



HYBRIS ONE-ON-ONE

Day 1

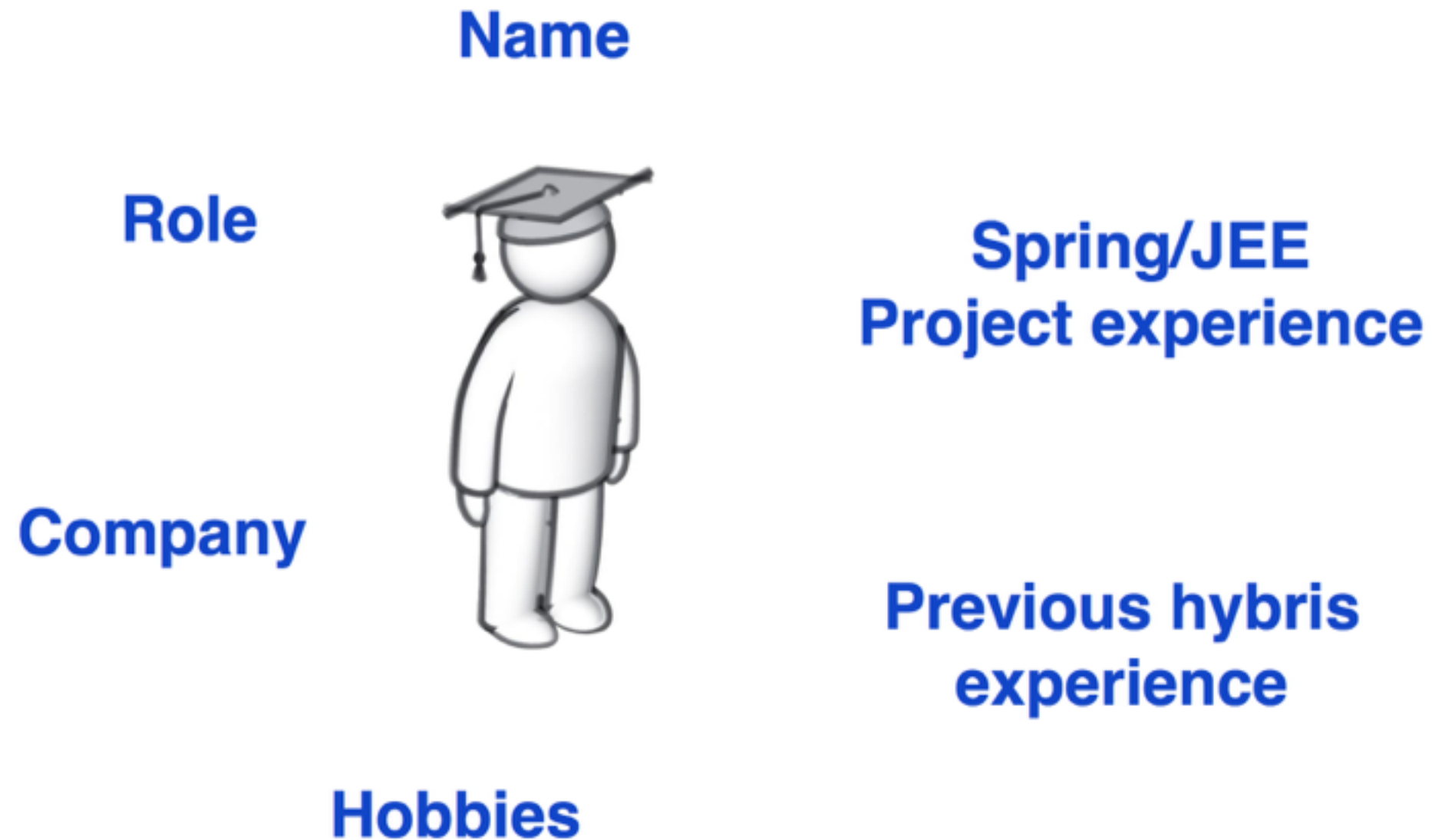
AGENDA

- Introduction
- About Hybris
- Architecture
- Hybris Server
- Deployment
- Hybris Installation
 - Build And Configure
 - Preparation and Trail
 - Extension, Add-on & Module Concept
- Eclipse Setup
- HAC
 - Initialisation and Update
 - Essential Data vs Project Data
- HMC

DISCLAIMER

- Content has been arranged from wiki.hybris.com
- Trainer holds no responsibility of the modifications in the content on hybris wiki (wiki.hybris.com) over the course of time and its differences to this content.
- This training is specific for hybris 5.7 and is not related to content which is going to come in future. A new training would be needed with changes in version.

INTRODUCTION

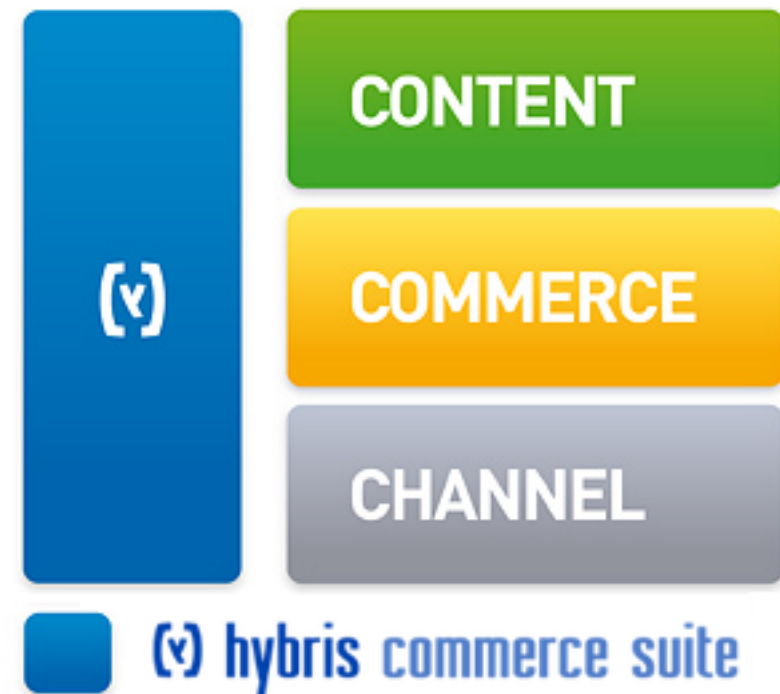


TRAINING CONTENT

- This course aims to:
 - illustrate the core telco functionality and main concepts of the hybris Platform
 - demonstrate the software
 - give the idea of what's available out-of-the-box and what has to be implemented
 - give you a chance to write a “hello world ” for each area presented
answer your questions
- This course won't:
 - solve project-specific problems
 - how hybris Commerce or PCM functionality
 - about hybris administration for that, there is hybris System Administrator Training

ABOUT HYBRIS

- The hybris Commerce Suite organizes data like product information to be propagated using multiple communication channels in a consistent and efficient way. This enables businesses to sell products across multiple distribution channels.
- Founded in 1997.

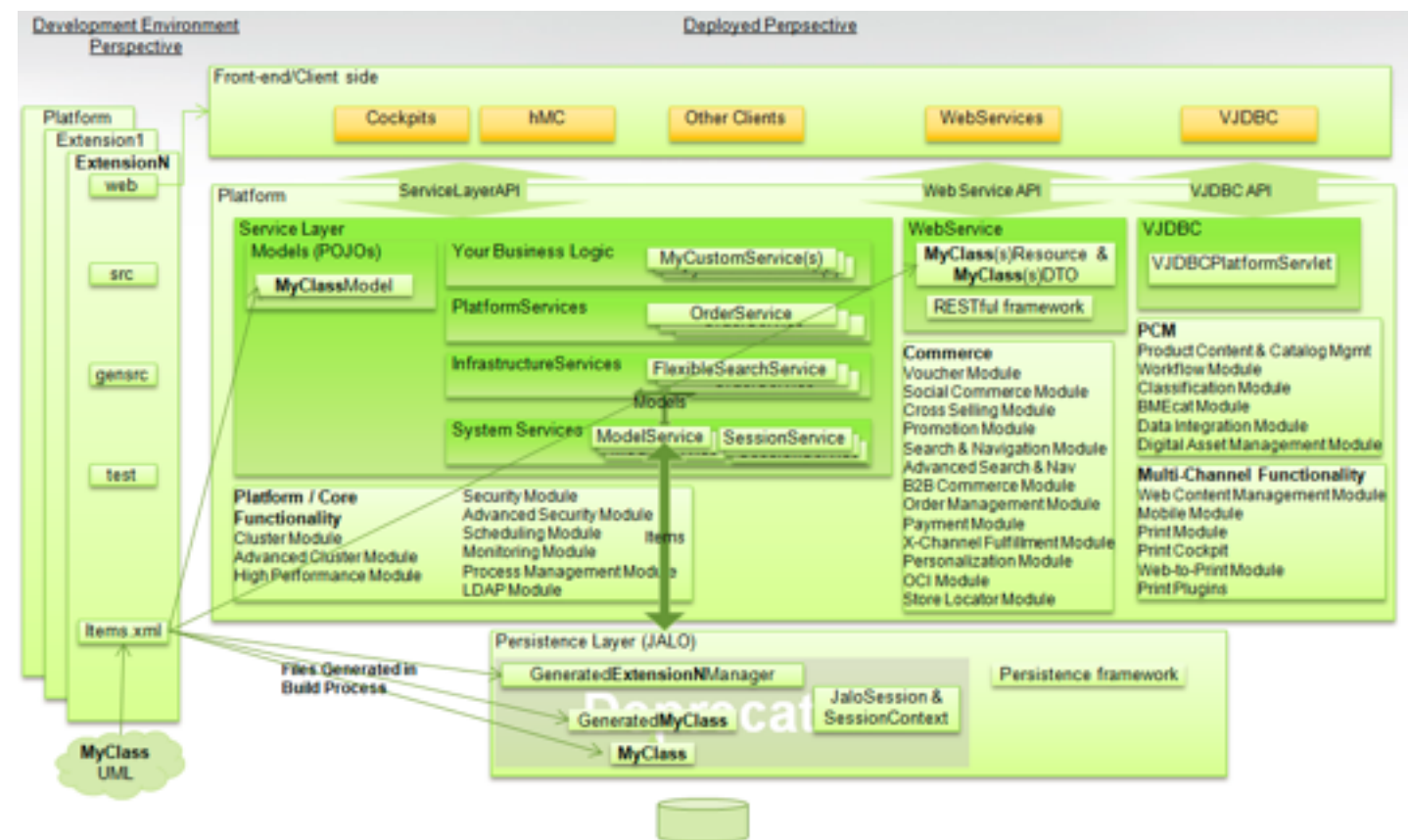


ABOUT HYBRIS

- Release Strategy 5.7.1.2 {Epic. Major. Feature. Patch}
 - Epic Release
 - New architecture, new build system etc.
 - Every 1-2 years
 - Major Release
 - New features
 - Every 6 months
 - Feature Release
 - New features which can be released without API changes
 - Every 2 months
 - Patch Release
 - Critical bugs fixed - released when necessary, no API changes
- Migration guides provided for Epic and Major releases
- Release notes are available on the wiki under Download page

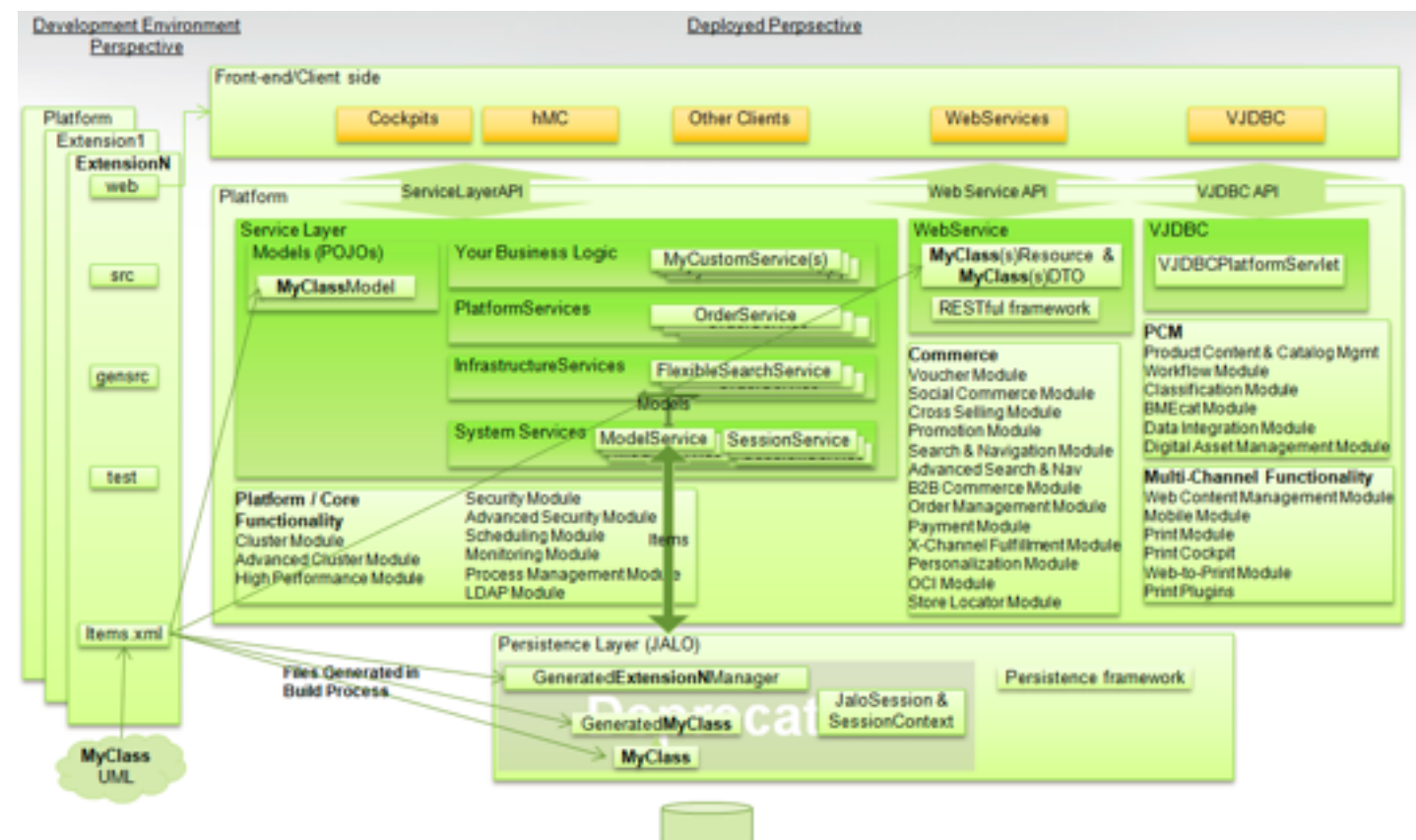
PACKAGE ARCHITECTURE

- The diagram below illustrates key concepts/features of the hybris Architecture that will be covered in this tutorial trail. Key points to take away from this diagram include: From a development perspective, the hybris Suite comprises of a platform holding the central hybris logic, multiple extensions and a persistence layer.
- The platform holds the central functionality of the hybris Suite: security, caching, jobs, JMX etc. The extensions written both by hybris and by partners contain functionality that builds upon the hybris Platform and focuses on business use cases and user stories.
- The persistence layer contains low-level persistence classes. While we used to encourage partners to implement business logic in the persistence classes, we no longer do so. Partners should implement their business logic in the services in the ServiceLayer.
- There are 3 main APIs that clients can use: the Service Layer API, Web Service API and VJDBC API. The Service layer follows a service oriented architecture, consisting of numerous services to which partners can add their own: see ServiceLayerPartners should code as much as possible in the service layer - creating new models and services as required. The services liaise with each other and with clients using Models, there are about 30 default services available ranging from high level platform services, to infrastructure and system services. The ModelService service converts from items (used in the persistence layer) to Models (used in the service layer). The Session Service is used to maintain Session-scoped data. The web service API is a RESTful webservice framework: see Web Services in the hybris Multichannel Suite write business logic in new services in the service layer



PACKAGE ARCHITECTURE

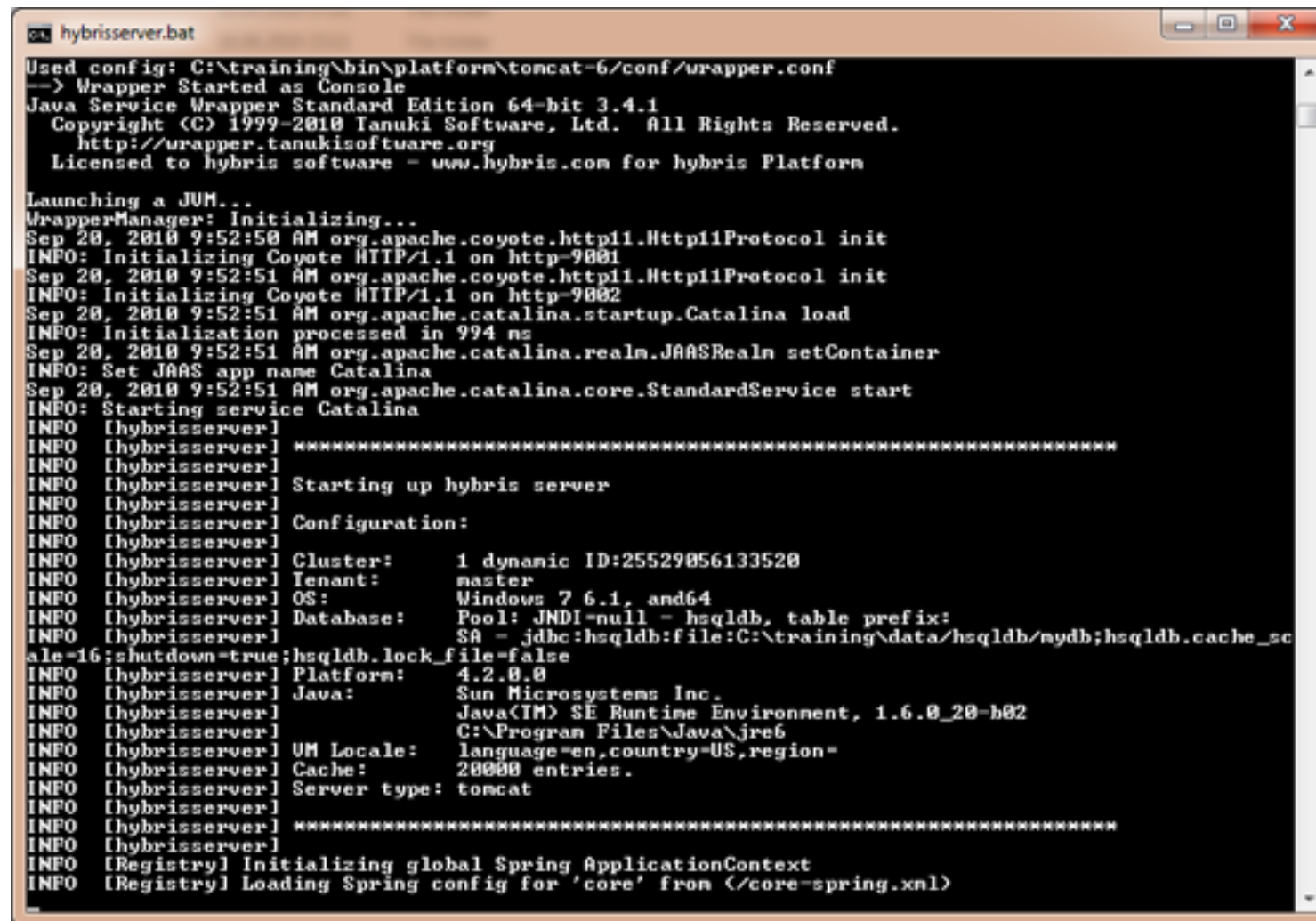
- The plumbing code is automatically generated such that all items specified in all extensions are automatically available via REST
- VJDBC offers secure remote access to the database: see VirtualJDBC Extension using both plain SQL and hybris' FlexibleSearch as communication takes place over HTTP(S), there are no firewall issues to introduce a new Data Model and associated logic, partners should create their own new extension specify their data model in the new extension's items.xml file.
- The hybris build process generates several files to support the back-end persistence of, and webservice access to the new items specified in the data model - see diagram below. In the diagram we assume `MyClass` is a new Data Model defined in items.xml, from which is generated a number of files:
- **MyClassModel.java** - a representation of the new Data Model in the service layer
- **MyClass(s)Resource.java** and **MyClass(s)DTO.java** - support for CRUD via the hybris RESTful web service And two low-level persistence-tier classes which you should not have to touch:
- **GeneratedMyClass.java** - low level persistence class for the model, which is regenerated on each build (so no custom logic should ever be added here)
- **MyClass.java** - inherits from GeneratedMyClass.java and since it is not regenerated if it already exists, it used to be possible to place custom logic here in previous releases of hybris. Custom logic now goes in the service layer. write business logic in new services in the service layer



ARCHITECTURE

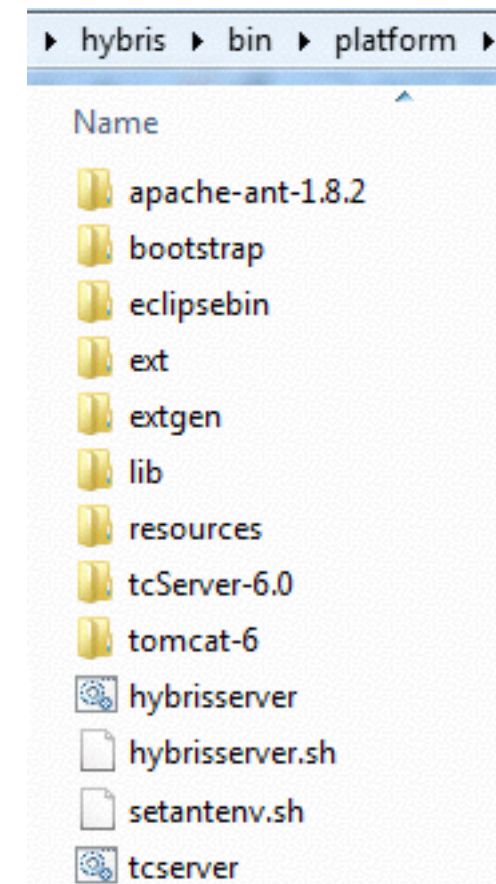
► Hybris Server

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



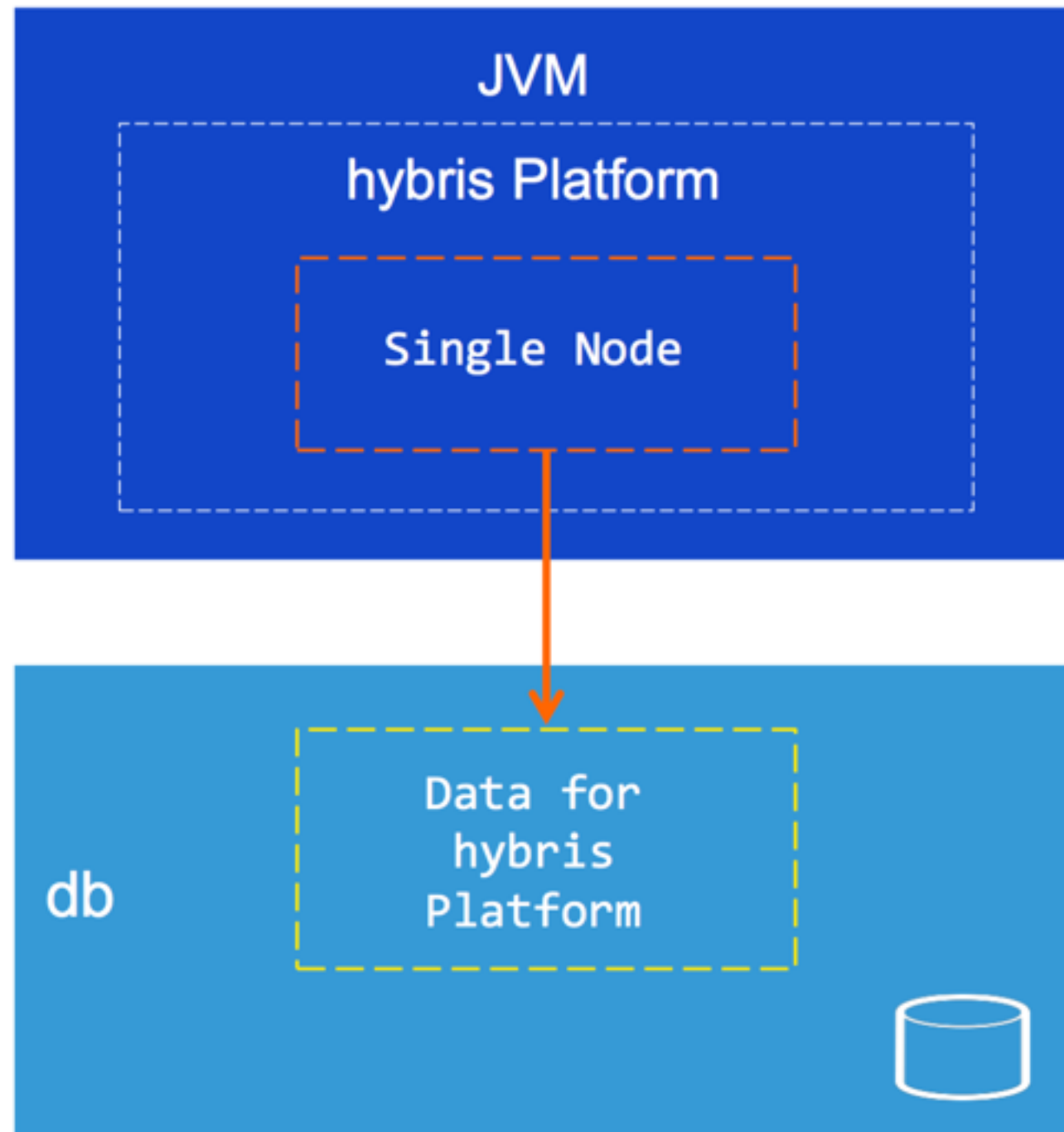
```
hybrisserver.bat
Used config: C:\training\bin\platform\tomcat-6\conf\urapper.conf
-> Wrapper Started as Console
Java Service Wrapper Standard Edition 64-bit 3.4.1
Copyright (C) 1999-2010 Tanuki Software, Ltd. All Rights Reserved.
http://urapper.tanukisoftware.org
Licensed to hybris software - www.hybris.com for hybris Platform

Launching a JVM...
WrapperManager: Initializing...
Sep 20, 2010 9:52:50 AM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-9801
Sep 20, 2010 9:52:51 AM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-9802
Sep 20, 2010 9:52:51 AM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 994 ms
Sep 20, 2010 9:52:51 AM org.apache.catalina.realm.JAASRealm setContainer
INFO: Set JAAS app name Catalina
Sep 20, 2010 9:52:51 AM org.apache.catalina.core.StandardService start
INFO: Starting service Catalina
INFO [hybrisserver] *****
INFO [hybrisserver] Starting up hybris server
INFO [hybrisserver] Configuration:
INFO [hybrisserver] Cluster: 1 dynamic ID:25529856133520
INFO [hybrisserver] Tenant: master
INFO [hybrisserver] OS: Windows 7 6.1, amd64
INFO [hybrisserver] Database: Pool: JNDI-null - hsqldb, table prefix:
INFO [hybrisserver] SA - jdbc:hsqldb:file:C:\training\data\hsqldb\mydb;hsqldb.cache_sc
ale=16;shutdown=true;hsqldb.lock_file=false
INFO [hybrisserver] Platform: 4.2.0.0
INFO [hybrisserver] Java: Sun Microsystems Inc.
INFO [hybrisserver] Java(TM) SE Runtime Environment, 1.6.0_20-b02
INFO [hybrisserver] C:\Program Files\Java\jre6
INFO [hybrisserver] UM Locale: language=en,country=US,region=
INFO [hybrisserver] Cache: 28888 entries.
INFO [hybrisserver] Server type: tomcat
INFO [hybrisserver] *****
INFO [Registry] Initializing global Spring ApplicationContext
INFO [Registry] Loading Spring config for 'core' from </core-spring.xml>
```



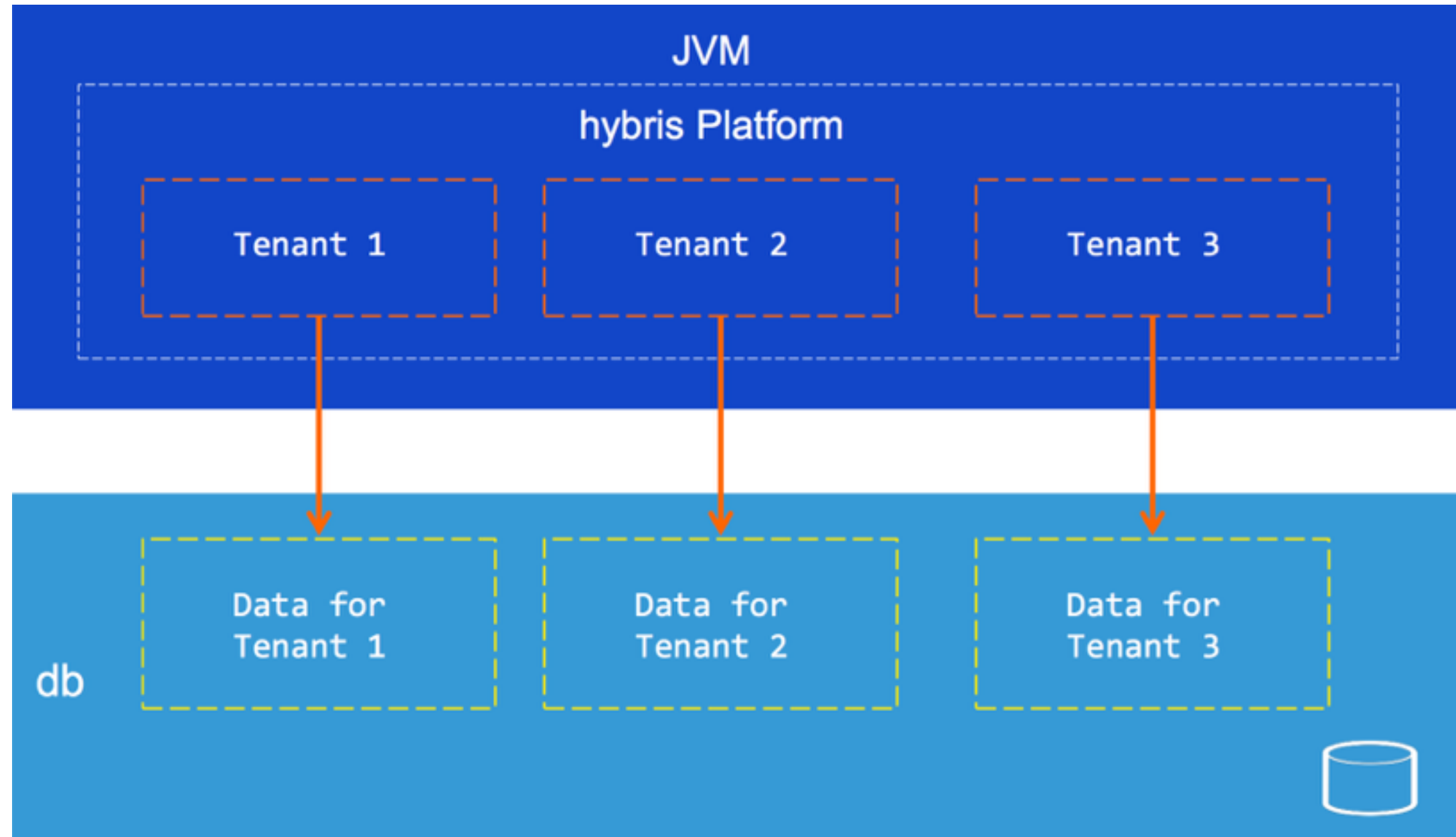
ARCHITECTURE – MODES OF OPERATION

➤ Single Tenant And Single Node



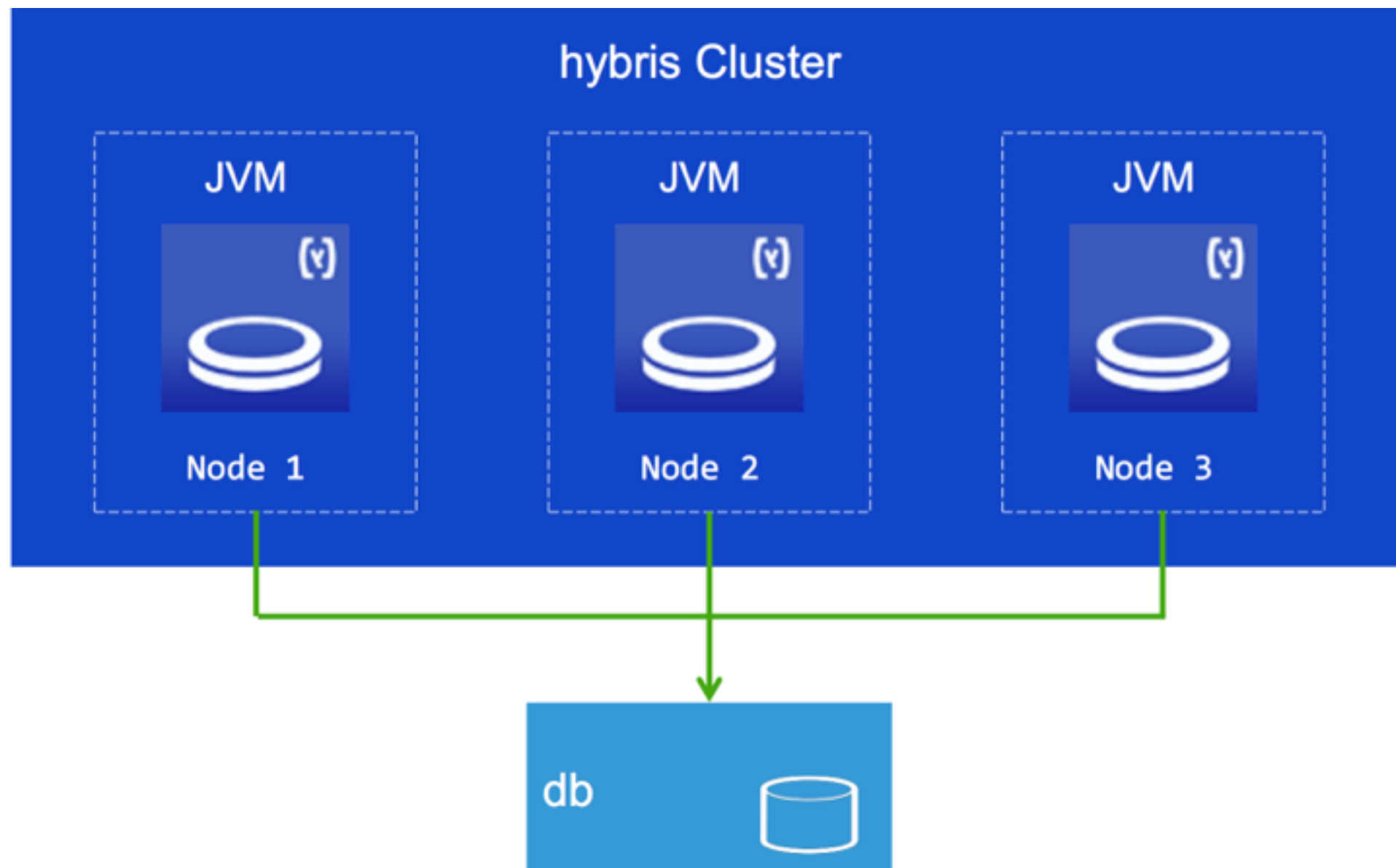
ARCHITECTURE – MODES OF OPERATION

➤ Multi Tenant / Single Node



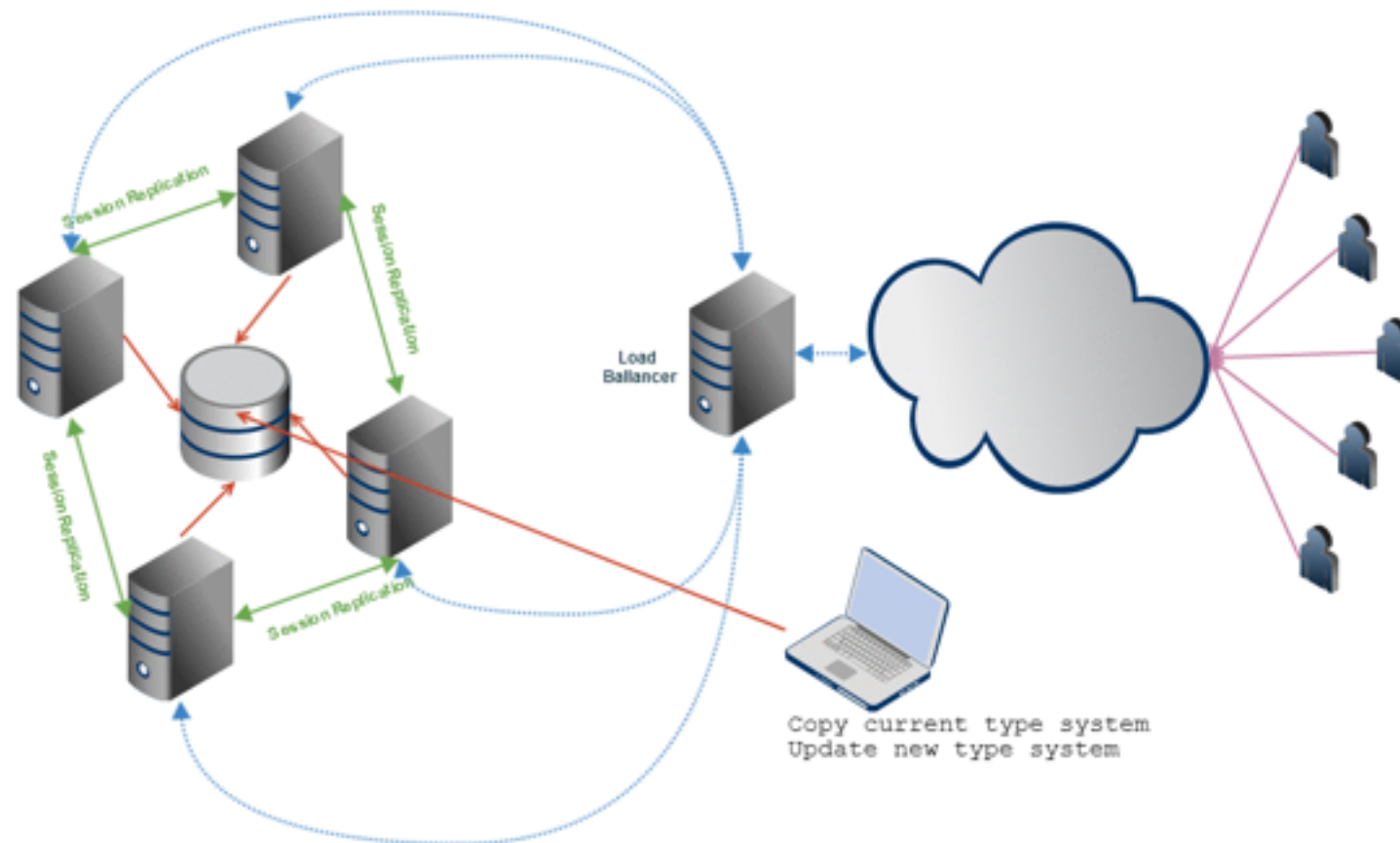
ARCHITECTURE – MODES OF OPERATION

➤ Cluster



ARCHITECTURE – MODES OF OPERATION

- Rolling update (Cluster only)
- Prior to 5.3 updating the type system required shutting down the entire cluster.
- Rolling update helps you eliminate cluster downtime during a system update. Each node has to be updated and restarted in sequence.
- Video



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
- 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.

HYBRIS SERVER

- Advantages: compared to other application servers with EAR deployment
 - Identical setup and usage for development, staging, and live system
 - Dramatically simplified installation, deployment, and configuration
 - Very small footprint – faster than WebLogic or WebSphere on the same hardware
 - Complete support for virtualized environments (“Ready for the cloud”)
- Free of charge, part of the hybris Commerce Suite (free security updates, for example)
 - Reduce hosting partners’ costs
- hybris Server Support is included in the standard hybris Service Level Agreements
 - Reduces customers’ cost for support and training

DEPLOYMENT – ON HYBRIS SERVER

- Call ant production in the `${HYBRIS_BIN_DIR}/platform` directory.
- The system will generate two ZIP files:
 - `hybrisServer-Platform.zip`
 - `hybrisServer-AllExtensions.zip`
- Copy both files onto the target machine.
- Unzip them and start the hybris Server
- For subsequent deployments: The same as above.

DEPLOYMENT – TC SERVER

- Via the local.properties file, you can set
- Which server type to use (hybris / tcServer)
Notice: you have to download the tcServer binaries yourself
- Where the bundled server is located
How your server instance is named
Which template to use (in case you have no instance yet)
- You have to use templates of bundled server.

```
# server type to use (tomcat or tcserver)|
bundled.server.type=tcserver
# path to tcserver
bundled.tcserver.home=${platformhome}/tcServer-6.0
# name of server instance
bundled.tcserver.instance=instance1
# path to instance template to use (hybris or hybris_insight)
bundled.tcserver.template=${bundled.tcserver.home}/templates/hybris
```


DEPLOYMENT – NON TOMCAT SERVER

- Call ant all cleanear ear in the `${HYBRIS_BIN_DIR}/platform` directory.
 - The system will generate:
 - `{HYBRIS_TEMP_DIR}/EAR/hybrisplatform.ear`
- Deploy this file to the Application server, and start the server normally
- Access hybris using the port number configured in the 3rd-party application server.

HYBRIS INSTALLATION

- Install hybris platform by either:

- Unzip from usb-stick

- After training use wiki download pages:

- wiki.hybris.com/display/release5/Download

- Or by using the hybris Package Manager (rather at production)

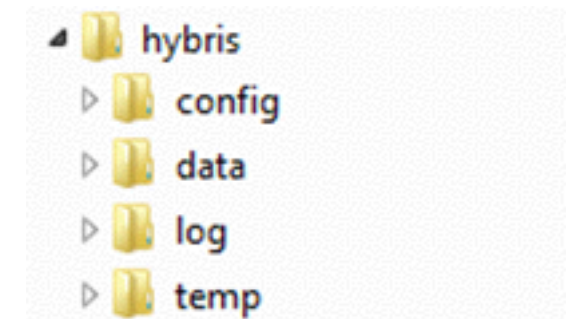
- Windows system users:

- Unzip into a directory close to the drive root and with no whitespaces in the name, such as C:\hybris

- In case ANT not installed

- Open a console window and go to hybris/bin/platform

- Call setantenv.bat (OSX/Linux: ". ./setantenv.sh")



- The hybris Platform has a build framework based on: Apache Ant for automation tasks

- The compiler from the Eclipse IDE

- Scope: platform-wide or extension-wide

- You can build the whole platform with all its extensions, or... build just one extension (this shortens the build process).

- There are predefined callbacks like Before/after compile

- Before/after build

- Before/after clean

- See buildcallbacks.xml for more

- When you call ant, the build framework: generates and compiles “Model” classes 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 folders (only in first run! as shown in the image)

HYBRIS INSTALLATION

► Build targets

Ant Target	Description
all	Builds and configures the server
clean	Cleans platform and all extensions
extensionsxml	Generates complete extensions.xml
runcronjob	Runs the specified cronjob
extgen	Runs extension generation tool
initialize	Initializes the hybris system with default settings
updatesystem	Updates the system
server	Configures hybris Server and restarts if running
startHybrisServer	Runs Easter egg
unittests	Executes all unittests
typecodetest	Checks if any reserved typecodes are being used
-p	Shows list of all Ant targets

HYBRIS INSTALLATION

- Configuration files
- Each extension has:
project.properties with extension specific configuration extensioninfo.xml with extension dependencies configuration
- For global settings and to override any extension specific configuration always use the config folder, it contains:
 - local.properties
 - Overrides any properties specified in each extension project.properties file
 - Introduces global properties like database url and credentials
 - localextensions.xml
 - List of extensions used by build framework

Beside that: configuration templates for bundled servers, language packs, license
- To use different configuration e.g. for test server: Run ant -Duseconfig=foobar
- Then localfoobar.properties file will be used

HYBRIS INSTALLATION

- 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 hac
- After server restart these changes are lost

HYBRIS INSTALLATION

- Local Extensions
- List of extensions used by build framework
- Dependencies are resolved automatically using the path parameter. Build will find dependent extensions which are not explicitly listed
- Extensions can be listed by name instead of location path It can be generated
- Define a lookup folder with the `<path>` tag (lazy loading):
- `<extensions>`
 - `<path dir="$ {HYBRIS_BIN_DIR}/ext-cockpit"/>`
 - `<path dir="$ {HYBRIS_BIN_DIR}/ext-hybris/">`
 - `<extension dir="$ {HYBRIS_BIN_DIR}/ext-hybris/cmscockpit"/> ...`
- All extensions found inside `dir="..."` are only pulled into the current extension configuration if referenced by another extension, or by name (see next slide).

HYBRIS INSTALLATION

► Load Extensions

► All Extensions in a folder

► Auto-loading of dependent extensions by path tag

► <extensions>

```
<path autoload="true" dir="${HYBRIS_BIN_DIR}/ext-cockpit" depth="3"/> <extension dir="${HYBRIS_BIN_DIR}/ext-hybris/hmc"/>
```

```
<extension dir="${HYBRIS_BIN_DIR}/ext-hybris/basecommerce"/> <extension dir="${HYBRIS_BIN_DIR}/ext-hybris/cms2"/>
```

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

► Extensions by name

► Extensions can be configured by name, provided one or more lookup folders have been specified:

► <extensions>

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

```
[echo] -----
[echo] --- Extensions (in dependency order, options:
[echo] --- p .. platform extension
[echo] --- * .. auto-required
[echo] --- ? .. lazy-loaded
[echo] --- i .. got <extname>-items.xml
[echo] --- b .. got <extname>-beans.xml
[echo] --- c .. got core module
[echo] --- w .. got web module
[echo] --- h .. got HMC module >
[echo] -----
[echo] core 5.0.0-SNAPSHOT [p*ci]
[echo] testweb 5.0.0-SNAPSHOT [p*w]
[echo] paymentstandard 5.0.0-SNAPSHOT [p*ci]
[echo] mediaweb 5.0.0-SNAPSHOT [p*cw]
[echo] maintenancweb 5.0.0-SNAPSHOT [p*w]
[echo] deliveryzone 5.0.0-SNAPSHOT [p*ci]
[echo] commons 5.0.0-SNAPSHOT [p*ci]
```

EXTENSION CONCEPTS

- Developing a new project with its own data model and business logic starts with the generation of a new extension.
 - That way 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.
- Examples:
 - Frontend extension: with all jsp pages, css, controllers
 - Testing extension: has all the test cases and data (such as testdata)
 - Infrastructure or utility extension (such as services)

EXTENSION CONCEPTS

- The module types

- Core

- For core functionality like data model definition.

- Web

- Enables WEB specific features and add a context root.

- HMC

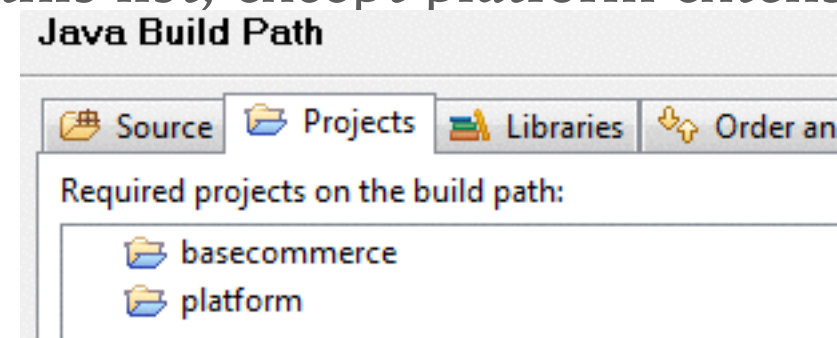
- HMC features. Allows modifying the HMC layout and item structure.

- Extensions are independent from other extensions by default except platform. Dependency can be added from the extensioninfo.xml for a given extension and dependency in IDE can added by modifying the build path.

- <!-- you should add all required extensions to this list, except platform extensions which are automatically required -->

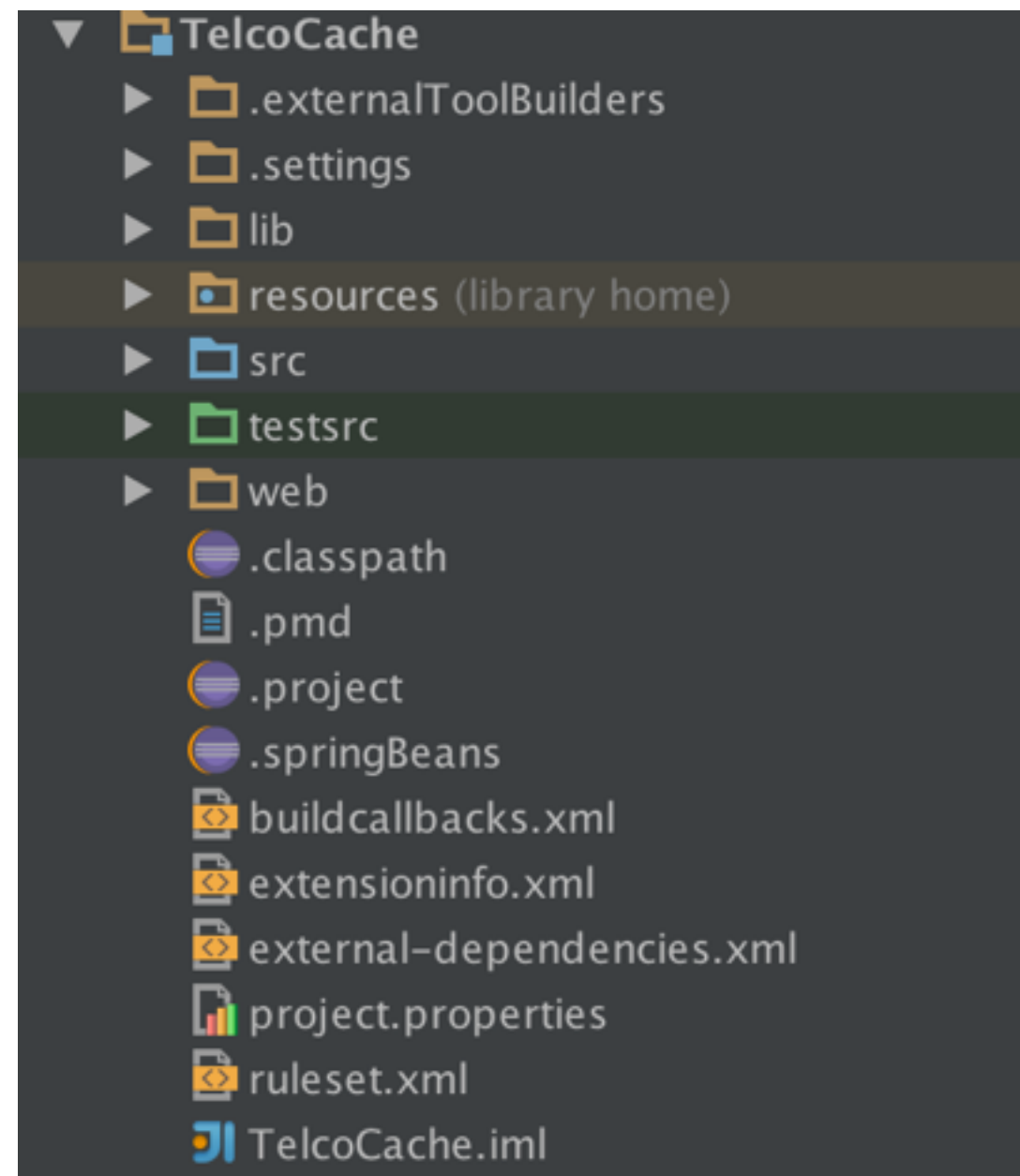
- <requires-extension name="cms"/>

- Allows easy to migrate and upgrade.



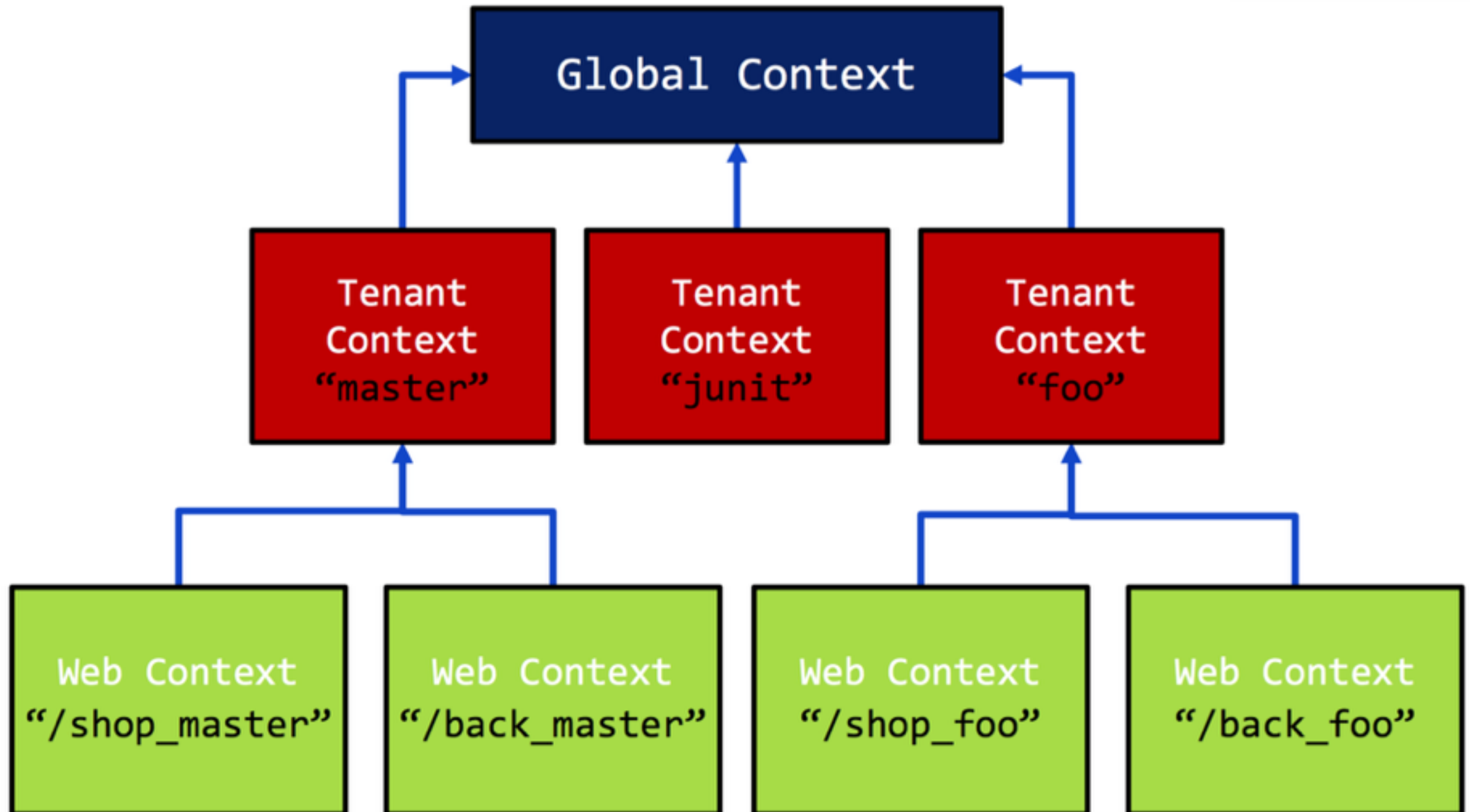
EXTENSION CONCEPTS

- Structure
 - Generated source
 - Java
 - hMC modules
 - Resources
 - localised files
 - type definition
 - externalised files
 - Library files
 - Web modules
 - Files for eclipse, apache and build.
 - Hooks into the build.



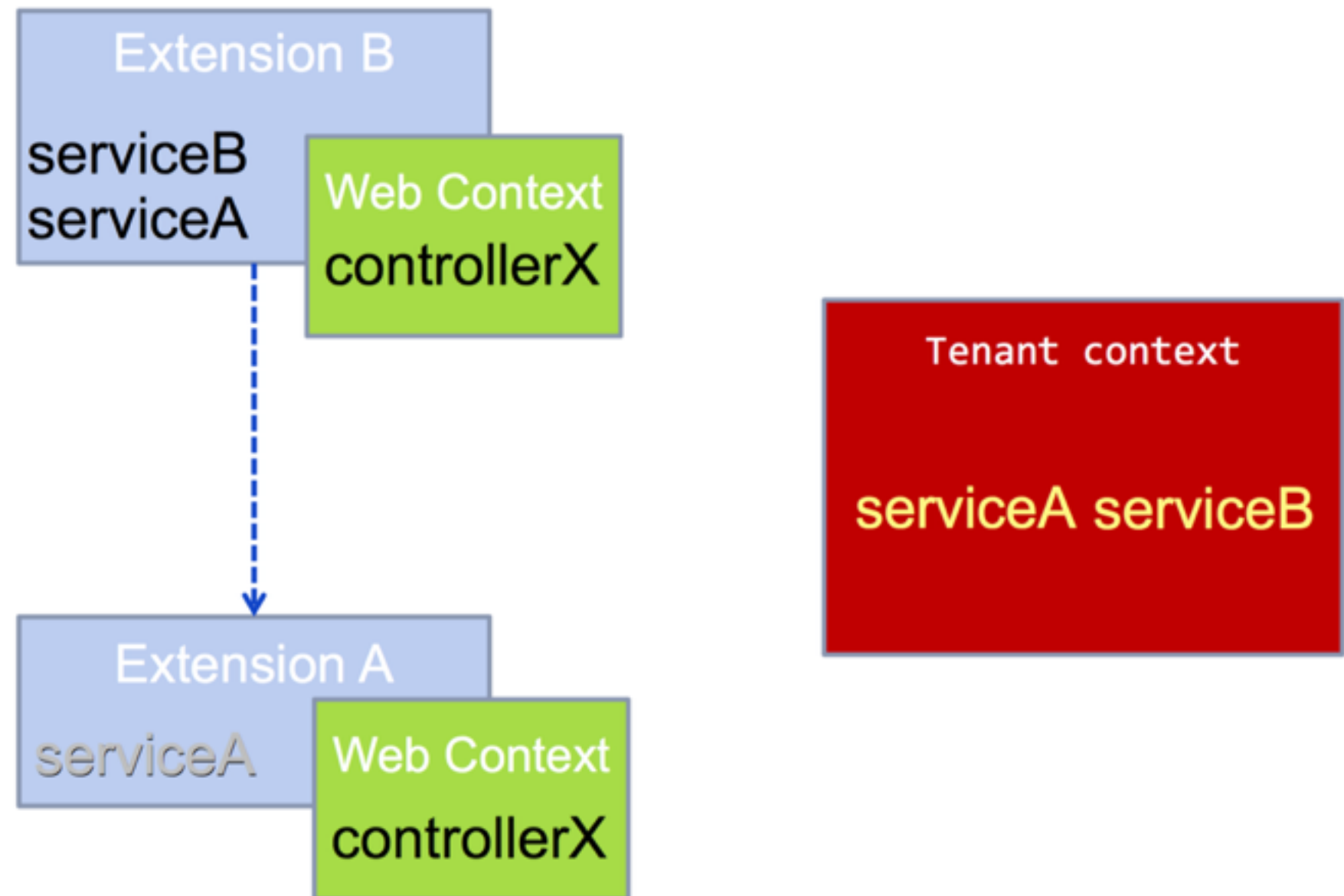
EXTENSION CONCEPTS

➤ Spring contexts



EXTENSION CONCEPTS

- Spring configurations of extensions
 - Beans from web contexts can access beans from tenant contexts.



EXTENSION CONCEPTS

- Spring Configuration
- 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

EXTENSION CONCEPTS

- Extgen tool
 - An ant target to help generate new extensions.
 - OOB templates. (yempty default template)
 - Configuration properties to add more custom templates and modify the default values for generating any extension.

Property	Description
extgen.template.default	The default template extension used for extgen. This is not used for modulegen.
modulegen.module.default	The preselected module that will be chosen if you do not enter a value for the module.
extgen.extension.path	The path where the new extension will be generated.
extgen.extension.name	The default name of the new extension if no name is chosen.
extgen.extension.package	The default Java package for the new extension's classes.

EXTENSION AND MODULE CONCEPTS

- The second step is to configure the **extgen.properties** file of each source extension. These files contain the following properties:

Property	Description
YEXTNAME_TOKEN	This is a token that is used during the cloning process to rename an extension with the value entered by user.
YMODULE_TOKEN	This is a token that is used during the cloning process to give a new name to the module.
YMODULE_PACKAGE_ROOT	This property is added to the root of the cloned extension package name for all classes.
YMODULE_CLASS_PREFIX	The suffix is used by the extension when used in modulegen. For example, "core" for yacceleratorcore.
YPACKAGE_TOKEN	This token in the package name will be replaced in the cloned extensions with the new package path.
YMANAGER_TOKEN	This manager token will be replaced in the cloned version with new value entered by user.
YCLASSPREFIX_TOKEN	The class prefix token will be replaced in the cloned version with the new value entered by user.
YGENERATED_TOKEN	The generated prefix is used for classes that will be generated during extension build process.

MODULE CONCEPTS

- Module is a combination of extensions which enable a complete feature. Examples: B2C, B2B etc.
- There are few OOB modules shipped with Hybris Commerce suite. Namely : Accelerator (B2C), B2BAccelerator, TelcoAccelerator (now an Addon), WebServices etc.
- Modules can be used as a base to kick start the complete development process for a client.
- B2C Module has following extension OOB
 - Core
 - Facade
 - Fulfilmentprocess
 - Initialdata
 - Test
 - Storefront
 - Cockpits

MODULE CONCEPTS

- Modulegen

- It is a wrapper over extgen .

- Creating a Customised Accelerator System Using modulegen

- The following procedure describes how to use the modulegen tool to create customised versions of the Accelerator B2C or B2B modules.
 - Configure the platform/extgen/project.properties file.
 - Configure the extgen.properties files for all the extensions that will be cloned.

The following example shows the extgen.properties file from the yacceleratorcore extension:

- YEXTNAME_TOKEN=yacceleratorcore
 - YMODULE_TOKEN=yaccelerator
 - YMODULE_PACKAGE_ROOT=core
 - YMODULE_CLASS_PREFIX=Core
 - YPACKAGE_TOKEN=de.hybris.platform.yacceleratorcore
 - YMANAGER_TOKEN=YAcceleratorCoreManager
 - YCLASSPREFIX_TOKEN=YAcceleratorCore
 - YGENERATED_TOKEN=Generated

- Run the modulegen ant task from `${HYBRIS_BIN_DIR}/platform`.
 - When prompted, enter a template for generation, the name of your module extension, and the name of the base package.
The default values for these properties can be configured in the platform/extgen/project.properties file, as described in the previous section.
 - When the ant task finishes, you will have a copy of the customised Accelerator project extensions. Using the example here, all cloned extensions are pl configurations in the .properties files, and the extensions are renamed as follows:
Telcotrainingcockpits Telcotrainingcore Telcotrainingfacades Telcotrainingfulfilmentprocess Telcotraininginitialdata Telcotrainingstorefront
Telcotrainingtest

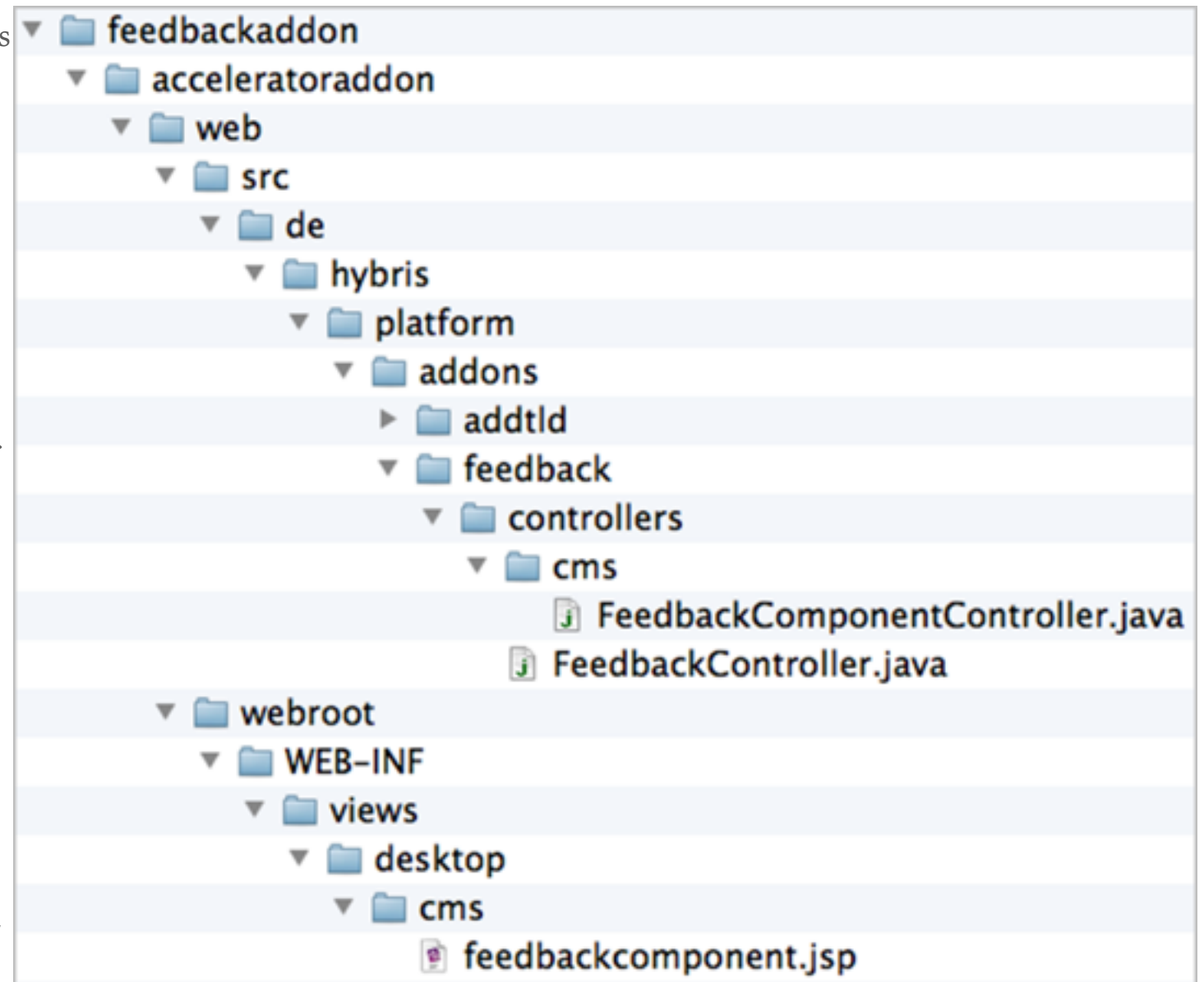
- Add your extension to your `${HYBRIS_CONFIG_DIR}/localextensions.xml` file.

ADD-ON CONCEPTS

- Add-on's are type of extensions which allow you to add front end files.
- You can add your facade data access objects, you can plug populators into existing converters without redefining them.
- Add-on's have to registered in the locaextensions.xml file.
- Quite similar structure to that of an extension.
- The AddOn concept aims at extending the Accelerator storefront without touching its core code base. For example, you could implement customer feedback functionality on your storefront without editing the source code. Instead, you would plug in all the functionality from within your AddOn.

ADD-ON CONCEPTS

- In this example, the **web** folder contains both the **src** and the **webroot** subdirectories. After putting all the components you want to add into the proper folders, you need to run the following command on your system: **ant build<Enter>**.
- The system automatically copies the files to the target storefront extension during the build phase, and creates two additional folders containing
- After the build phase finishes, the target extension contains the following new folders:
 - **web/addonsrc** , which contains the source code for each installed AddOn. This gets compiled automatically.
 - web/webroot/WEB-INF/addons**, which contains all the front-end components, such as images, **.jsp** files, **.html** files, and **.tag** files .
- All the contents of these folders are processed automatically during the development phase. By default, this mechanism is disabled on the production systems so as not to impact the performance of the system. This is because the process continuously checks if the contents of the folders have changed. Benefits of the AddOn Approach
- The benefits of the AddOn Concept approach are as follows:
 - AddOn files are kept separate from the rest of the front-end files.
 - When you upgrade the Accelerator, it will not overwrite your files.
 - You can easily remove your AddOns without refactoring the code of your whole extension.



ADD-ON CONCEPTS

- Installing an Add-on
 - Register the Add-on in the localextensions.xml
 - Execute the ant target of
 - `ant addoninstall -Daddonnames="<myaddonname>" -DaddonStorefront.yacceleratorstorefront="<mystorefrontextensionname>"`

HYBRIS INSTALLATION

- Hybris Package Structure.
 - Hybris : All content for Hybris commerce suite.
 - Installer : for installing different Hybris OOB recipes.
 - Recipe : A set of packages providing a given feature.
 - We would be using the telco recipe
 - Its a new way of doing setup from 5.7

- To list recipe go to Installer Folder
- Fire the command `install.bat -l`.
- Fire command `install.bat -r b2c_telco setup`
- Make configuration changes to connect to MySQL DB.
- Fire command `install.bat -r b2c_telco initialize`
- Fire command `install.bat -r b2c_telco start`
- fire command `install.bat -r b2c_telco stop`



HYBRIS INSTALLATION

➤ Configurations for hybris/config/local.properties

```
#Generated by hybris installer
#Wed Oct 28 00:37:18 IST 2015
cis.client.subscription.mock=true
#####
# Default desktop UI experience to indicate the appropriate resources to load
# Available options are 'responsive' or 'desktop' (the original accelerator desktop exp)
# Indicates to the import services which impex to load on initialization and view resolvers which path
# for resources
commerceservices.default.desktop.ui.experience=desktop

cronjob.timertask.loadonstartup=false

mail.from=*****@gmail.com
mail.replyto=*****@gmail.com
mail.smtp.server=smtp.gmail.com
mail.smtp.port=587
mail.smtp.auth=true
mail.smtp.debug=true
mail.smtp.user=*****@gmail.com
mail.smtp.password=*****
mail.use.tls=true
mail.smtp.starttls.enable=true

website.b2ctelco.http=http://b2ctelco.local:9001/Telcotrainingstorefront
website.b2ctelco.https=https://b2ctelco.local:9002/Telcotrainingstorefront

# Specifies the location, from which the hMC retrieves its configuration:
# -- "true" - for the database
# -- "false" - for the server's file system
# Use "true" for clustered systems.
hmc.structure.db=false

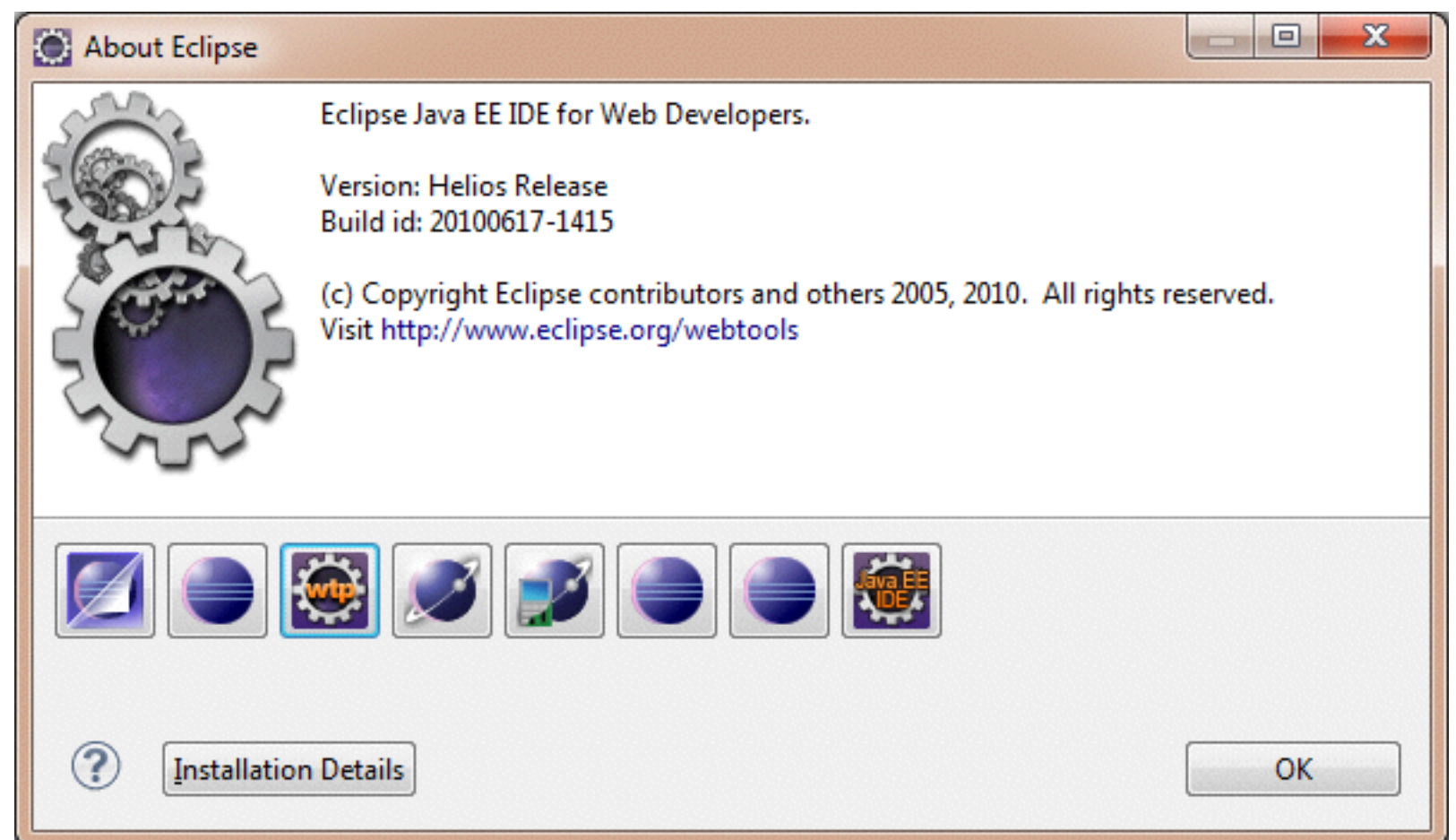
# System language used by default in the hMC
hmc.language=en

# Set to "true" to enable automatically logging into the hMC,
# but it makes only sense during the development, because of the security risk.
hmc.default.autologin=true

db.url=jdbc:mysql://localhost:3306/TelcoTraining?useConfigs=maxPerformance&characterEncoding=utf8
db.driver=com.mysql.jdbc.Driver
db.username=root
db.password=
db.tableprefix=
mysql.optional.tabledefs=CHARSET=utf8 COLLATE=utf8_bin
mysql.tabletype=InnoDB
```


ECLIPSE SETUP

- You can use an out-of-the-box Eclipse installation.
- Be sure to use the latest Eclipse version IDE for Java EE Developers:
 - www.eclipse.org/downloads
- Or Spring Tool Suite



ECLIPSE SETUP

- Before coding in Eclipse..
 - We advise familiarity with Hybris development documentation in the Wiki. In particular, check out “The coding landscape” under wiki.hybris.com/display/general/Development+Landscape and the links you find there.
 - Hybris approach to ensuring Code Quality – see also
 - wiki.hybris.com/display/general/Code+Quality :
 - Code Reviews
 - Code walkthroughs
 - Coding conventions
 - Test Suites

ECLIPSE SETUP

- The hybris Suite comes with configuration files for the Eclipse IDE.
- Changes in an items.xml file automatically trigger:
 - source file generation (for example, web service resources)
 - model generation
 - a refresh of the project in which the items.xml file was modified
 - a refresh of the gensrc directory of the ServiceLayer extension
- This relies on general Eclipse tools.

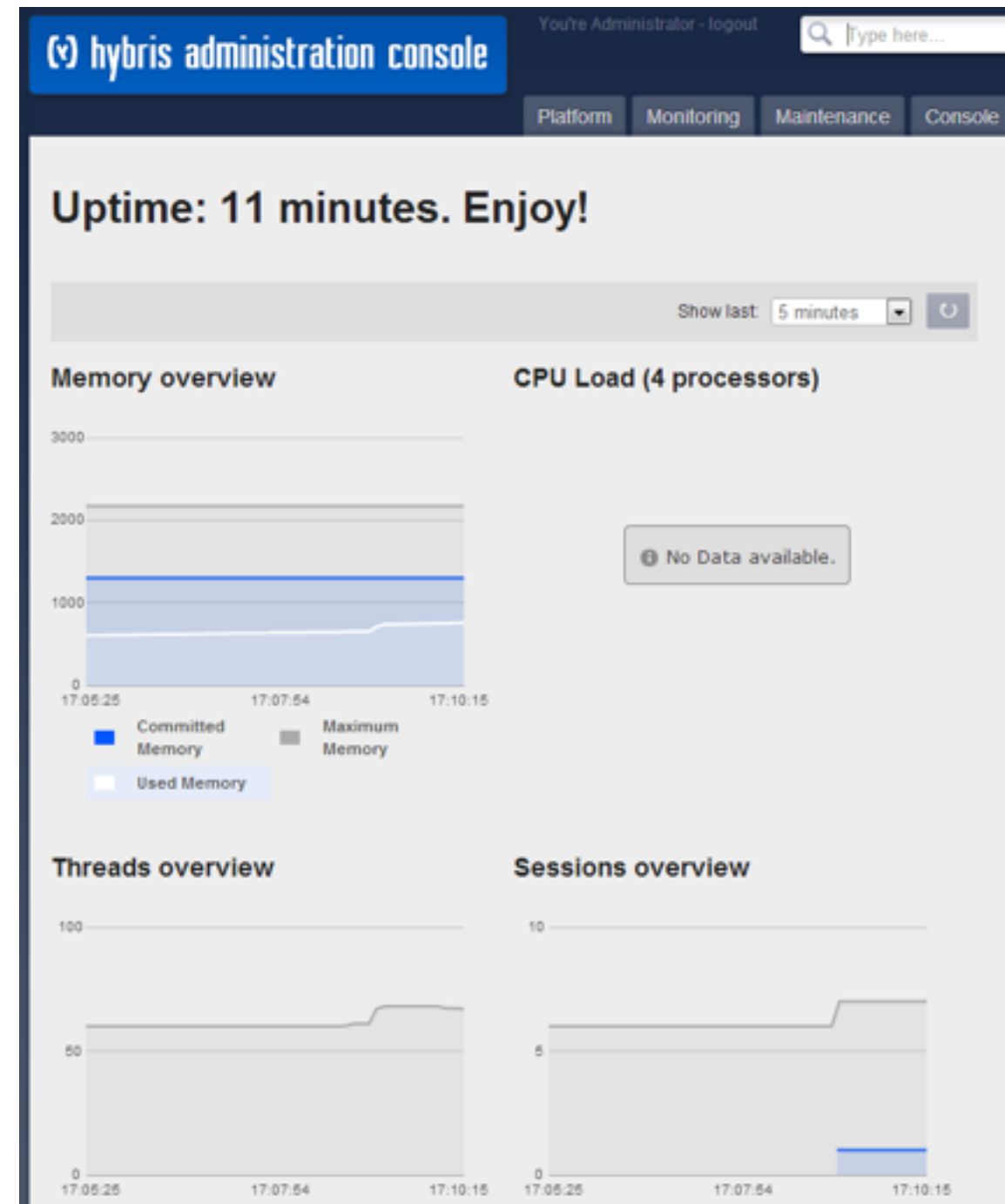
ECLIPSE SETUP

- Importing hybris suite
 - Only import the projects which are mention in the localextensions.xml and these extension dependencies (extensioninfo.xml)
 - Also import platform , platformhmc
 - Also import config
- Debugging with eclipse IDE
 - Start the hybris server in the debug mode
 - you can change the debug setting in the local.properties
 - In Eclipse go to Run —> Debug Configurations
 - create a new remote application
 - connection type : standard
 - Host Port — localhost : 8000 (configurable in local.properties)

HYBRIS ADMINISTRATION CONSOLE

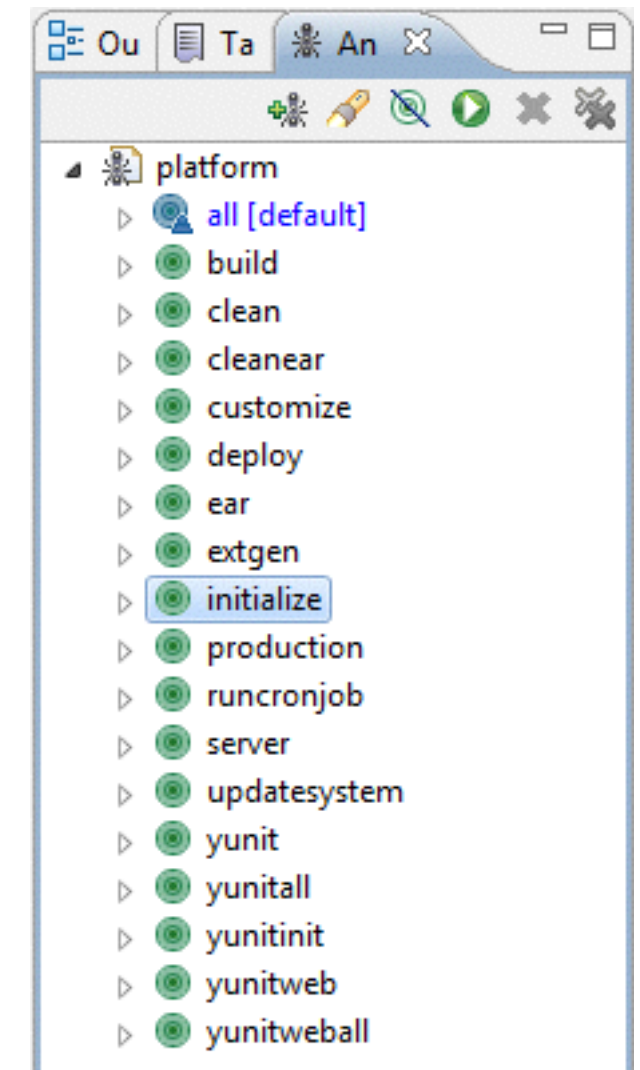
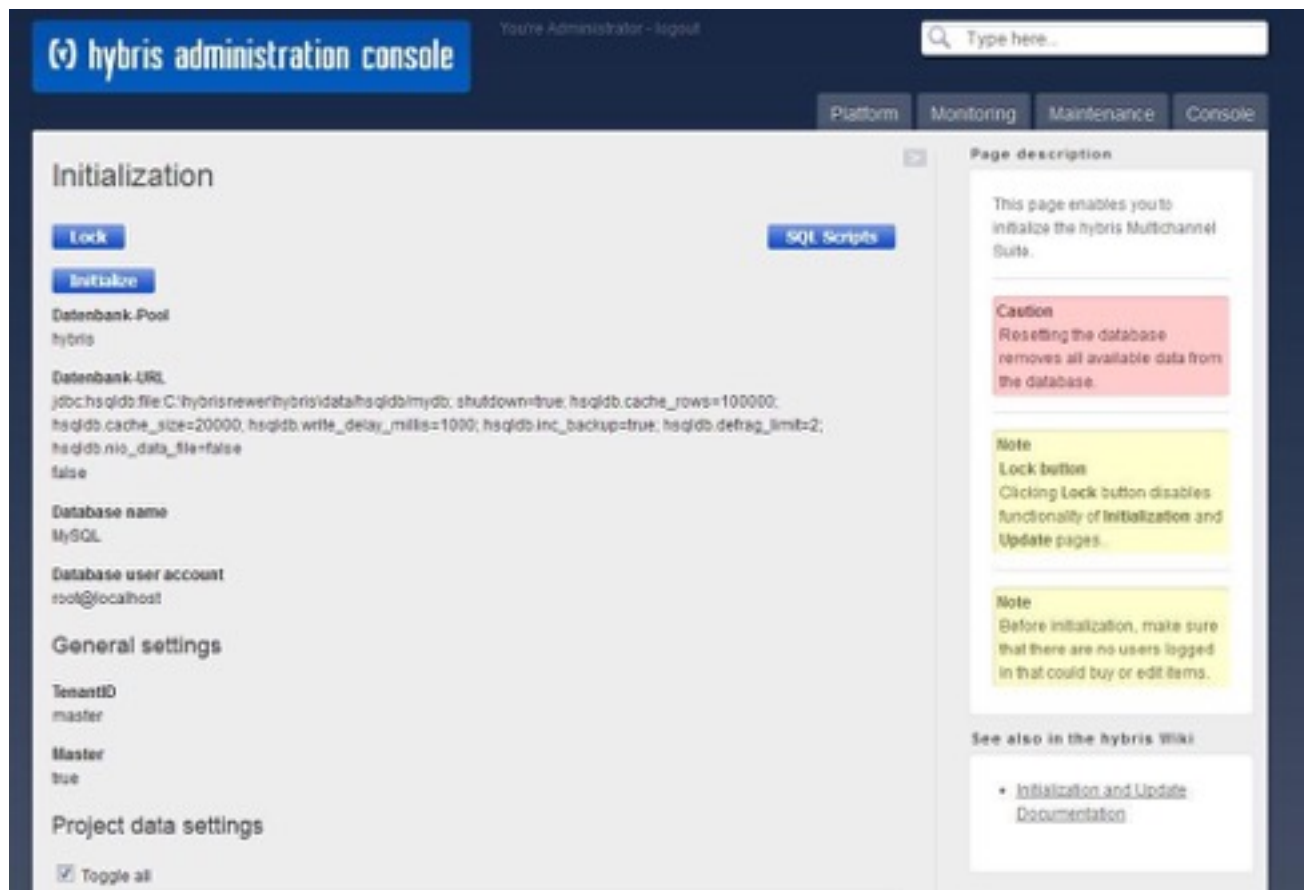
.....

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



INITIALISATION AND UPDATE

➤ hac -> Platform -> Initialise



➤ Ant view in eclipse + tenant

➤ Command line

```
C:\hybris\bin\platform>ant initialize -Dtenant=master
```


ESSENTIAL DATA VS PROJECT DATA

- Essential Data

- Necessary during initialisation:

- 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

HYBRIS MANAGEMENT CONSOLE

- Available at <http://localhost:9001/hmc/hybris>
- Item-oriented view
- Everything is displayed as it is defined in the items.xml
- Useful for developers during the implementation phase Check the data model
 - Manage complex business processes (imports, automated tasks)
 - Manage user-related settings (access rights, groups)
- Not recommended for business/end users (too complex, too technical)
 - Use different Cockpits instead, such as the Product Cockpit
 - For more complex or proprietary processes, create your own Cockpits.

HYBRIS MANAGEMENT CONSOLE

► Tree Structure

The screenshot displays the Hybris Management Console interface. The top navigation bar includes the user 'Administrator@localhost/master:9001: Product' and the 'hybris platform' logo. A toolbar with icons for 'New', navigation, and user management is present. A 'Quick Search' field and a 'Language' dropdown (set to 'English') are also visible.

On the left, a tree structure is shown, with the 'Products' node highlighted. The tree includes nodes for 'System', 'Catalog', 'Catalog Browser', 'Catalogs', 'Categories', 'Products', 'Product Variant Types', 'Units', 'Keywords', 'Catalog Management Tools', 'Classification Systems', and 'Multimedia'.

The main content area features a search interface. It includes a 'Type' dropdown (set to 'Product') and a 'Category' field. Below this is a table with columns: 'Attribute', 'Locale', 'Comparator', and 'Value'. The table contains the following rows:

Attribute	Locale	Comparator	Value
Article Number		contains	
Identifier	de	contains	
Product variants type		is equal	
Catalog version		is equal	Default - Staged

Below the table is a 'Search' button and a checkbox for 'Include Subtypes'.

A red box highlights the tree structure and the search interface. A red arrow points from the 'Products' node in the tree to the search interface. Below the red box, the XML structure for the tree is shown:

```
<explorertree width="185" columns="2">
  <group name="system" expand="false" description="group.system.description">
  </group>

  <group name="catalog" expand="false" description="group.catalog.description">
    <typeref type="Product" description="typeref.product.description"/>
    <typeref type="Unit" description="typeref.unit.description"/>
  </group>
```

HYBRIS MANAGEMENT CONSOLE

- Storage location for xml definitions
- All hmc.xml configuration files are assembled into one final file during the system build
- The final hmc.xml is loaded into database during the system update
- In order to avoid lengthy system update during the development, it's recommended to specify:
 - the `hmc.structure.db=false` property in the `local.properties` file.
- It's possible to upload this file in the hMC manually:
 - Go to group "System" -> "hMC configuration"
 - Click on the button "hmc.xml upload"

PRACTICE QUESTIONS

- Explain the concept of extensions in hybris
- Is it possible for an extension to have dependencies on other extensions?
- Where do you specify extension dependencies?
- Explain the concepts of global context and web context
- What does “ant extgen” do?
- What are extension templates?
- What does “autoload = true” do?
- What does ant -p do?
- How can you build hybris?
- What is configured in extensioninfo.xml?
- Do you need to shut down the server while executing ant all?

CONTACT

- Available at abhaykumartr@gmail.com