and stored.

The CommandManager reads input from and writes output to the console. In the console app these are ConsoleInput and ConsoleOutput, based on the Input and Output interfaces respectively, which define high-level abstractions.

In the gui version the ConsoleInputManager should check for the -g parameter and launch the class GuiAppManager. This class should function akin to the former, with the difference that there´s no “single phase” where parameters are being parsed and executed by the CommandManager.



Rather, the GUIAppManager does the following concrete steps at start-up:

1. Menu should be build up with information by query from the CommandRegistry.
2. The generated menu item do hold a reference to the command, and, at click of a menu item, the method GUIAppManager#showCommandForm will be called with that reference being passed.
3. GUIAppManager#showCommandForm can autodiscover the information in the CommandRegistry and can dynamically generate and display varying ”things”.
4. Only after click on the “OK”” button will the command being generated.

Note: the GUIInputImpl class which implements the Input interface can for the moment remain “empty” as it currently will be used by the CommandManager (planned for a future release). However, it should already be facilitated for by the GUIAppManager (passed to constructor, assigned to field).

References

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. See: https://www.ietf.org/rfc/rfc2119.txt