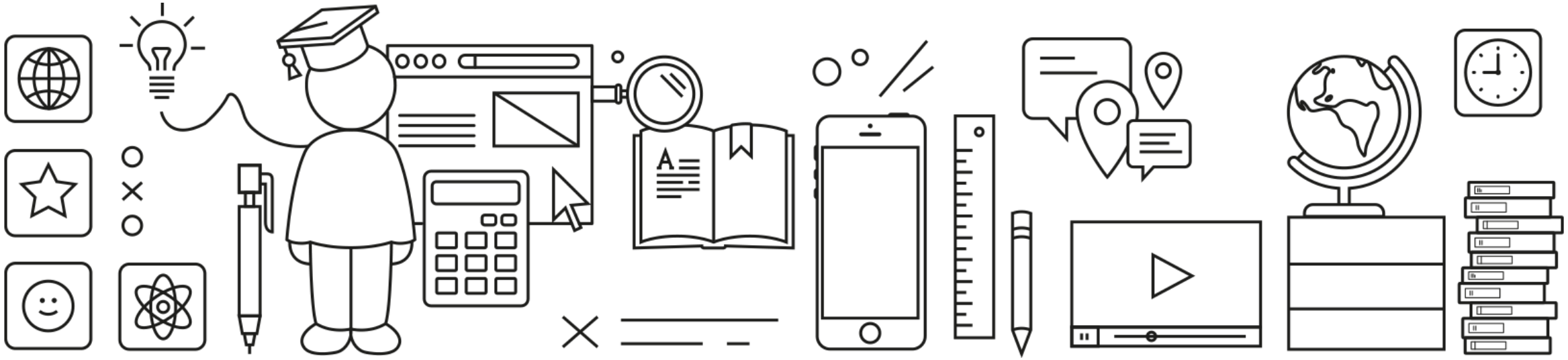
SAP Customer Experience

SAP Commerce Cloud Backoffice Framework Developer Training

Development Environment

THE BEST RUN SAP



Environment Set Up

Environment Set Up
Rapid Prototyping
Hot Deployment
Training Labs Tool
Exercise

Use an IDE

We recommend using an IDE

- Here, we are using Eclipse Oxygen *Java Developer* Edition

Import all the required extensions to your workspace:

- `config`
- `platform`
- `auditreportservices`
- `backoffice`
- Your new, ***custom*** Backoffice extension

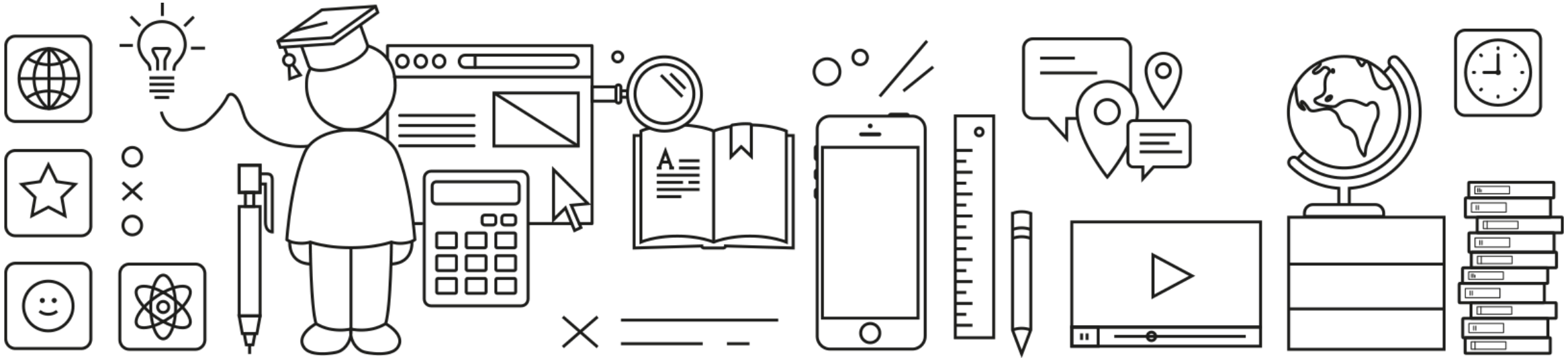
IntelliJ IDEA

Set Up Ant

- **Point the Ant classpath to the one that comes with the platform**
  - `${HYBRIS_BIN_DIR}/platform/apache-ant-*`

The platform's build targets are in

`${HYBRIS_BIN_DIR}/platform/build.xml`



Rapid Prototyping

Rapid Prototyping

Rapid Prototyping:
The process of quickly and easily creating prototypes of a product at early stages of a project to develop ideas, get feedback, and work towards the final product.

The Backoffice framework provides you with some tools for rapid prototyping.

Breadboard Widget

A widget that's installed on the Backoffice *application*

Simulates sending input data to the widget

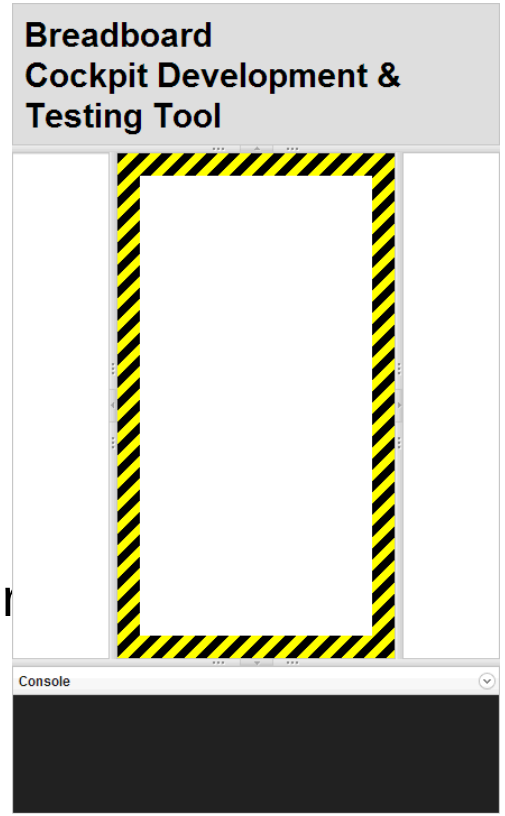
Provides the output data as well

To access it, first login to Backoffice, then go to URL:

https://localhost:9002/backoffice/?mode=breadboard

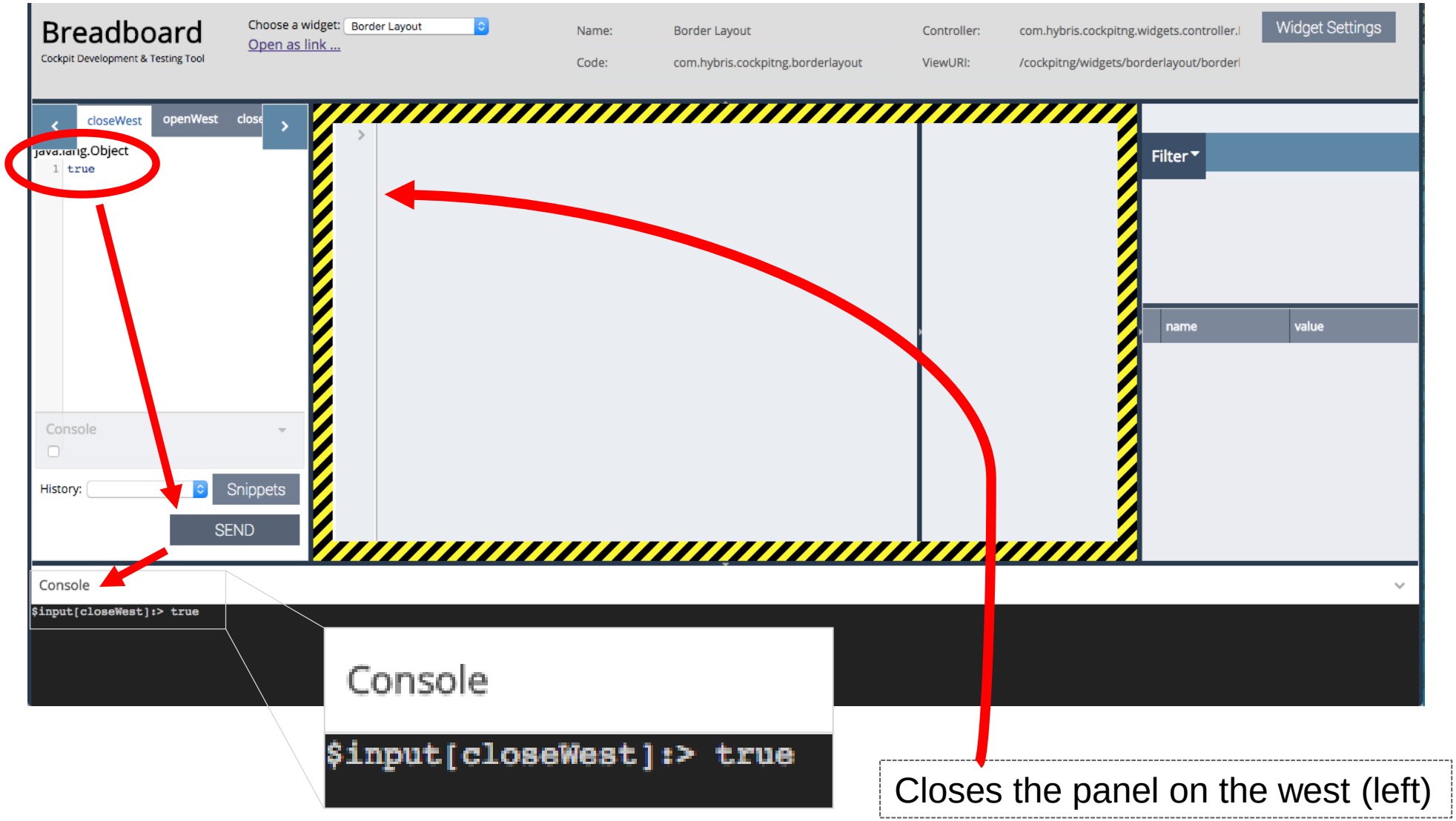
(Note: Backoffice Framework development mode must be enabled via the pr
backoffice.cockpitng.development.mode=true

Exit via URL https://localhost:9002/backoffice



Demo

Breadboard in Action!



Closes the panel on the west (left)

Disabling Caching of Components and Controllers

You can disable caching to force reloading of components at every page refresh

Quite handy *during development* – quickly and conveniently see your changes to component views!

Set the following properties inside `local.properties`

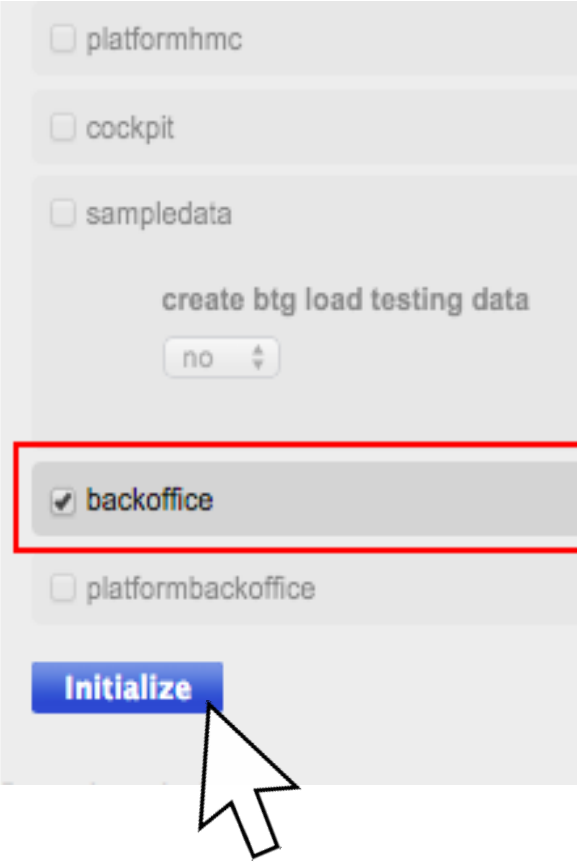
`${HYBRIS_CONFIG_DIR}/local.properties`

```
backoffice.cockpitng.uifactory.cache.enabled=false
backoffice.cockpitng.widgetclassloader.resourcecache.enabled=false
backoffice.cockpitng.resourceloader.resourcecache.enabled=false
```

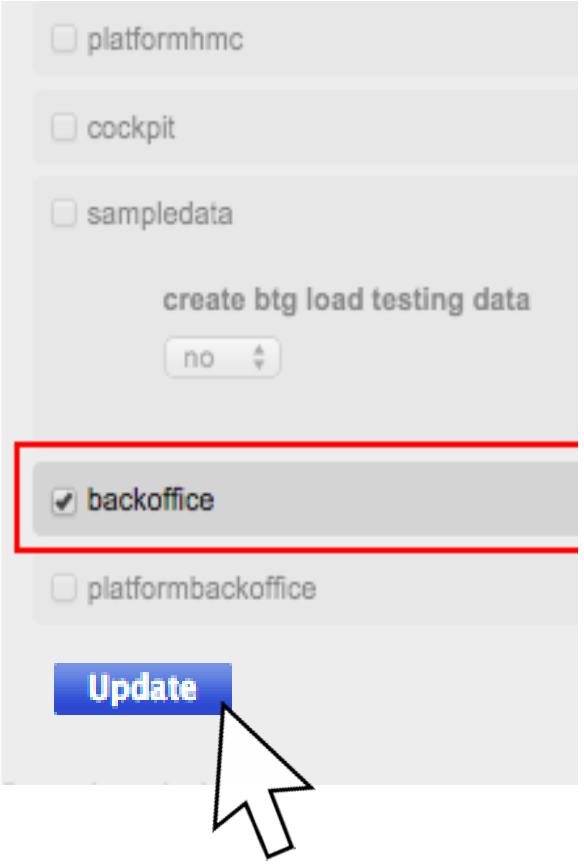
Resetting Backoffice Configuration (Manual Invocation)

Via the Administration Console (HAC)

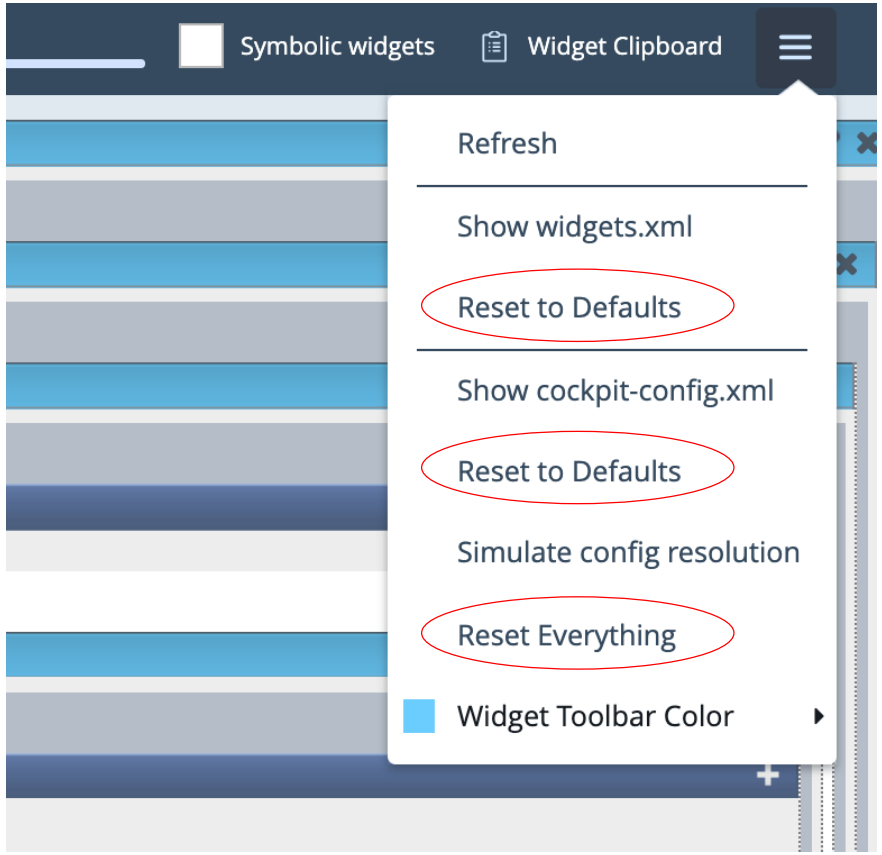
Initialize:



Update:



Via the Backoffice Application Orchestrator



Resetting Backoffice Configuration – What it Does

- Two configurations can be reset:
  - **`widgets.xml`** – Backoffice will ***discard*** all current in-memory widget instances:
    - Runtime changes (via Application Orchestrator or programmatically) are lost
      - Each widget loses its ***widget settings***
      - Backoffice discards entire widget-composition structure (widgets nesting into slots)
    - All widget instances, settings, and compositional structure are re-created by re-reading all **`*-backoffice-widgets.xml`** and **`*.zul`** files from each Backoffice extension
  - **`cockpit-config.xml`** – Backoffice discards all in-memory context-dependent widget configurations (mostly informing each widget instance on how to render):
    - Runtime changes (via Application Orchestrator or programmatically) are lost
    - All widget ***context configuration*** (XML) components are re-read into memory from the **`*-backoffice-config.xml`** files of each Backoffice extension

Resetting Backoffice Configuration (Auto-Triggered)

- **Auto-triggering of resets can be convenient and can save time**

Key	Description	Possible values
`backoffice.cockpitng.reset.triggers`	list of triggers that should induce a reset	`login,start`
`backoffice.cockpitng.reset.login.scope`	list of ***what*** should be reset, if *login* trigger occurs	`widgets,cockpitConfig`
`backoffice.cockpitng.reset.start.scope`	list of ***what*** should be reset, if *start* trigger occurs	`widgets,cockpitConfig`
`backoffice.cockpitng.reset.scope`	list of ***what*** should be reset, for *both* triggers (**cannot** be used in combination with trigger-specific scopes)	`widgets,cockpitConfig`

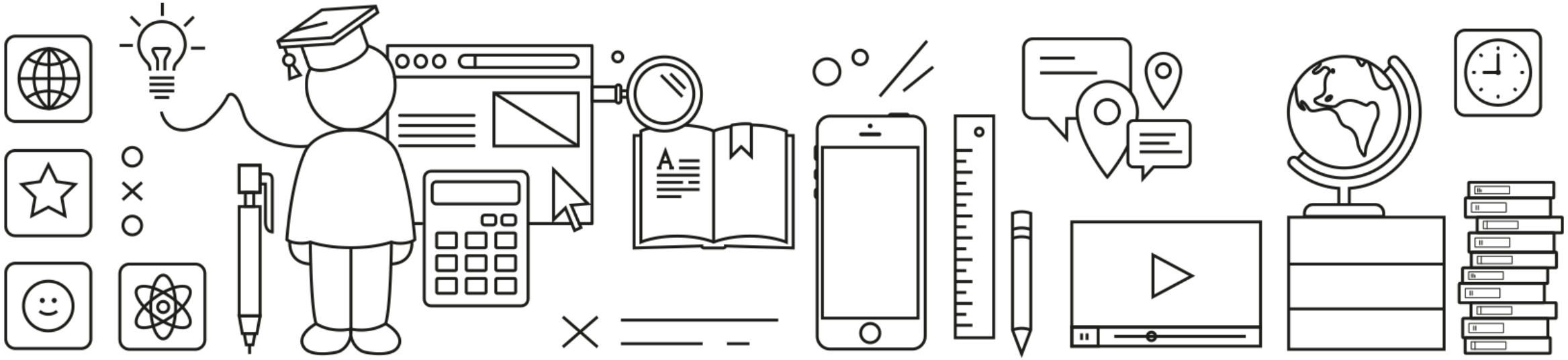
Examples:

`${HYBRIS_CONFIG_DIR}/local.properties`

```
backoffice.cockpitng.reset.triggers=login,start
backoffice.cockpitng.reset.scope.start=widgets
backoffice.cockpitng.reset.scope.login=cockpitConfig
```

`${HYBRIS_CONFIG_DIR}/local.properties`

```
backoffice.cockpitng.reset.triggers=login
backoffice.cockpitng.reset.scope=widgets,cockpitConfig
```



Hot Deployment

Hot Deployment

The ability to deploy executables to a running system without any need for stopping or restarting the system.

Enabling Hot Deployment

First you need to enable it inside of `local.properties`

```
${HYBRIS_CONFIG_DIR}/local.properties
backoffice.cockpitng.hotDeployment.enabled=true
```

After restarting the system, a new button appears in the Backoffice app

Using the Redeploy Button

1. Perform an `ant build` at the command line
2. Press the Redeploy button
   This reloads the JAR files with all Backoffice extensions (including yours) and recreates the spring context
3. Refresh the page!

Tools for Hot Deployment

The *Redeploy* button

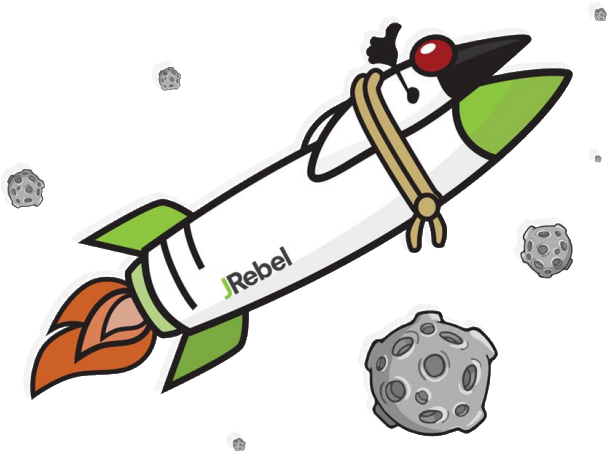
Reloads the JAR files from all Backoffice extensions (including yours) and recreates their spring contexts

Dynamic Code Evolution VM (DCEVM)

A modification of the Java HotSpot(TM) VM that allows unlimited class redefinition at runtime.
http://ssw.jku.at/dcevm/

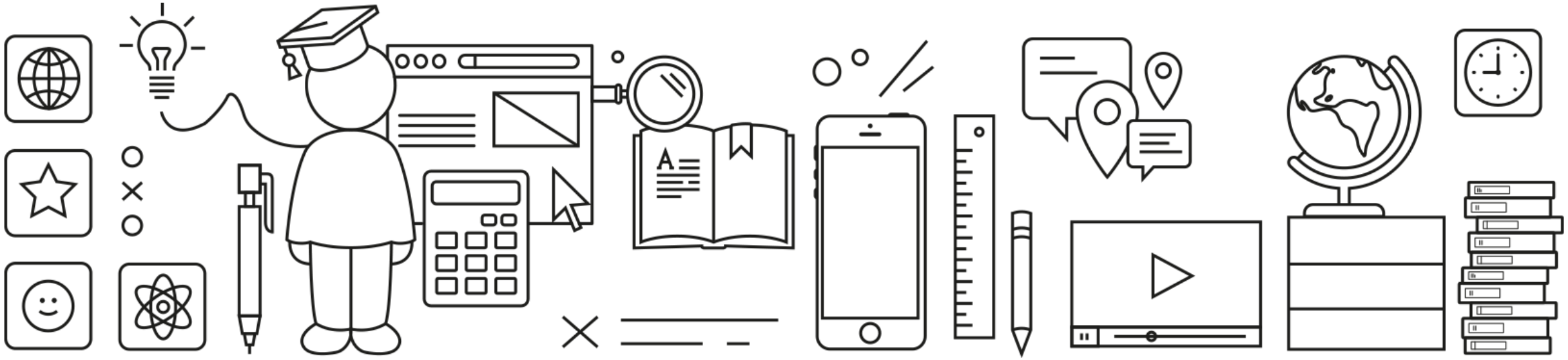
JRebel 6

Using JRebel with the Commerce Platform



Comparison of the Tools

	Java	JVM Hot Swap (HotSpot VM)	Backoffice Hot Deployment (Redeploy button)	DCEVM (Enhanced Debug Mode)	JRebel 6
1	Changes to method bodies	✓	✓	✓	✓
2	Adding/removing Methods	✗	✓	✓	✓
3	Adding/removing constructors	✗	✓	✓	✓
4	Adding/removing fields	✗	✓	✓	✓
5	Adding/removing classes	✗	✓	✗	✓
6	Adding/removing annotations	✗	✓	✓	✓
7	Changing static field value	✗	✓	✓	✓
8	Adding/removing enum values	✗	✓	✓	✓
9	Modifying interfaces	✗	✓	✓	✓
10	Replacing superclass	✗	✓	✗	✓
11	Adding/removing implemented interfaces	✗	✓	adding ✓ / removing ✗	✓
12	Initializes new instance fields	✗	✓	✓	✓
13	Adding/Removing bean	✗	✓	✗ (but planned)	✗ (doesn't work in a Backoffice extension)



Training Labs Tool

TrainingLabsTool

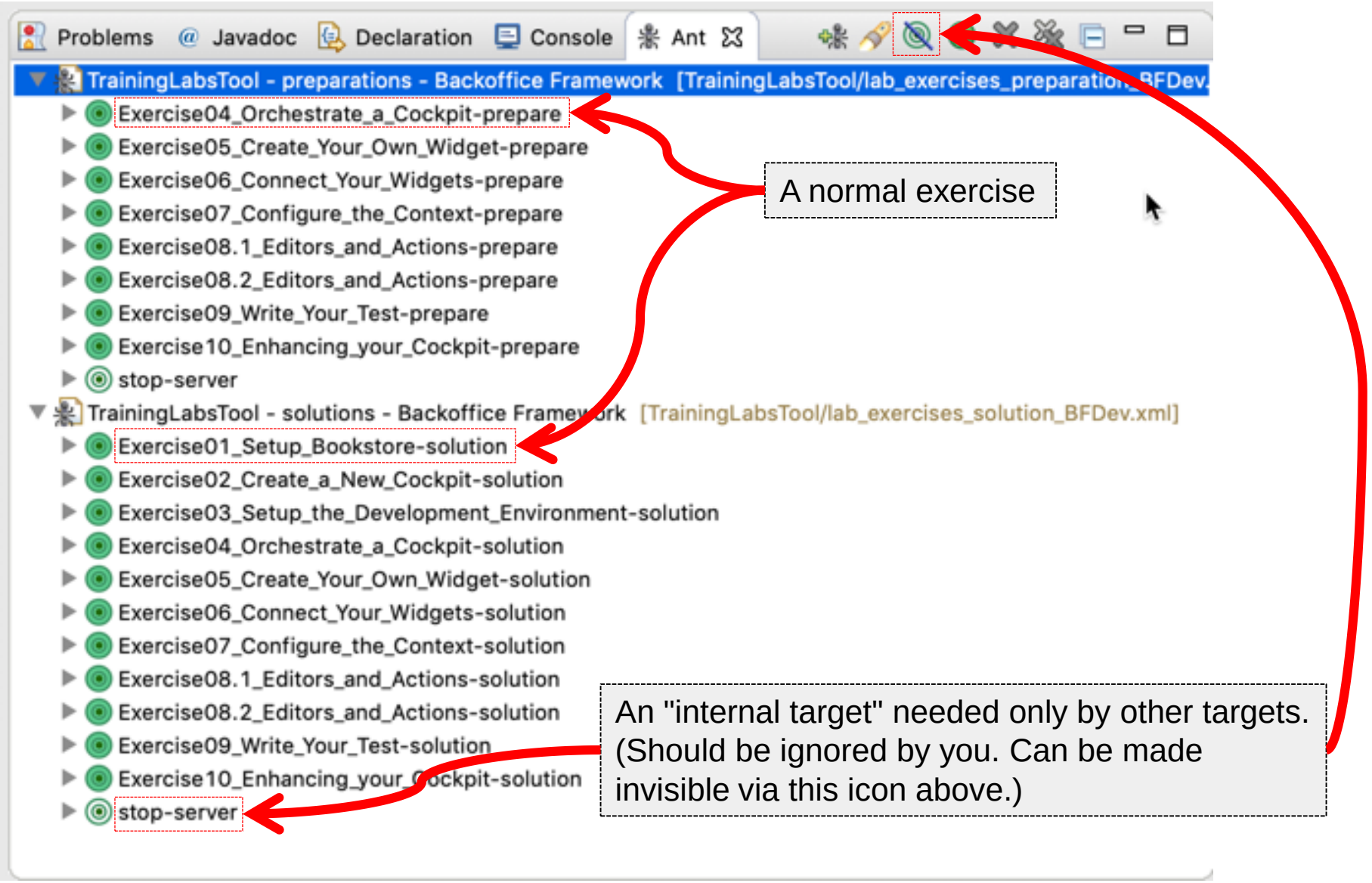
- A special project we made for classroom use
- Automates the exercise setup
- Preinstalled in your build environment with Ant scripts and file resources for pre-exercise preparation (and, if desired, the solution)
- Before each exercise:
  - Run `ExerciseX-prepare` ant target
- For each exercise there is an
  - `ExerciseX-prepare`, and an
  - `ExerciseX-solution`

Ant Scripts

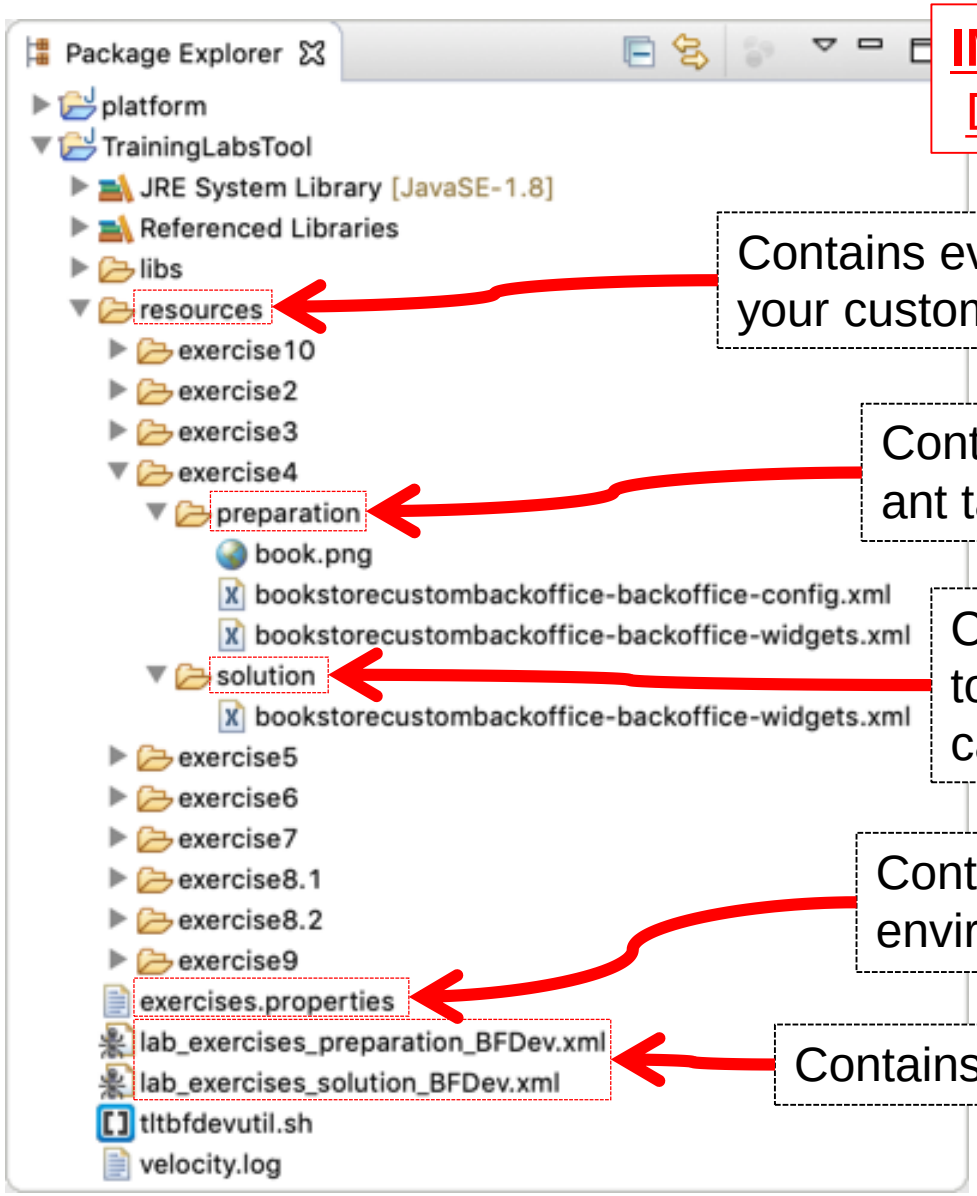
Add these two Ant scripts from the *TrainingLabsTool* project/folder to the *Ant* view inside your IDE:

1. `lab_exercises_preparation_BFDev.xml`
   - includes targets that prepare the environment for each exercise
2. `lab_exercises_solution_BFDev.xml`
   - includes targets that solve the exercise automatically.

Training Labs Tool – Ant View



Training Labs Tool



IMPORTANT Be ***VERY*** careful:
Do not make any changes to ANY file in TrainingLabsTool!

Contains every file the Ant targets (a.k.a. *scripts*) copy to your custom extensions to help you with the exercises

Contains the files needed by a specific *ExerciseXX-*-prepare* ant target to prepare your project(s) for an exercise

Contains the files needed by the solution scripts to provide a solution to an exercise (plus, you can compare your solution with them)

Contains properties/values to adapt TrainingLabsTool to your environment (like where your **platform** project/folder is located)

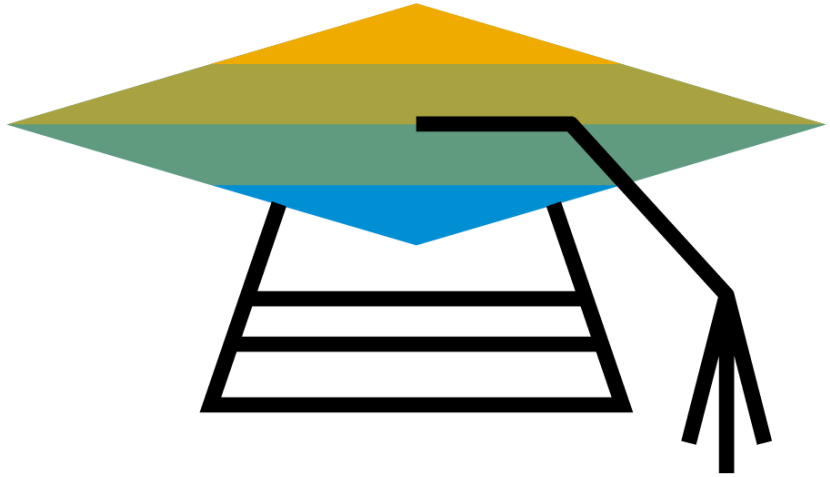
Contains definitions of all TrainingLabsTool **Ant** targets

Exercise 3

Exercise 3 – Development Environment Set Up

- Prepare your IDE
- Disable caching
- Enable automatic configuration reset
- Enable the “Redeploy” button

Thank you.



THE BEST RUN SAP®