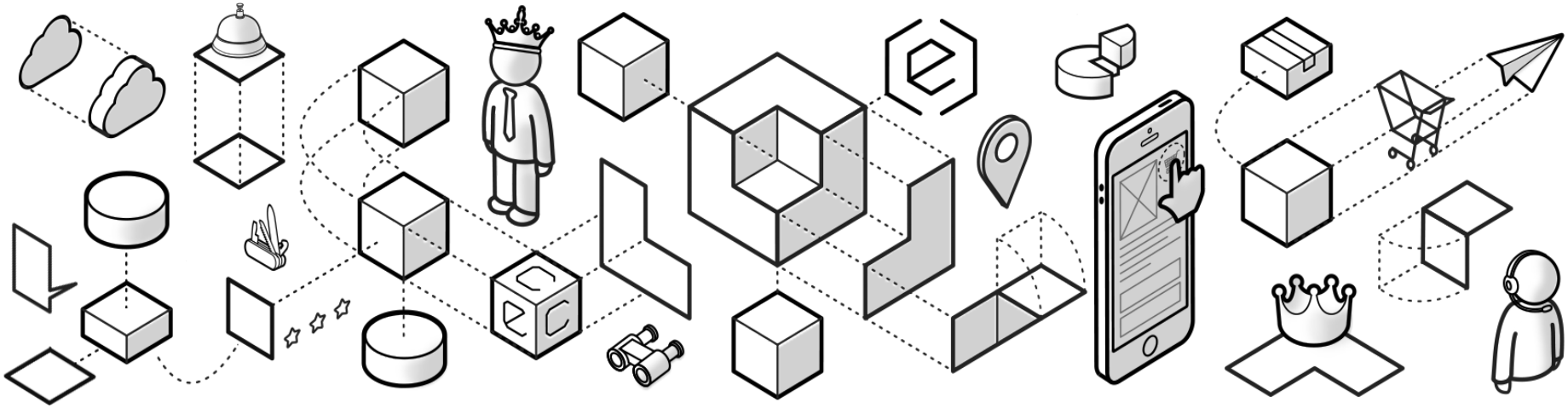
hybris software

An SAP Company

Installing the hybris Accelerator

hybris **Commerce Developer** Training – Part I



Build Framework

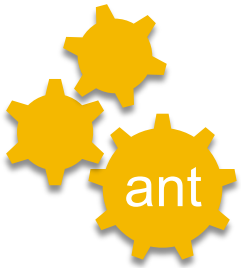
Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

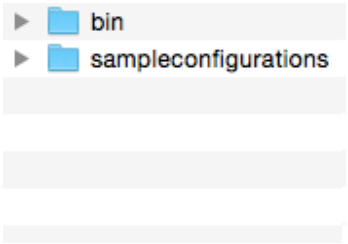
Build Framework

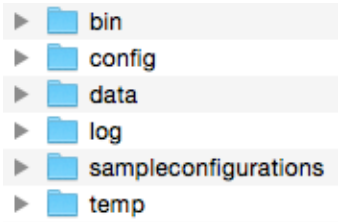
- The hybris Platform has a build framework based on Apache Ant
  - Ant handles compilation and a number of automation tasks
  - Class executables compiled by the Eclipse IDE are not used by hybris
- There is a build file in every extension, but we generally just use the one in the platform extension, which builds the entire suite
- It builds every extension listed or referred to by localextensions.xml

Building with ant

- When you call `ant`, the build framework:
  - generates and compiles Model classes (defined in the next module)
    - according to the definitions in the `*-items.xml` files
    - In order of extensions dependency (hierarchy)
  - collects localization property files (`locales_XY.properties`).
  - collects layout configuration files (`hmc.xml`).
  - updates configuration of the hybris Server
  - generates four new folders (only in the first run!)

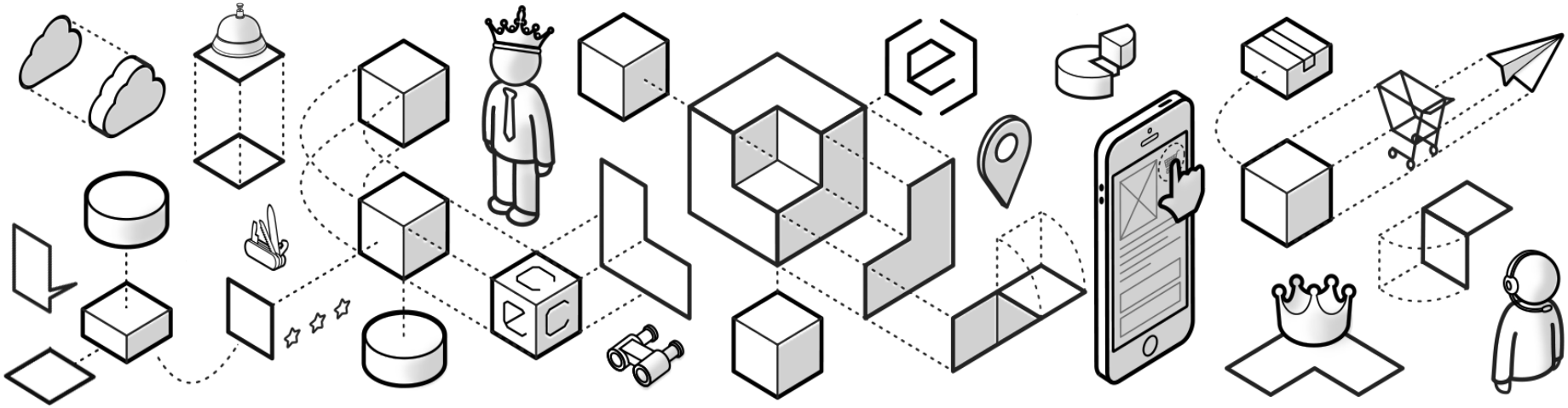






Build Targets

Ant Target	Description
all	Builds and configures the server
clean	Cleans up platform and extensions, deletes .class files
modulegen	Generates multiple extensions using Accelerator templates
extgen	Runs extension-generation tool
initialize	Initializes the system
-p	Shows list of all Ant targets (many more than listed here)



Extension Concept

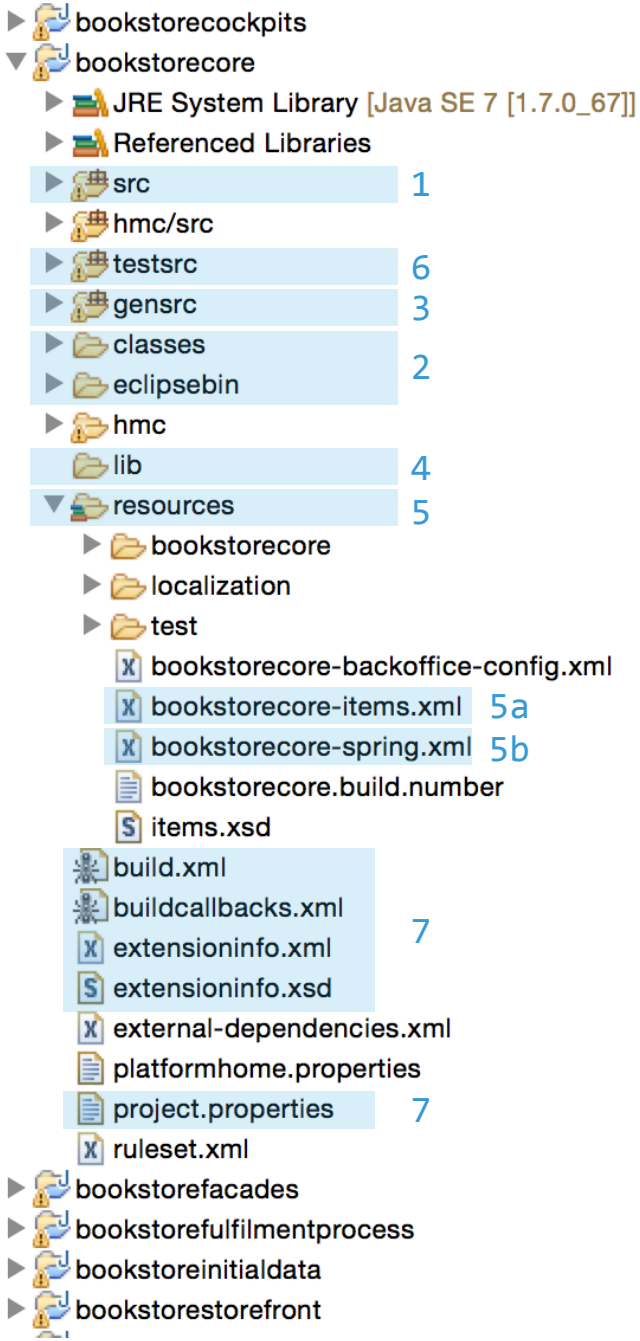
Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

Where to Begin with hybris

- Starting any new project involves creating one or more extensions.
  - Each extension will contribute a part of the greater whole, including modifying or adding to the data model, business logic, backoffice tools configuration, etc.
  - Your (project) code is separated from hybris (framework) code, making your code easier to reuse and to migrate to future versions.
  - hybris bundles a proprietary code generator
- In general, a project will include
  - A front end extension, with all jsp pages, tag and css files, controllers, etc.
  - A testing extension, containing the test cases and data
  - One or more infrastructure or utility extension (such as services)

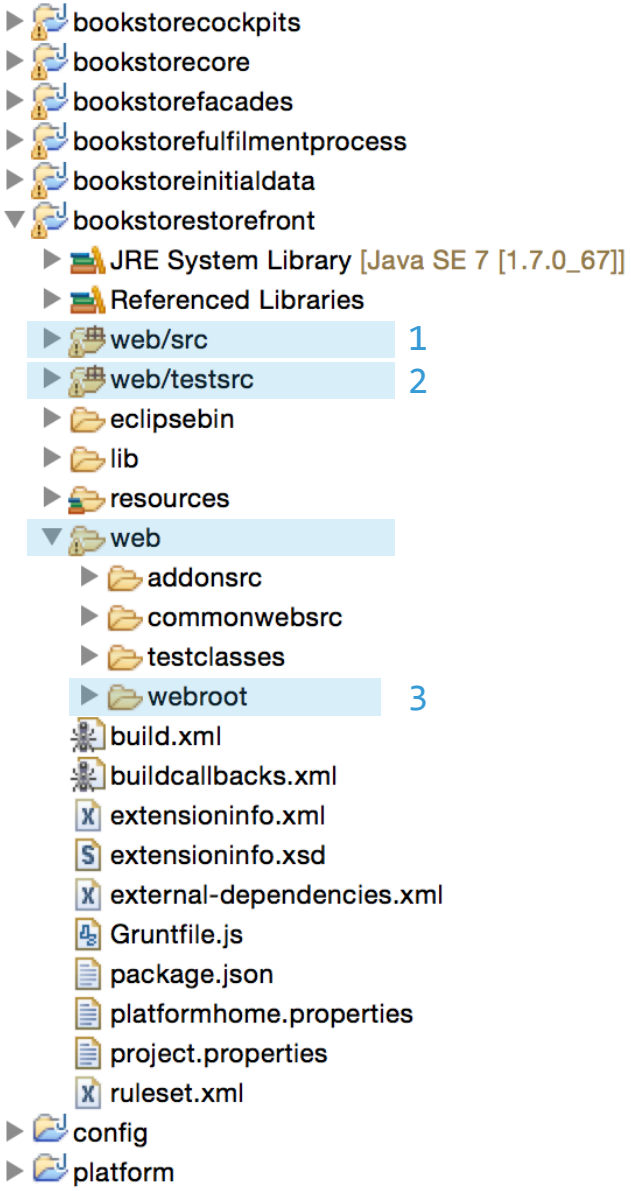
Structure of the Core Extension

1. Business Logic
2. Compiled sources
3. Generated Java files
4. External library files
5. Resources folder for external data, type definitions & localization
   - 5a. Model definition
   - 5b. Spring configuration
6. JUnit test classes
7. Files for Eclipse, Apache Ant, and extension-specific configuration



Structure of the Storefront Extension

1. Source code for web app
2. Test code for web app
3. Web module configuration and pages



Creating a New, non-Accelerator Extension

- Non-Accelerator extensions use a template, such as

yempty	Empty extension with minimal configuration
yaddon	Basic template for writing AddOns
ybackoffice	Structure for a Custom Backoffice Extension
ycockpit	Create a new cockpit extension
ycommercewebservices	Exposes parts of the commerce Façade as a REST-based web services

- To create single extension, invoke `ant extgen`
- Reference any required extensions in extensioninfo.xml
  - See the next section for details
- Add your extension to config/localextensions.xml
- Invoke `ant clean all`

Creating an Accelerator-based Extension

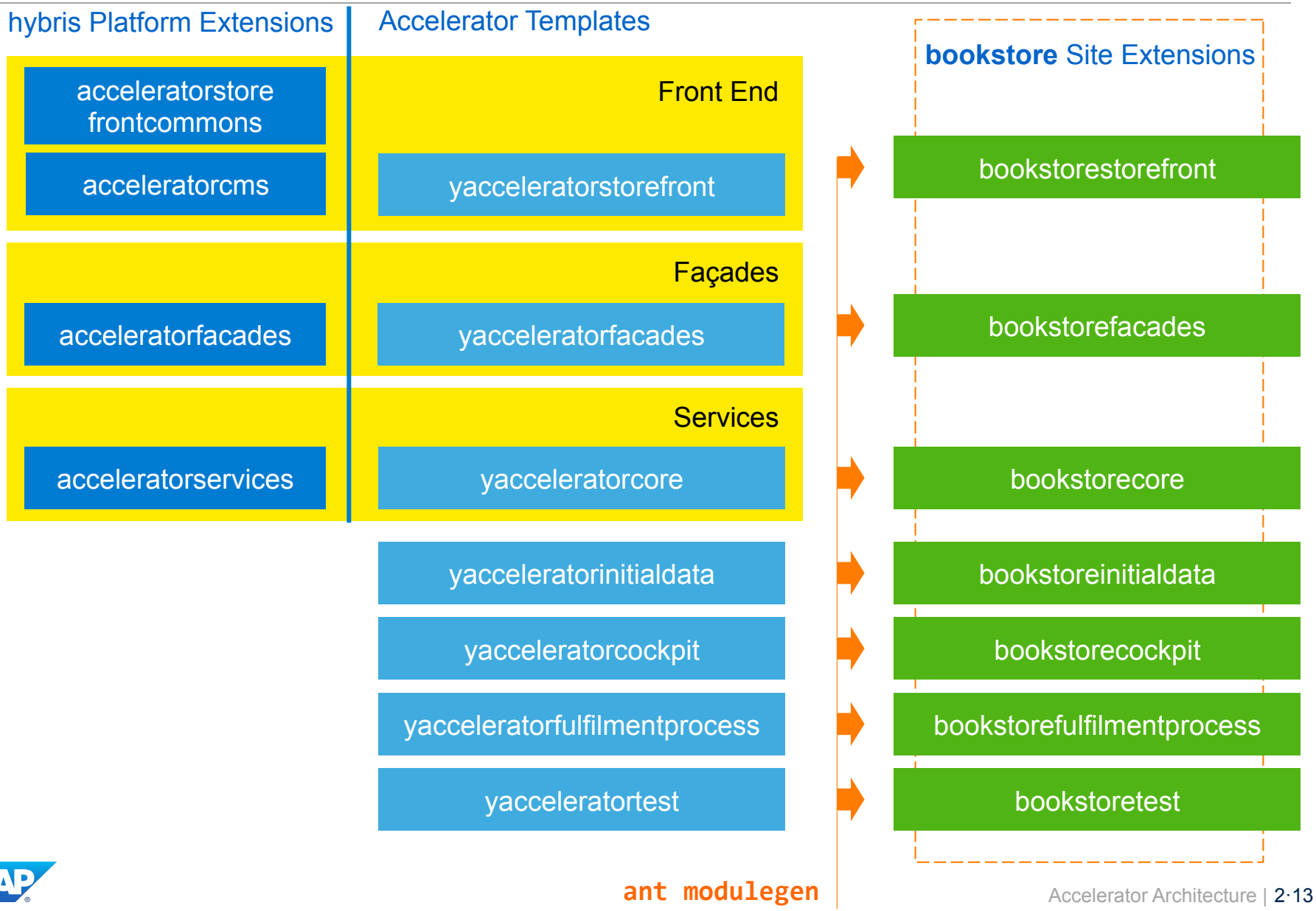
- An Accelerator-based project's extension set is defined by the Accelerator template you are using as the basis for your site
  - Your project's extensions are created by `ant modulegen`
  - For example, if you base your bookstore shop on the B2C Accelerator, ant will generate the following extensions for you:
    - bookstorecockpit
    - bookstorecore
    - bookstorefacades
    - bookstorefulfilmentprocess
    - bookstoreinitialdata
    - bookstorestorefront
    - bookstoretest

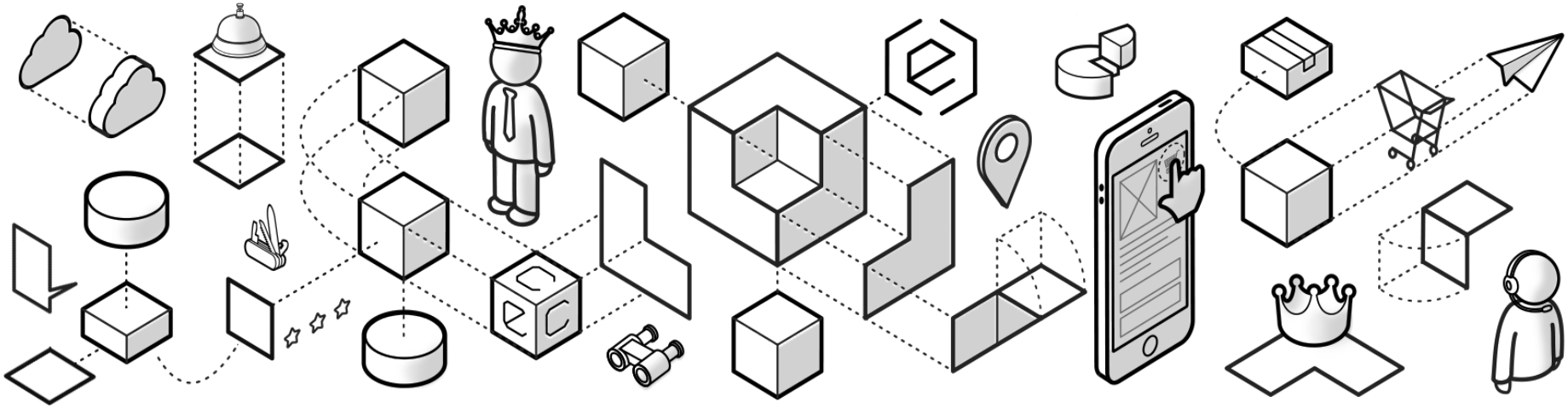
Most of your business logic will go in these extensions

Essential and project data loaded here

Your storefront web app

Template and Product Extensions





Basic Configuration

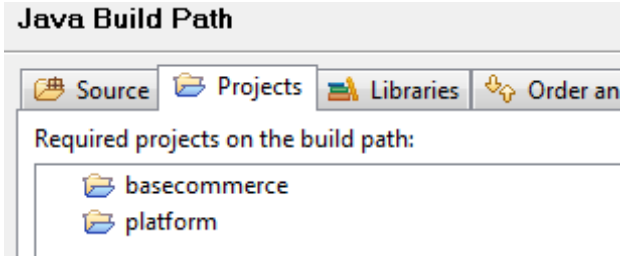
Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

Configuration Files

- The list of extensions included in the ant build is defined in
  - `config/localextensions.xml`
- Each extension configures its dependencies in
  - *`extensionName`*`/extensioninfo.xml`

```
<!-- add all required extensions -->
<requires-extension name="basecommerce"/>
```

- Configure dependencies in Eclipse as well



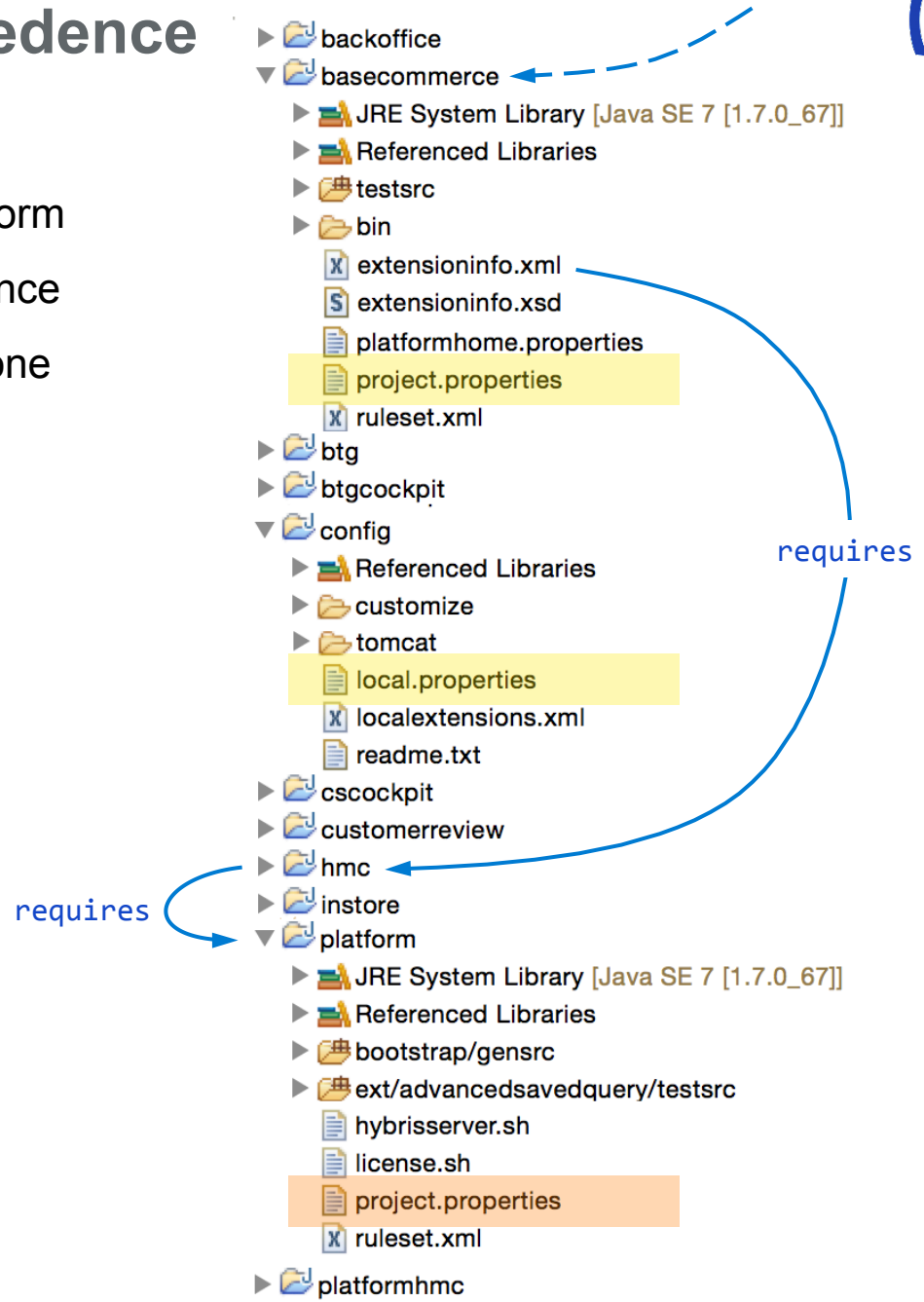
- Put extension-specific configuration in
  - *`extensionName`*`/project.properties`
- Override extension configuration in
  - `config/local.properties`
- Server configuration such as database URL and credentials
- To use a different configuration, i.e. for a test server,
  - Run `ant -Duseconfig=testserver` will use `localtestserver.properties` as override

project.properties Precedence

- All extensions implicitly depend on platform
- Dependency chain determines precedence
- local.properties settings override everyone
- cyclic dependencies rejected by ant

```
config/local.properties
...
b2ccommerce/project.properties
basecommerce/project.properties
hmc/project.properties
platform/project.properties
```

Each layer overrides those below to determine system configuration



Configuration changes

- Modify the `local.properties` file and restart the application server
  - No need to call `ant` in the console
  - Configuration files are read during startup.
  - All configuration files are inside the `config` directory
- However if you change any Tomcat configuration settings (such as `tomcat.http.port`), you need to call `ant server`
- If in doubt, call `ant all`
- It's possible to change the properties on the runtime in the hAC
  - However, after a server restart, these changes are lost

localextensions.xml

- List of extensions used by build framework
- Extensions can be listed by name instead of location path
- Dependencies are resolved automatically using the `path` parameter. Build will find dependent extensions which are not explicitly listed
- A complete localextensions.xml can be generated using the extensionsxml ant target
- Sample localextensions.xml files are provided in the hybris/sampleconfigurations directory; for example,
- `hybris/sampleconfigurations/b2c_acc+cis+oms_extensions.xml`
- During install process, overwrite hybris/config/localextensions.xml with content of desired sample file, then run ant all.

Load Required Extensions

- Define extensions explicitly

```
<extensions>
   <extension dir="${HYBRIS_BIN_DIR}/ext-hybris/cmscockpit"/>
   ...
```

- Or by using a lookup folder, defined with the `<path>` tag
  - Allows extensions to be loaded by name rather than path
  - Allows lazy loading — hybris searches path directories for any extension referenced by another extension, and pulls it into the current configuration

```
<extensions>
   <path dir="${HYBRIS_BIN_DIR}"/>
   <extension name="commerceservices"/>
   ...
```

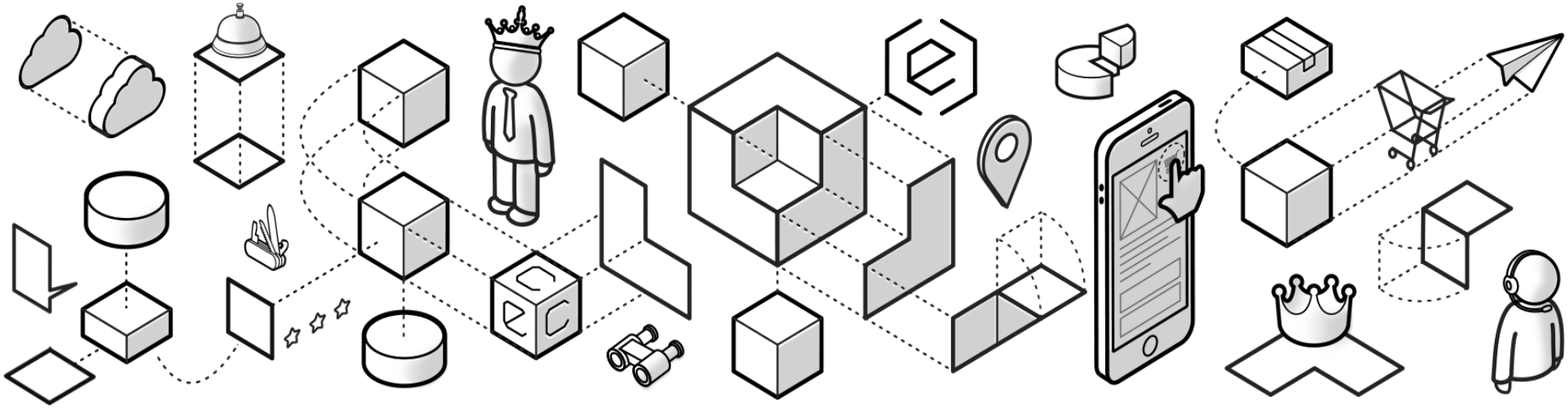
requires basecommerce
cms2
customerreview
payment
...

Loading All Extensions in a Folder

- You may autoload entire extension directories with the `path` tag, and limit lookup to specific directories.
  - OOTB, the localextensions.xml file specifies the path `hybris/bin`

```
<extensions>
   <path autoload="true" dir="${HYBRIS_BIN_DIR}/ext-cockpit" depth="3"/>
   <path dir="${HYBRIS_BIN_DIR}/ext-commerce"/>
   <extension name="basecommerce"/>
   <extension dir="${HYBRIS_BIN_DIR}/ext-content/cms2"/>
</extensions>
```

```
[hybrisserver] ------------------------------------------------------------
[hybrisserver] --- Extensions in dependency order ( options:
[hybrisserver] ---  @deprecated: is deprecated, p: platform extension,*: auto-required
[hybrisserver] ---  ?: lazy-loaded, i: got items.xml, b: got beans.xml, c: got core module
[hybrisserver] ---  w: got web module, h: got HMC module )
[hybrisserver] ------------------------------------------------------------
[hybrisserver] core 5.4.0.1 [p*cib]
[hybrisserver] testweb 5.4.0.1 [p*w]
[hybrisserver] scripting 5.4.0.1 [p*ci]
[hybrisserver] paymentstandard 5.4.0.1 [p*ci]
[hybrisserver] mediaweb 5.4.0.1 [p*cw]
[hybrisserver] maintenanceweb 5.4.0.1 [p*w]
[hybrisserver] deliveryzone 5.4.0.1 [p*ci]
[hybrisserver] commons 5.4.0.1 [p*ci]
[hybrisserver] processing->(scripting,commons) 5.4.0.1 [p*ci]
[hybrisserver] impex->processing 5.4.0.1 [p*ci]
[hybrisserver] validation->impex 5.4.0.1 [p*ci]
[hybrisserver] catalog->(validation,commons) 5.4.0.1 [p*ci]
[hybrisserver] europe1->(catalog,impex) 5.4.0.1 [p*ci]
```

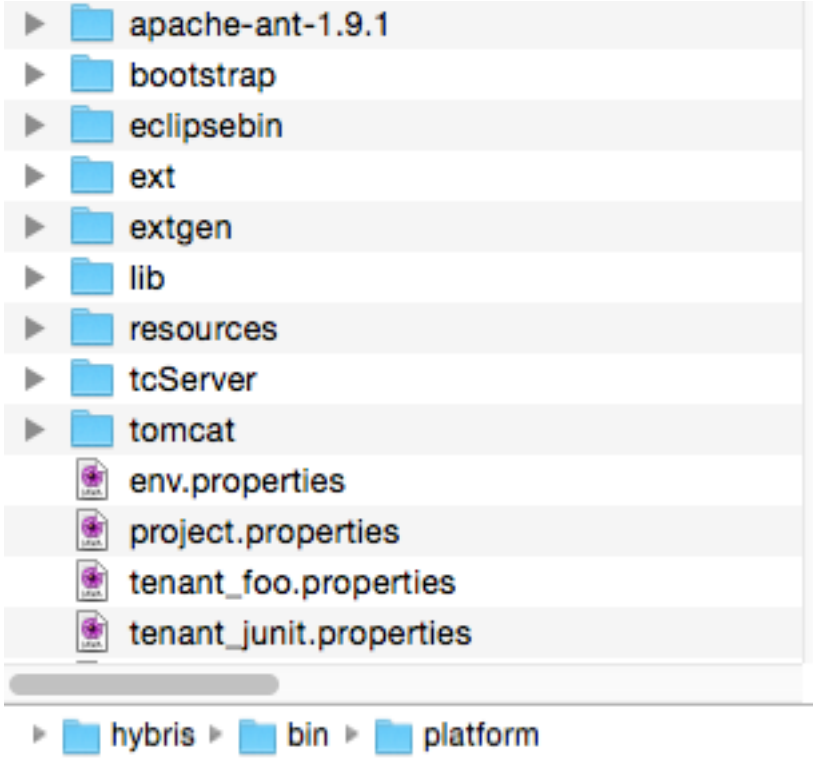


hybris Server

Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

Bundled With hybris

- hybris comes with a bundled hybris server and a configuration template for the tcServer

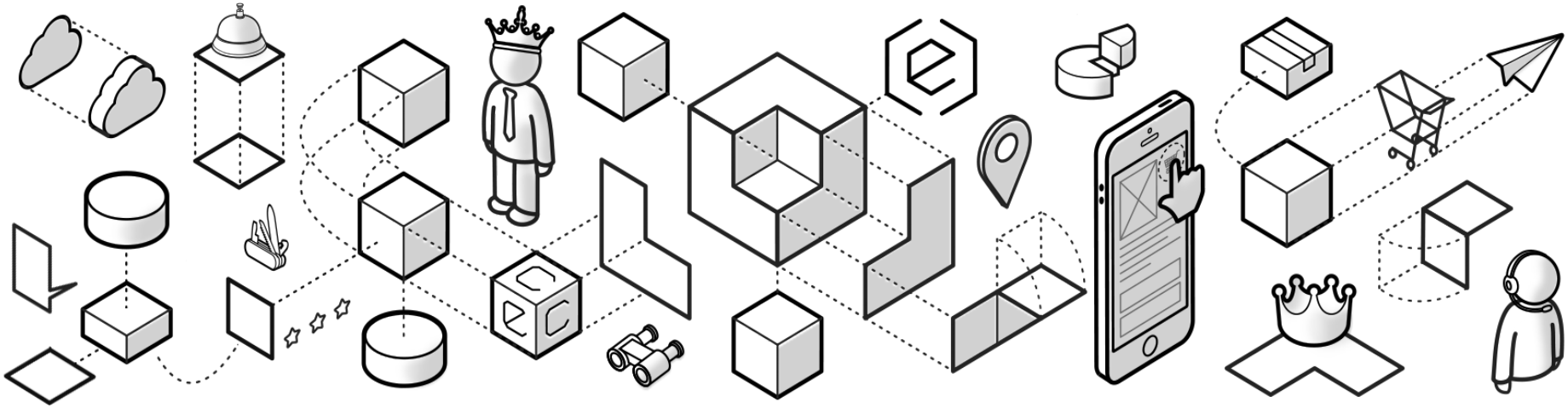


What is the hybris Server?

- Optimized and pre-configured server based on Apache Tomcat
- Production-ready quality and best suited to run all applications of the hybris Commerce Suite

Apache Tomcat

- Configuration templates available for development and production deployment
- Independent of the operating system
- Easy installation
- Can be run as a system service under Microsoft Windows or Linux
- Contains a wrapper that automatically restarts the Apache Tomcat Java Virtual Machine if the Apache Tomcat hangs or stops.



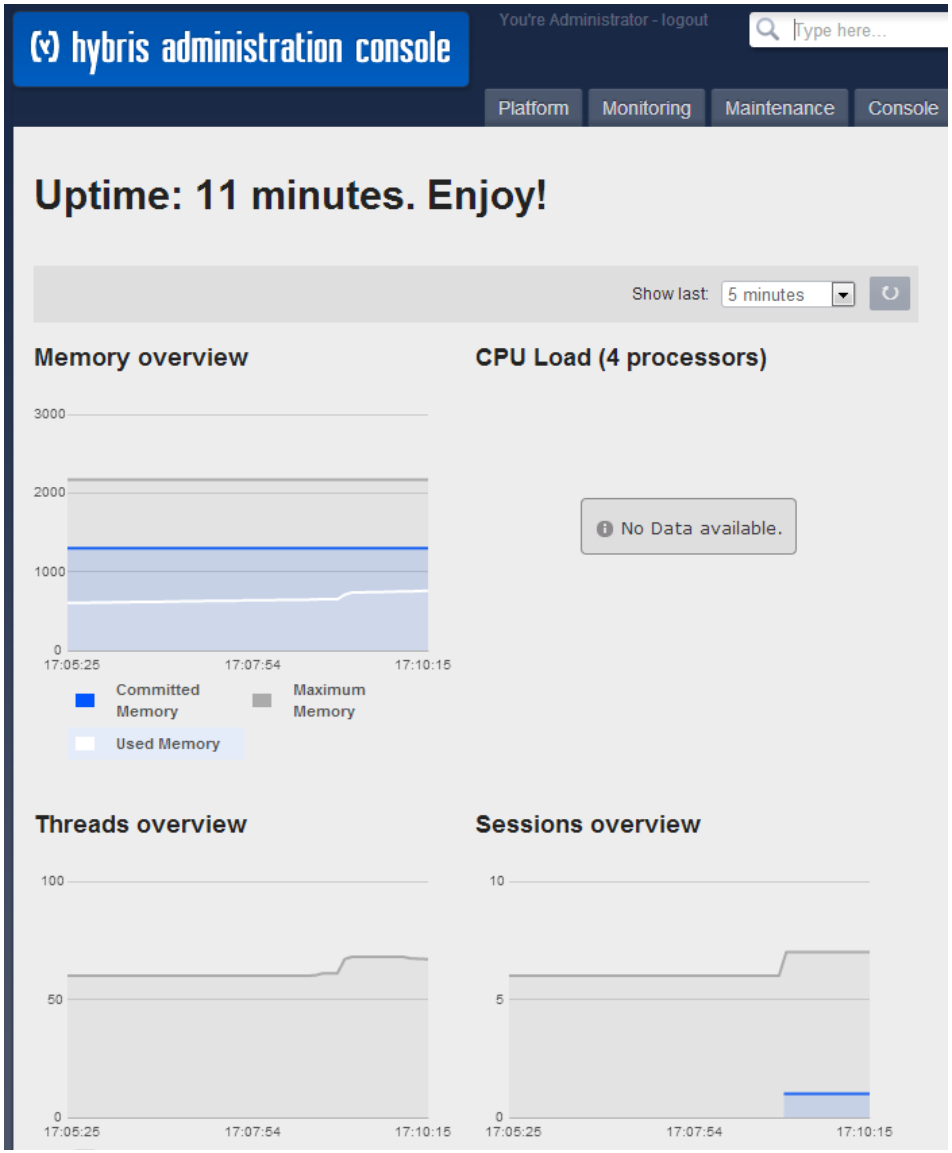
hAC, Initialization, and Update

Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

hac - hybris Administration Console

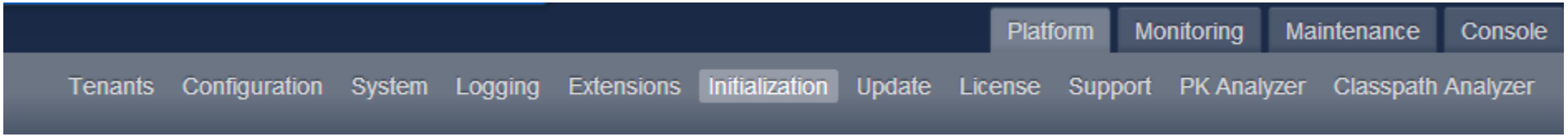
- Administration
- Monitoring
- Configuration
- `localhost:9001/` *(by default)*
- `localhost:9001/hac`

(In this training)



Demo

Initialize or Update the System



System Initialization:

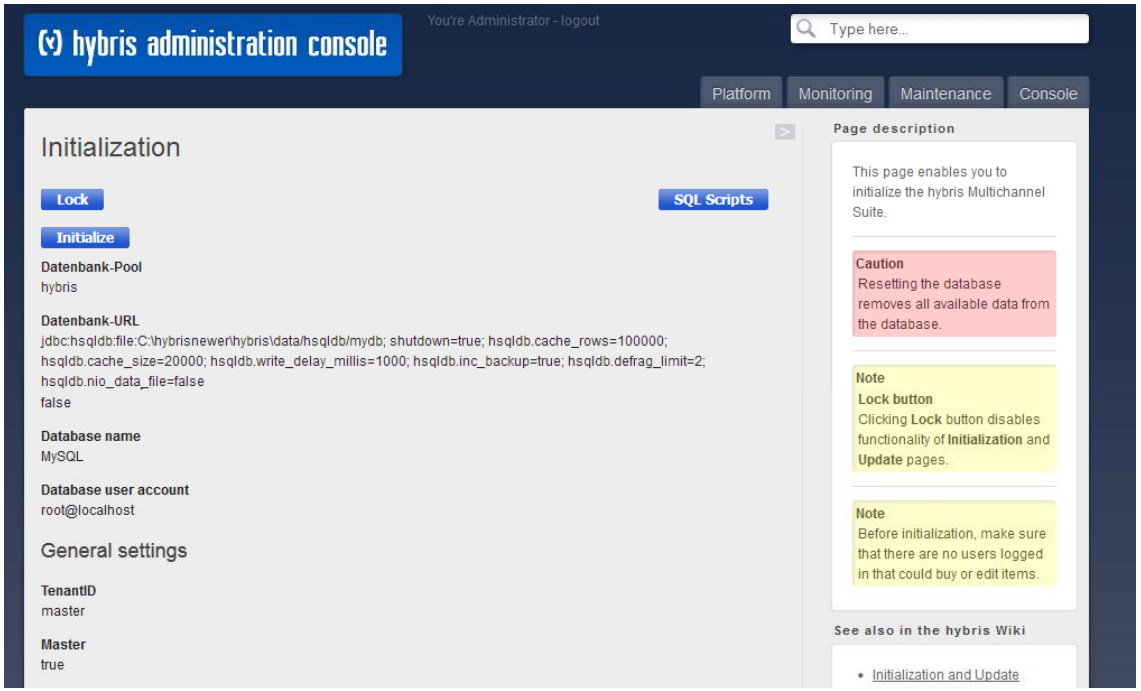
- Entire type system is created from scratch.
- ALL database tables are dropped.
- Data model is created from scratch as defined in the `items.xml` files.
- New tables with initial dataset are created
- Existing data model definitions will be lost!

System Update:

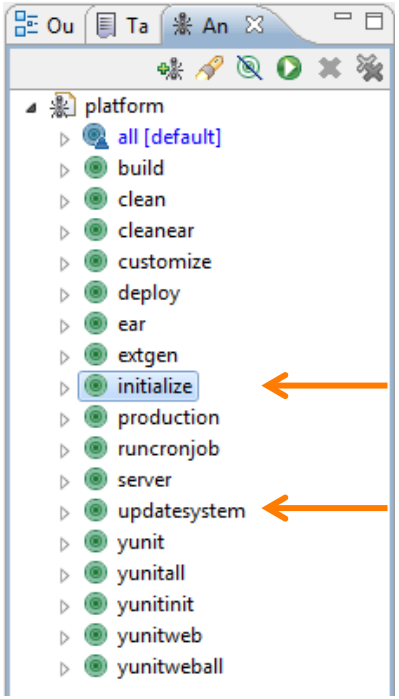
- Existing tables are updated to match changes in the domain model.
- No loss of data!
- Two major aspects:
  - Adding newly defined types to the type system definition in the database
  - Modifying type system definition in the database to match the definition in the domain model

Initialization and Update

1. hac -> Platform -> Initialize | Update



2. Ant view in eclipse + tenant



3. Command line

```
/hybris/bin/platform $ ant initialize -Dtenant=master
```

```
/hybris/bin/platform $ ant updatesystem -Dtenant=master
```

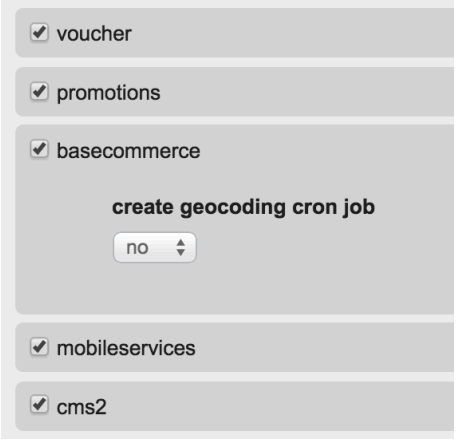
Essential Data vs. Project Data

Essential Data

- Necessary during initialization:
  Creates the Default catalog, restrictions, and basic CronJobs, for example.

Project Data:

- Extension-specific project data

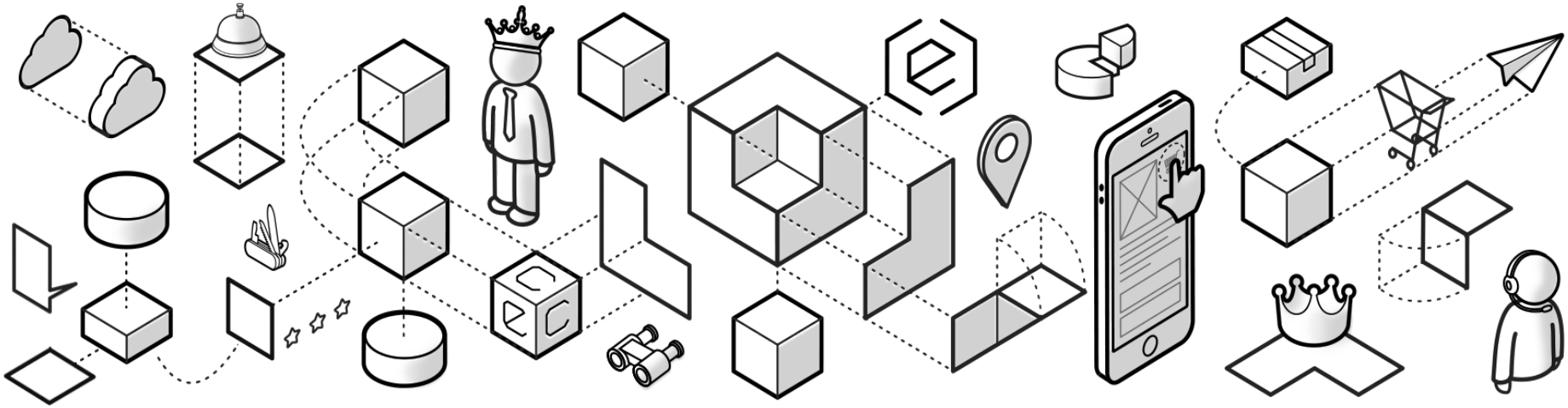


How to include:

- Convention over Configuration `essentialdata*.impex`, `projectdata*.impex`
- Hook Service Layer code into hybris initialization and update life-cycle events @SystemSetup annotation

More Information

http://wiki.hybris.com/display/release5/Initializing+and+Updating+the+hybris+Commerce+Suite



Spring in hybris

Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

What is Spring?

The Spring Framework is a lightweight open source application framework for the Java platform provided and maintained by SpringSource.

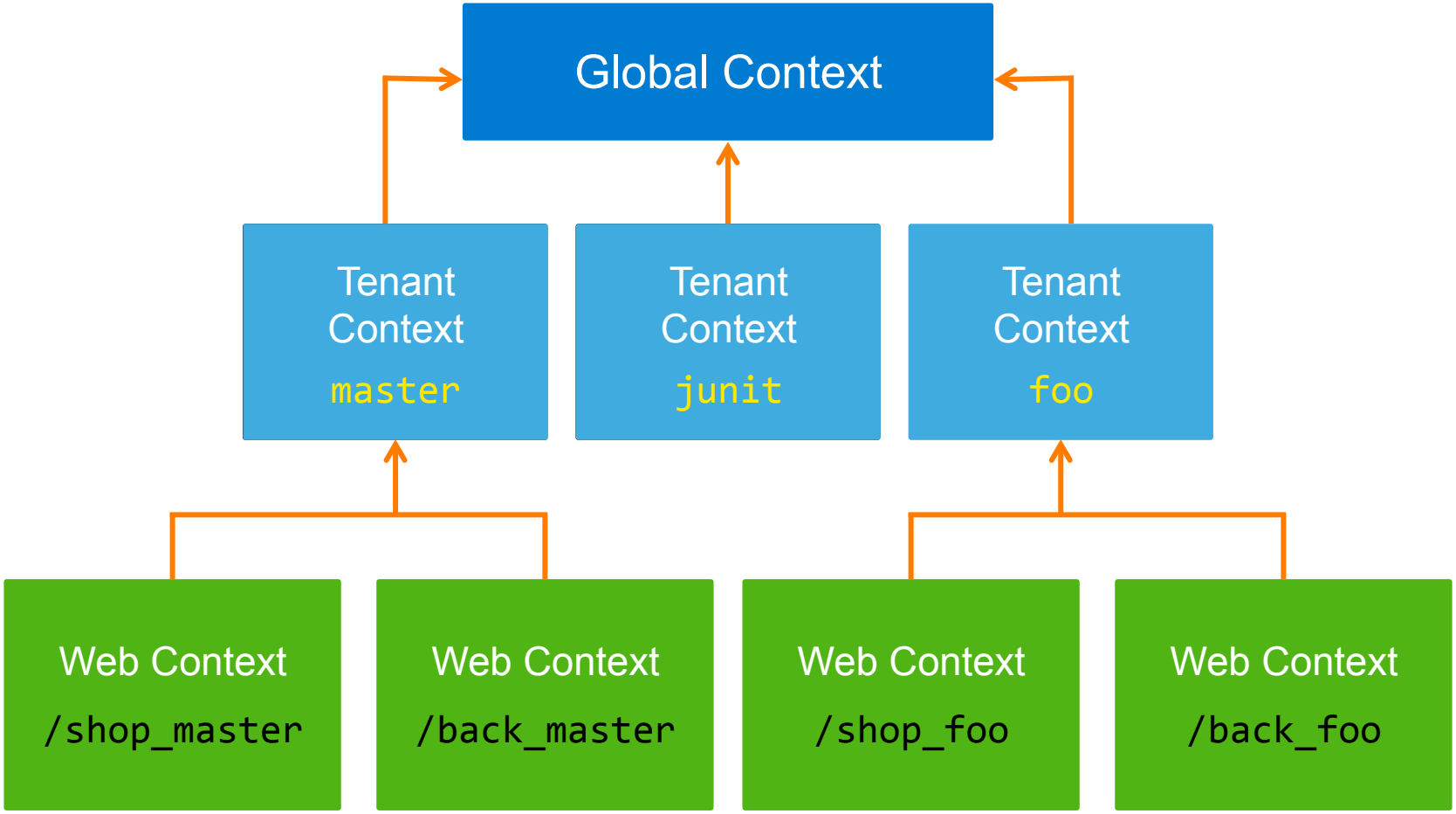
- Provides many components, not all of them used in hybris
- The most important ones that are used by hybris:
  - Dependency Injection (also known as “Inversion of control”), used heavily, provides better decoupling and improves testability
  - Aspect-Oriented Programming, not used by default, but usable for extending stuff which isn’t customizable by default or implementing cross-cutting-concerns
  - Spring MVC, request based framework used in the accelerators
  - Spring Security, used for authentication and basic authorization

Spring Configuration Review

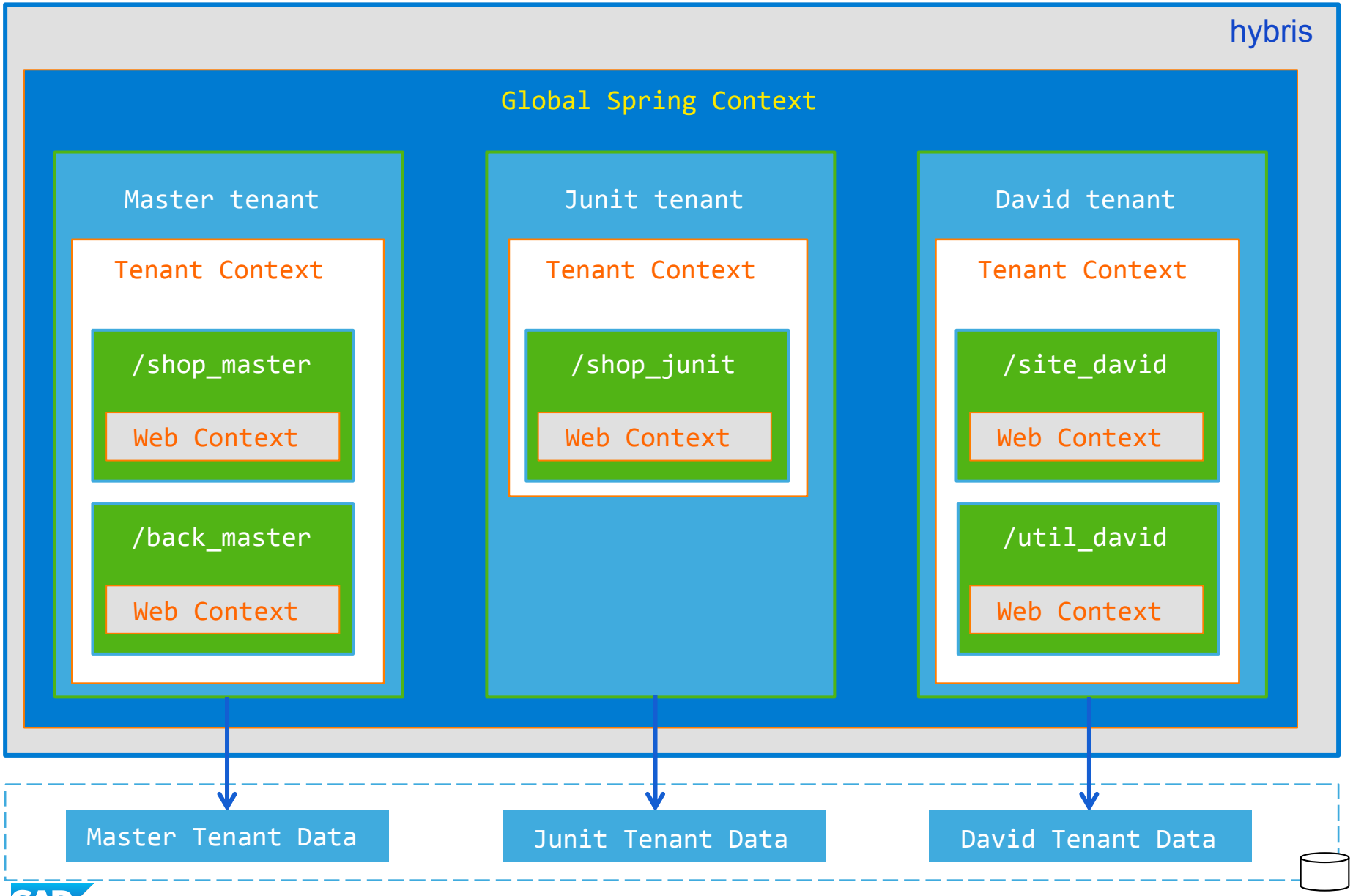
- Configuring a bean in Spring
  - specify parent bean to inherit its configuration
  - property value can be literal or reference to another bean
  - lists and maps may be merged with definition in other extensions

```
<bean id="bean1" class="Bean1Class" parent=”bean1Parent">
   <constructor-arg value="arg1" />
   <property name="property1" value="literal" />
   <property name="property2" ref="otherBean1" />
   <property name="property3">
      <list merge="true"> <ref bean="otherBean2" /> </list>
   </property>
   <property name="property4">
      <map> <entry key="key1" value-ref="otherBean3" /> </map>
   </property>
</bean>
```

Spring contexts



Context visibility



Spring configuration of extension

- There are 3 xml files for your bean definitions

`global-{ext-name}.xml`

- Beans are shared among all extensions

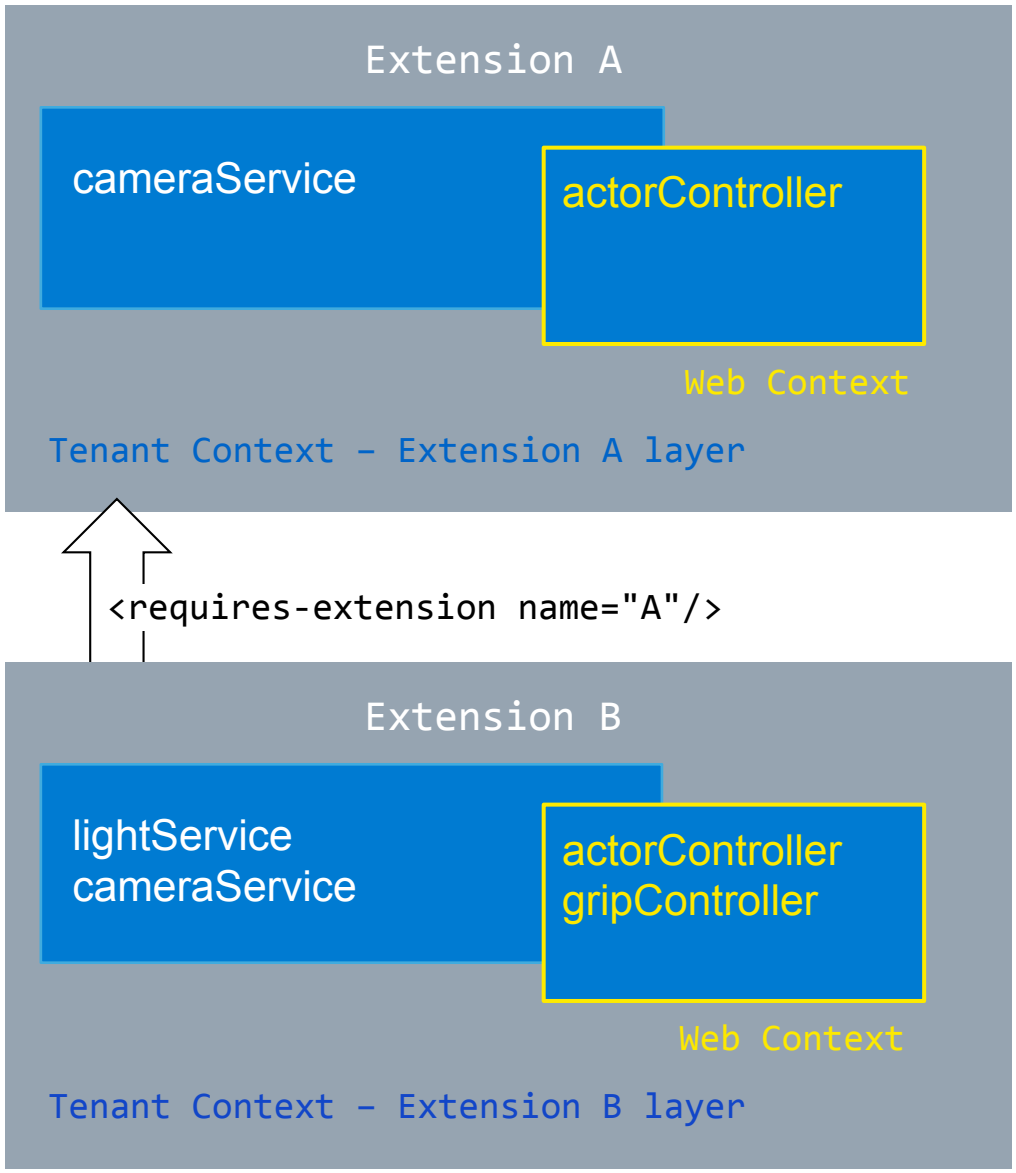
`{ext-name}-spring.xml`

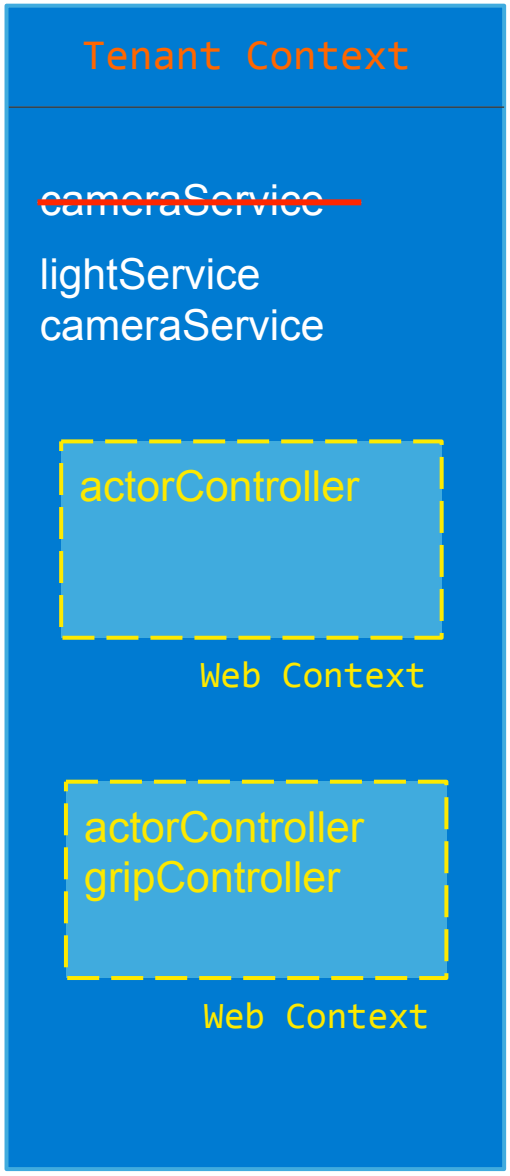
- Beans are shared among all extensions
- Beans will have as many instances as there are tenants.

`web-application-config.xml`

- Beans are available only for selected extension and per tenant

Spring Configuration In Extensions





Using Alias to avoid Overwriting Services

Extension A

```
<alias alias="cameraService"
       bean="defaultCameraService"/>

<bean name="defaultCameraService"
      class="35mmFilmCamera"/>
```

Tenant Context – Extension A layer

```
<requires-extension name="A"/>
```

Extension B

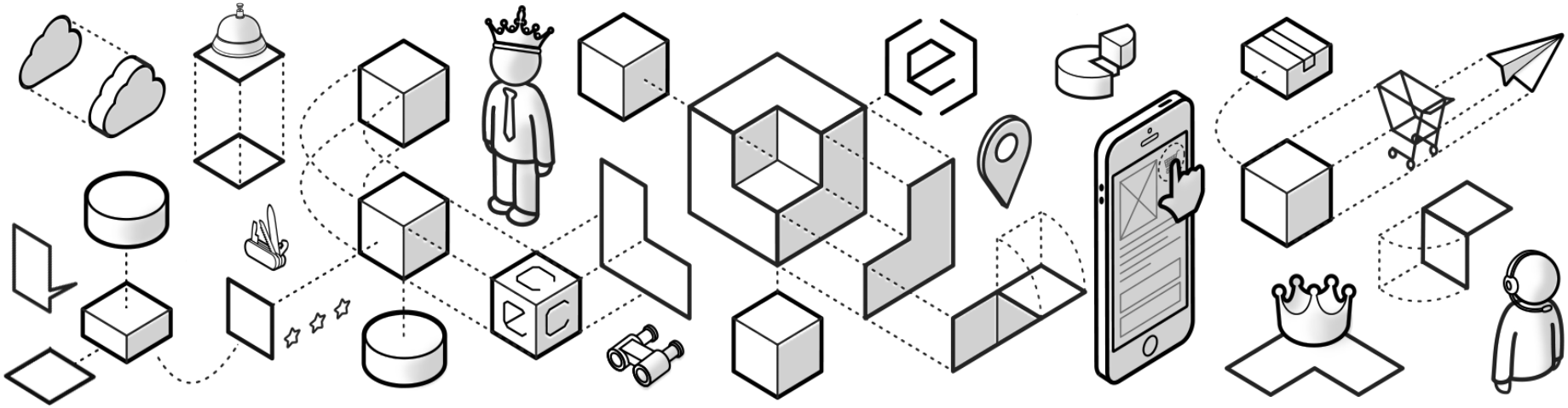
```
<alias alias="cameraService"
       bean="newCameraService"/>

<bean name="newCameraService"
      class="IMAXCamera"/>
```

Tenant Context – Extension B layer

Tenant Context

cameraService → defaultCameraService

cameraService → newCameraService



Exercise How-to

Build Framework
Extension Concept
Basic Configuration
hybris Server
hAC, Initialization, and Update
Spring in hybris
Exercise How-to

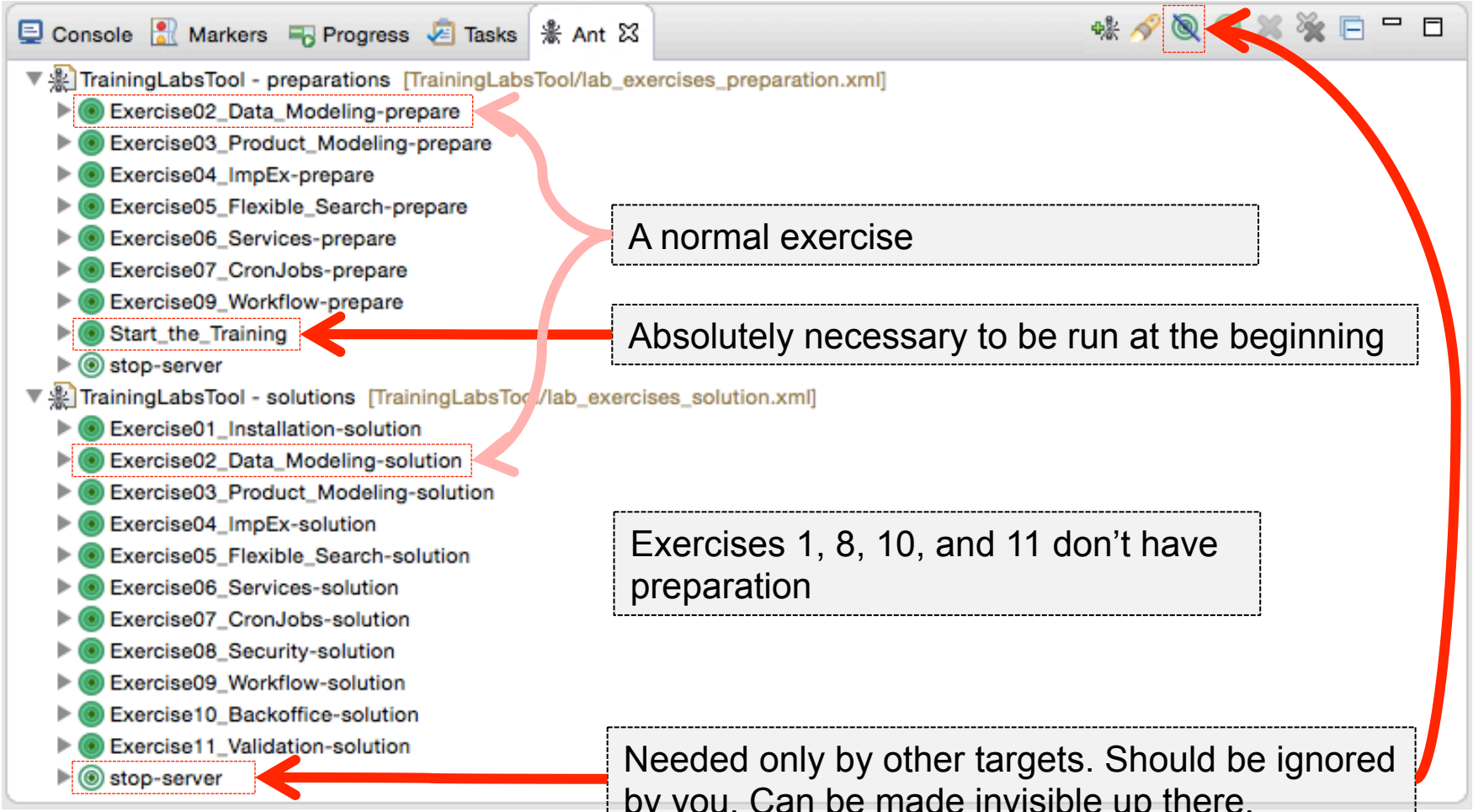
General Information

- On the USB stick: zip + handouts + Instructions
- Zip file : STS + hybris platform + Training Labs Tool
- Want to use your own IDE? follow the alternative scenario
- Configure your IDE and your system to use latest Java 1.7

Training Labs Tool

- **Helps the automation of doing exercises**
- **At the beginning run** *Start_the_Training*
- **Before each exercise**
  - Stop the server
  - Run *ExerciseX-prepare* ant target
- **For each exercise there is an**
  - *ExerciseX-prepare,* and an
  - *ExerciseX-solution*

Training Labs Tool – Ant View



Training Labs Tool

DON'T change the content of any file in TrainingLabsTool!

- TrainingLabsTool
  - JRE System Library [JavaSE-1.7]
  - libs
  - resources
    - exercise1
    - exercise10
    - exercise11
    - exercise2
      - preparation
        - bookstorecore-items.xml
        - bookstorecore-locales_en.properties
        - bookstorecore-spring.xml
        - PointToNextLevelAttrHandler.java
      - solution
        - bookstorecore-items.xml
        - bookstorecore-locales_en.properties
    - exercise3
    - exercise4
    - exercise5
    - exercise6
    - exercise7
    - exercise8
    - exercise9
  - exercises.properties
  - lab_exercises_preparation.xml
  - lab_exercises_solution.xml
  - velocity.log

Contains every file and data you'll need for doing the exercises.

Contains the files needed by *ExerciseX-prepare* ant target to prepare the environment for an exercise.

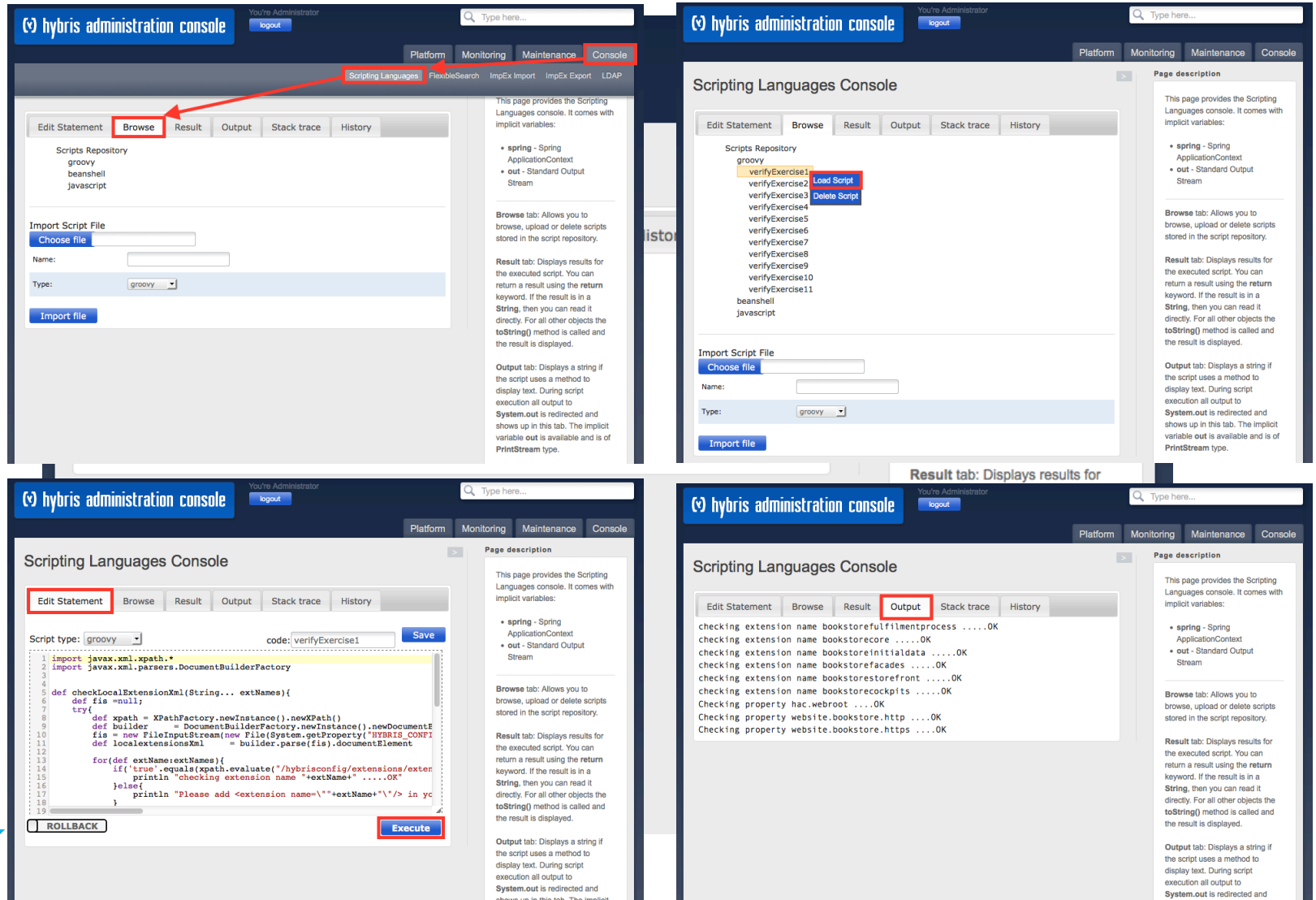
Contains the files needed to solve an exercise. You can compare your solution with them.

Properties like where the platform is located locally.

Contain ant targets that the tool uses.

Verification of Your Solution

- **Verification scripts are already put into platform**



Exercise 1