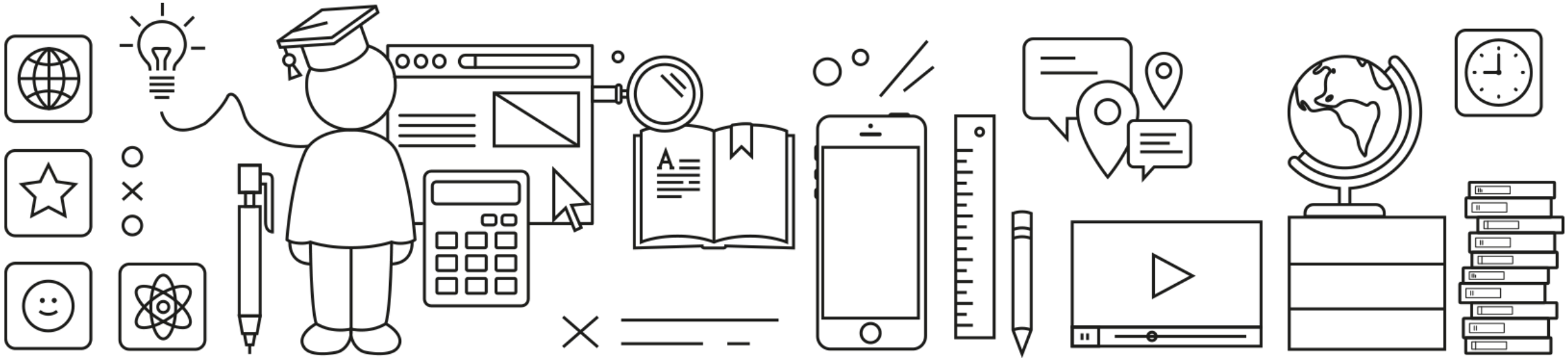
SAP Customer Experience

SAP Commerce Cloud Backoffice Framework Developer Training

Look & Feel and Localization

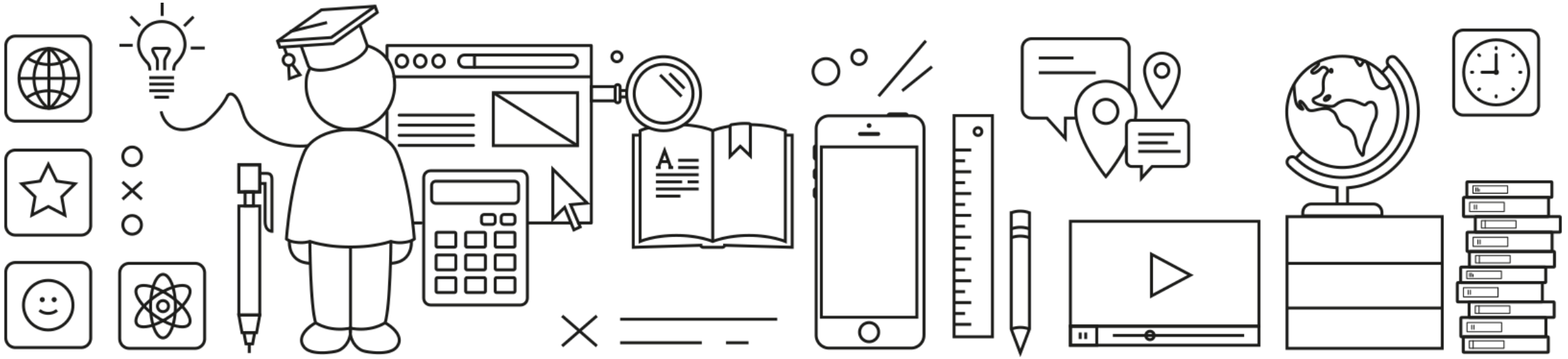
THE BEST RUN SAP



Overview

Overview

- ResourceBundle-based (with language fallback support)
- Application and component based localization
- Pluggable implementation
- Leverages ZK localization for visual components
- Styling basically uses ZK and CSS
- White-labeling allows branding and changing of global standalone application elements
- Sass is fully supported by simply adding one extension
- Support for AngularJS to help those who prefer using it



Localization

Overview
Localization
Styling Components
White-Labeling
Sass
AngularJS

Application and Component Localization

Application String Localization

- Backoffice framework decorates the **ZK Labels** utility
- `com.hybris.cockpitng.util.labels.WebappLabelLocator` implements ZK's `LabelLocator`
- `LabelService` interface's `getObjectLabel(...)` method retrieves the appropriate label
- Localization files are stored by convention:

Item	Location
Widgets	`myextension/backoffice/resources/widgets/mywidget/labels`
Editors	`myextension/backoffice/resources/widgets/editors/myeditor/labels`
Actions	`myextension/backoffice/resources/widgets/actions/myaction/labels`

Widget String Localization via Standard Java Resource Bundles

- Location and naming of file group (a.k.a. *resource bundle*)
  - `myextension/backoffice/resources/widgets/myWidget/labels`
  - Default labels and catch-all (also, the ***base name***): `labels.properties`
  - Language-specific overrides: `labels_`***`xx`***`.properties`
    (where ***xx*** is the language ISO code)
    - ❑ Ex: `labels_es.properties` for Español (i.e., Spanish)
    - ❑ Ex: `labels_de.properties` for Deutsch (i.e., German)
  - `WidgetInstanceManager` interface `getLabel(String key)`
    - ❑ Delegates to `LabelService`

Localization Usage

- From within ZUL file:

```
<button id="sendBtn" label="${labels.button}" />
```

- From within Widget controller:
  - Via `WidgetInstanceManager`

```
String widgetName = getWidgetInstanceManager().getLabel(WIDGET_NAME_LABEL);
```

    - ...or via widget's convenience method):

```
String widgetName = getLabel(WIDGET_NAME_LABEL);
```

  - Editor and Action controllers can use the `getLabel()` method of the `EditorContext` and `ActionContext` classes, respectively

Styling Components

Overview
Localization
Styling Components
White-Labeling
Sass
AngularJS

Styling Components

- Use CSS
- Based on ZK
- ZK Java API or inside `.zul` view files: `sclass` property
- Actions: change icons inside definition file

```
<action-definition ...>
    ...
    <iconUri>images/icon.png</iconUri>
    <iconHoverUri>images/icon_hover.png</iconHoverUri>
    <iconDisabledUri>images/icon_disabled.png</iconDisabledUri>
</action-definition>
```

Styling Components

ZK Java API

BookDetailsController.java

```
final Div div = new Div();
div.setHeight("100%");
div.setSclass(CSS_CLASS_IMAGE_DIV_LEFT);
```

Inside `.zul` view files

bookDetails.zul

```
<style src="${wr}/styles.css"/>
<div id="bookDetailContainer" sclass="yw-bookdetail-container"/>
```

White-Labeling

White-Labeling Your Application

- White-label products, services, or applications:
  - Intentionally *brand-less*
  - designed by a *supplier* to be branded by a *reseller* to sell as if their own

- Change general standalone application appearance e.g. login page design:

`custombackoffice/project.properties`

```
backoffice.cockpitng.mainpage.css=/cng/css/mainpage_whitelabel.css
backoffice.cockpitng.loginpage.css=/cng/css/loginpage_whitelabel.css
backoffice.cockpitng.overridewidgetsandeditors.css=/cng/css/mywidgetseditors.css
```

Sass

Overview
Localization
Styling Components
White-Labeling
Sass
AngularJS

Sass

Syntactically Awesome Style Sheets

A scripting language, a CSS preprocessor, that gives better experience in styling your application.

Sass – Main Features*

- Fully CSS-compatible
- Language extensions such as variables, nesting, and mixins
- Many useful functions for manipulating colors and other values
- Advanced features like control directives for libraries
- Well-formatted, customizable output

* Sass © 2006–2015. "Sass Documentation". http://sass-lang.com/documentation/file.SASS_REFERENCE.html. (accessed Jan. 17, 2017).

Enabling Sass Support

- Add the `npmancillary` extension to your SAP Commerce Cloud platform
  - Add it to `localextensions.xml`
  - Contains NodeJS and all necessary Grunt libraries for Sass support
- Set `backoffice.sass.enabled` to `true` in the Commerce platform's `local.properties`

`local.properties`

```
backoffice.sass.enabled=true
```

- Restart the Commerce server
  - The resources are cached by the server by default
  - You can disable caching by adding the following line to your `local.properties`

`local.properties`

```
backoffice.cockpitng.resourceloader.resourcecache.enabled=false
```

Registering a Sass Extension

1. When creating an extension using `ant extgen`, and `ybackoffice` as the template...
   - You're asked "Register as a SASS extension?"
   - You should type in `true`
2. Or, if you haven't registered it as a Sass extension,
   - Add the following block to the extension's `buildcallback.xml`

custombackoffice/buildcallback.xml

```
<macrodef name="custombackoffice_before_build">
  <sequential>
    <register_sass_extension extensionname="custombackoffice"/>
    <register_sass_extension extensionname="custombackoffice"
                             resources="resources"  destination=""/>
  </sequential>
</macrodef>
```

More is on HELP

For more details about Sass support in the Backoffice Framework, please read the documentation on SAP Commerce Help:

Backoffice - Sass Integration

http://help.hybris.com search term: 'Sass Integration'

https://help.hybris.com/1811/hcd/830f7ed55f804b1980dcfad2f83ce3a7.html

AngularJS

Overview
Localization
Styling Components
White-Labeling
Sass
AngularJS

AngularJS

“AngularJS is a structural framework for dynamic web apps.” *

Not our recommended tool for creating widgets!

But we support it anyway ☺

For more information and tutorials, visit

AngularJS in Backoffice Framework

* Google © 2010-2017. "Developer Guide". https://docs.angularjs.org/guide/introduction. (accessed Jan. 20, 2017).

Thank you.