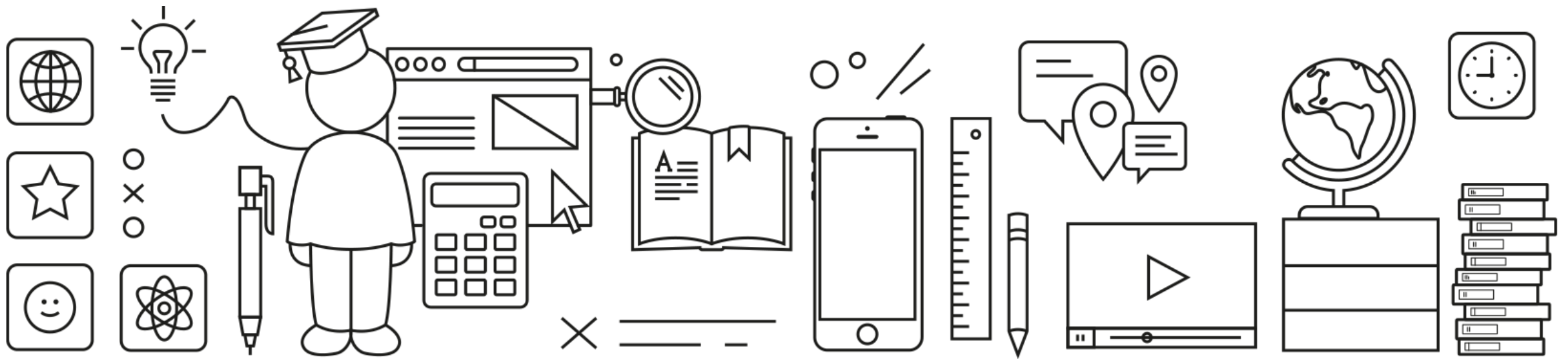




SAP Customer Experience



SAP Commerce Cloud Backoffice Framework Developer Training Backoffice Testing Framework

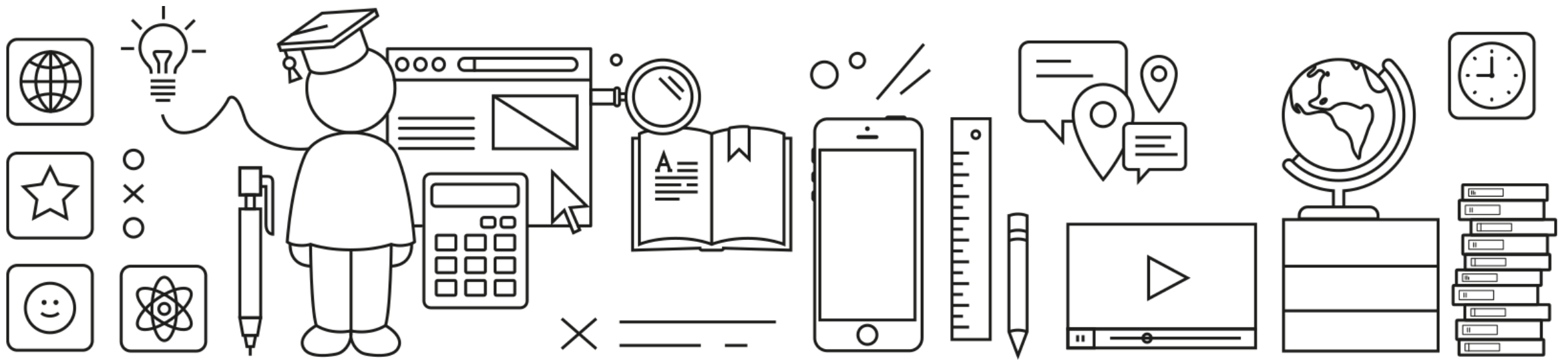


Introduction

Introduction
Unit-Testing Widgets
Annotations
Behavior Testing
Action and Editor Testing
Exercise

Overview

- Supports testing of your custom widgets, editors, and actions
- Supports unit testing
- Based on  JUnit
- Supports  Mockito
- Framework-specific configurations using annotations provided by Backoffice
- Supports integration and black-box testing (using the Breadboard Widget)
- Provides OOTB tests for widget definitions, boundaries, and some code compatibility checks (e.g. existence of an editor's no-arg constructor)



Unit Testing Widgets

- Introduction
- Unit-Testing Widgets**
- Annotations
- Behavior Testing
- Action and Editor Testing
- Exercise

Unit-Testing Widgets

- Unit test should extend `AbstractWidgetUnitTest`
- Intended for TDD - Write the tests and then test the widget
- When the declaration in the test becomes inconsistent with the widget definition, the test will fail
- Encourages use of global constants for socket names
- Test interactions with input and output sockets
- Test interactions with UI elements

AbstractWidgetUnitTest

■ AbstractWidgetUnitTest provides tests checking consistency between the test declaration and the implementation of the following elements

- Sockets
- Commands
- View events
- Global events
- Null-safe check
 - Executed against all declared inputs of the widget
 - Checks for input parameters with null or default values
 - For primitives, it checks some default values, like 0 for int
 - For non-primitives, the default value is the instance created by the no-arg constructor, if one exists
- Extensible fields check
 - Checks if all the fields/attributes in a widget would be accessible by its inheriting children
 - For all fields either declared protected or with a getter method associated with them

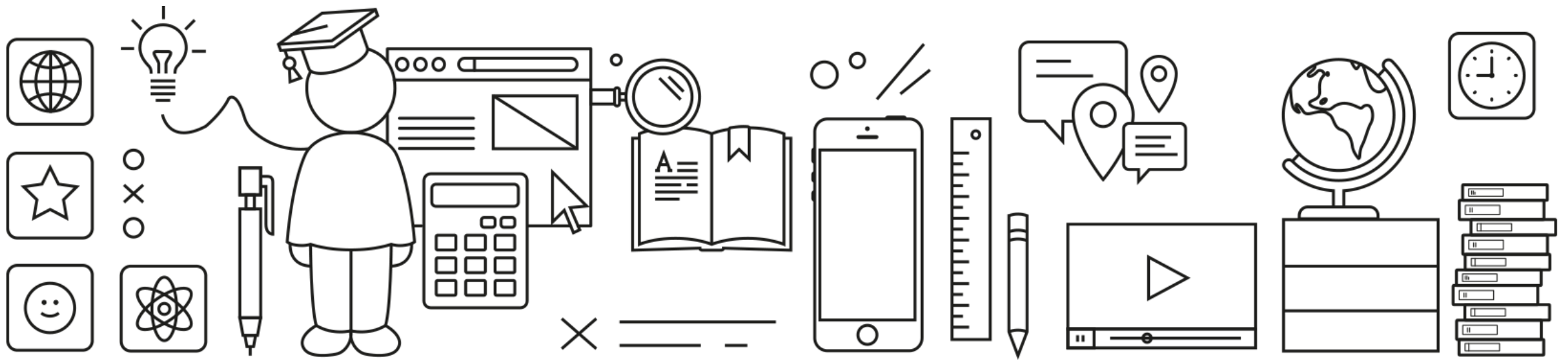
Unit-Testing Widgets

Extend `AbstractWidgetUnitTest<WidgetController>`

DemoWidgetTest.java

```
public class DemoWidgetTest extends AbstractWidgetUnitTest<DemoWidgetController>
{
    private final DemoWidgetController controller = new DemoWidgetController();

    @Override
    protected DemoWidgetController getWidgetController()
    {
        return controller;
    }
}
```



Annotations

Introduction
Unit Testing Widgets
Annotations
Behavior Testing
Action and Editor Testing
Exercise

Annotations

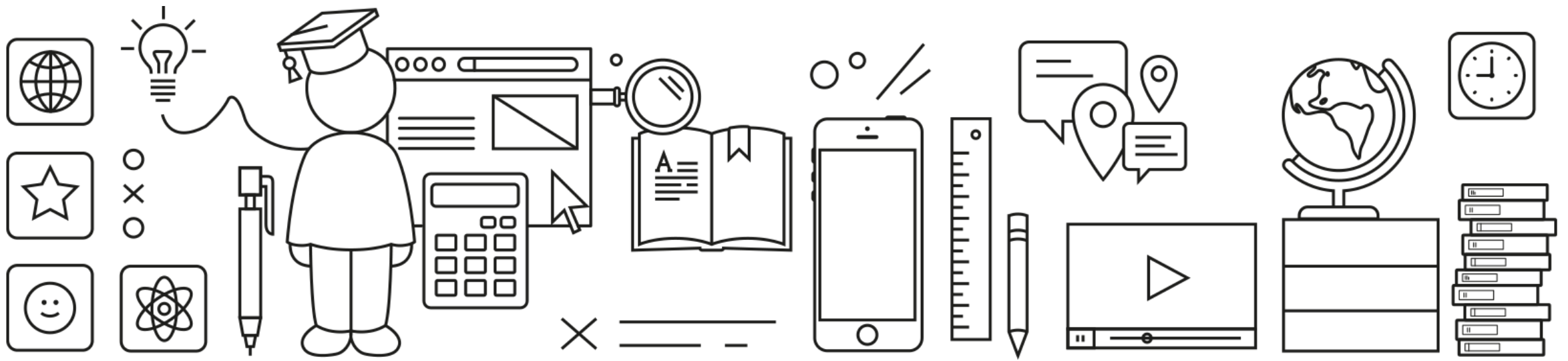
- Annotations provide fixture declarations of specific Backoffice runtime elements (View, Socket, Global)
- `@DeclaredInput`, `@DeclaredViewEvent`, `@DeclaredGlobalCockpitEvent`, `@NullSafeWidget`

```
@DeclaredInput(value = DemoWidgetController.IN_SOCKET, socketType = String.class)
```

- When having more than one: `@DeclaredInputs`, `@DeclaredViewEvents`, `@DeclaredGlobalCockpitEvents`

```
@DeclaredInputs(value = {@DeclaredInput(value = ...), @DeclaredInput(value = ...)})
```

- When using Mockito, typically you `@Mock` your Service and View elements and `@InjectMocks` your Controller



Behavior Testing

Introduction
Unit Testing Widgets
Annotations
Behavior Testing
Action and Editor Testing
Exercise

Behavior Testing your Widget

- Simulate interactions with the widget
 - execute widget methods
 - `assertSocketOutput()`
 - `assertValueSet()`
 - `assertValueNotNull()`
- Example:
 - Simulate a click on the Last Size Button

```
executeViewEvent(DemoWidgetController.LAST_SIZE_BUTTON_ID, Events.ON_CLICK);
```

- Test that the expected output on the *Size* output socket is 0

```
assertSocketOutput(DemoWidgetController.SIZE_SOCKET, Integer.valueOf(0));
```

More Examples

- Send String "dot.separated.string" as an input on widget's input socket

```
executeInputSocketEvent(DemoWidgetController.IN_SOCKET, "dot.separated.string");
```

- Expected socket output as array of 3 strings – "dot", "separated", "string"

```
assertSocketOutput(DemoWidgetController.OUT_SOCKET,  
                  Lists.newArrayList("dot", "separated", "string"));
```

- Expected model variable, lastResultSize, has value of 3

```
assertValueSet(DemoWidgetController.LAST_RESULT_SIZE, Integer.valueOf(3));
```

Behavior Testing your Widget

@Test

```
public void testSocketAndViewEvent()
{
    executeViewEvent(DemoWidgetController.LAST_SIZE_BUTTON_ID, Events.ON_CLICK);
    assertSocketOutput(DemoWidgetController.SIZE_SOCKET, Integer.valueOf(0));

    executeInputSocketEvent(DemoWidgetController.IN_SOCKET, "dot.separated.string");
    assertSocketOutput(DemoWidgetController.OUT_SOCKET,
        Lists.newArrayList("dot", "separated", "string"));
    assertValueSet(DemoWidgetController.LAST_RESULT_SIZE, Integer.valueOf(3));

    executeViewEvent(DemoWidgetController.LAST_SIZE_BUTTON_ID, Events.ON_CLICK);
    assertSocketOutput(DemoWidgetController.SIZE_SOCKET, Integer.valueOf(3));
}
```



Action and Editor Testing

Introduction
Unit Testing Widgets
Annotations
Behavior Testing
Action and Editor Testing
Exercise

Action and Editor Testing

Extend `AbstractActionUnitTest` to test Actions

Extend `AbstractCockpitEditorRendererUnitTest` to test Editors

OOTB both of these provide:

- no-arg constructor checks
- null safety test

Testing with Breadboard Widget

- Displays the widget's info – name, controller, ViewURI
- Allows access to a widget's settings
- Allows testing your widget through its view
 - Can access visual components
- Send data to input sockets
 - Console allows Groovy code
 - Can access Spring context
- View data sent from socket outputs

Testing with Breadboard Widget

Breadboard
Cockpit Development & Testing Tool

Choose a widget:
Demo
[Open as link...](#)

Name: Demo
Code: com.hybris.sample.d
Description: Demo
DefaultTitle: Demo

Controller: com.hybris.sample.D
ViewModel:
ViewURI: /cockpitng/widgets
/demoWidget
/demo.zul
CategoryTag: Demo

Widget Settings

in
java.lang.String
"dot.separated.string"

Try me!

out: [dot, separated, string]

Filter

[dot, separated, string]

name	value
MAX_ARRAY_SIZE	2147483639
elementData	Object[]
serialVersionUID	8683452581122892189
size	3

Console

dot.separated.string

History: Snippets

SEND

Console

```
$output[out]:> [dot, separated, string]
$input[in]:> dot.separated.string
```

Refer back to the *Development Environment* module to refresh your memory on how to use this widget

Exercise 9



Exercise 9 – Test Your Widget

- Complete the basic implementation of a widget controller's test class that extends `AbstractWidgetUnitTest<>`
- Write a unit test that checks for the correct behavior of the controller after receiving a book as input

Thank you.

