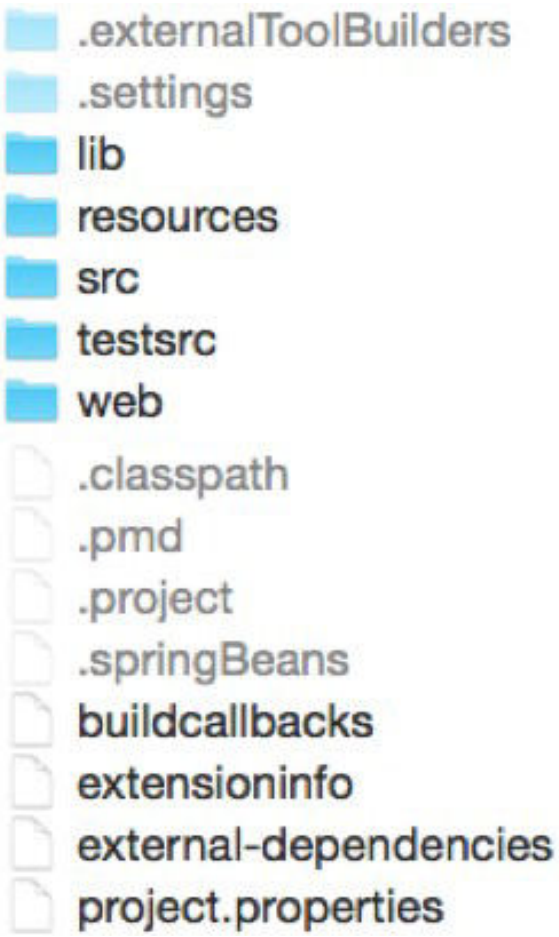
About Extensions

- The hybris Commerce Suite is based on a flexible, modular concept that allows you to put new functionality into extensions.
- An extension is an encapsulated piece of the hybris Commerce Suite that can contain business logic, type definitions, a web application, or a hybris Management Console (hMC) configuration. That way, you link up in one place all of the functionality that covers a certain field of use, for example a webshop.
- Because extensions are independent of one another by default, you can migrate an extension from one hybris Commerce Suite version to another.

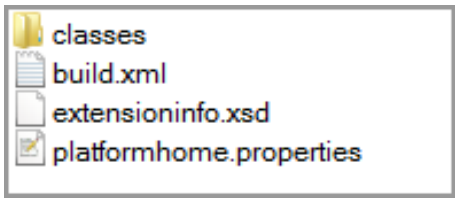
Basic Structure of an Extension



Directory or File of Blank Extention	Description
`.classpath` file	Classpath file for Eclipse.
`.pmd` file	Control file for PMD with hybris specific settings. Control file for PMD with hybris specific settings.
`.project` file	Project file for Eclipse.
`.ruleset` file	Ruleset file for PMD with hybris-specific settings.
`.springBeans` file	Configuration for Spring IDE Eclipse plugin.
`/.externalToolBuilders` directory	Contains the Eclipse builder that automatically generates hybris Models if any extension's **`items.xml`** file has been modified in Eclipse.
`/.settings` directory	Contains configuration files for Eclipse in general and Eclipse-related Spring configuration files.
`/lib` directory	All external library files should be stored here.
`/resources` directory	Contains the *$extension* **`-items.xml`** file and the localization for the extension types (within the **`/localization`** subdirectory).
`/src` directory	Contains the source code files for the extension's **core** extension module.
`/testsrc` directory	JUnit test files for the extension are located here.
`/web` directory	This directory contains the **web** extension module parts of the extension (JSP files, libraries). Sources and resources from this directory are only accessible to a Web application containing the extension. They cannot be accessed from another extension.
buildcallbacks.xml	Allows you to implement custom build framework logic.
`extensioninfo.xml` file	Configures the extension modules. This file is used by the build framework. It also determines which extensions are required for proper functioning.
`external-dependencies` file	Third party dependencies of Hybris Commerce extensions are defined here.

`project.properties` file	Configuration properties for the extension.

The directories and files above exist after the extension is generated (or after **`ant clean`** is run). During an extension build, some additional directories and files are created:



Directory or File of Blank Extention	Description
`/classes` directory	Contains all **`.class`** files for **`src`** , **`testsrc`** and **`gensrc`** folders of the extension.
`build.xml` file	This build file calls the platform's build file for actual action and provides the necessary parameters to it. This way **`ant`** targets are consistent throughout the entire installation.
extensioninfo.xsd	Copied from *${HYBRIS_BIN_DIR}* **`/platform/resources/schemas`** . Allows validating the **`extensioninfo.xml`** file.
platformhome.properties	Automatically generated file. It contains only the relative path to *${HYBRIS_BIN_DIR}* **`/platform`** . Do not modify it.

Checking Installed Extensions

To check the installed extensions of the hybris Commerce Suite do the following:

1. Open the hybris Administration Console.
2. Go to the **Platform** tab and select **Extensions** option.
3. The **Extensions** page displays with list of all installed extensions

Extension templates

The following extension templates are available with the hybris platform:

Extetnion Template	Descrption
yempty	Skeleton for an extension with a core and web module
ycockpit	Sample cockpit application
ycommercewebservices	Provides a working example how a REST API can be exposed dependent on the hybris commercefacades
ybackoffice	Template to generate a backoffice extension to start developing widgets
yaddon	Base template for writing Accelerator AddOns.

and other Templates are but these templates used rarely

yacceleratorfulfilmentprocess, yoccaddon, yacceleratorordermanagement, yatddtests, ygroovy, yscala, merchandisefulfilmentprocess, ycommercewebservicestest, ycommercewebserviceshmc, ychinaacceleratorstorefront,

Task1:Create A NewExtention

Here createas the lycatrail Extention create your own Extetnion wich you want.

- Open the Ant view in Eclipse and invoke the platform's task ant extgen. This will ask you to

  1.Select yempty for the extension template
  2. Enter the name **lycatrail** for the extension.
  3. Enter the package prefix com.techouts.lycamobile

- The Ant script will now create a skeleton extension for you in YOURPATH/bin/custom/**lycatrail** but this is not yet imported into Eclipse.
- Append the new extension to config/localextensions.xml:

config/localextensions.xml

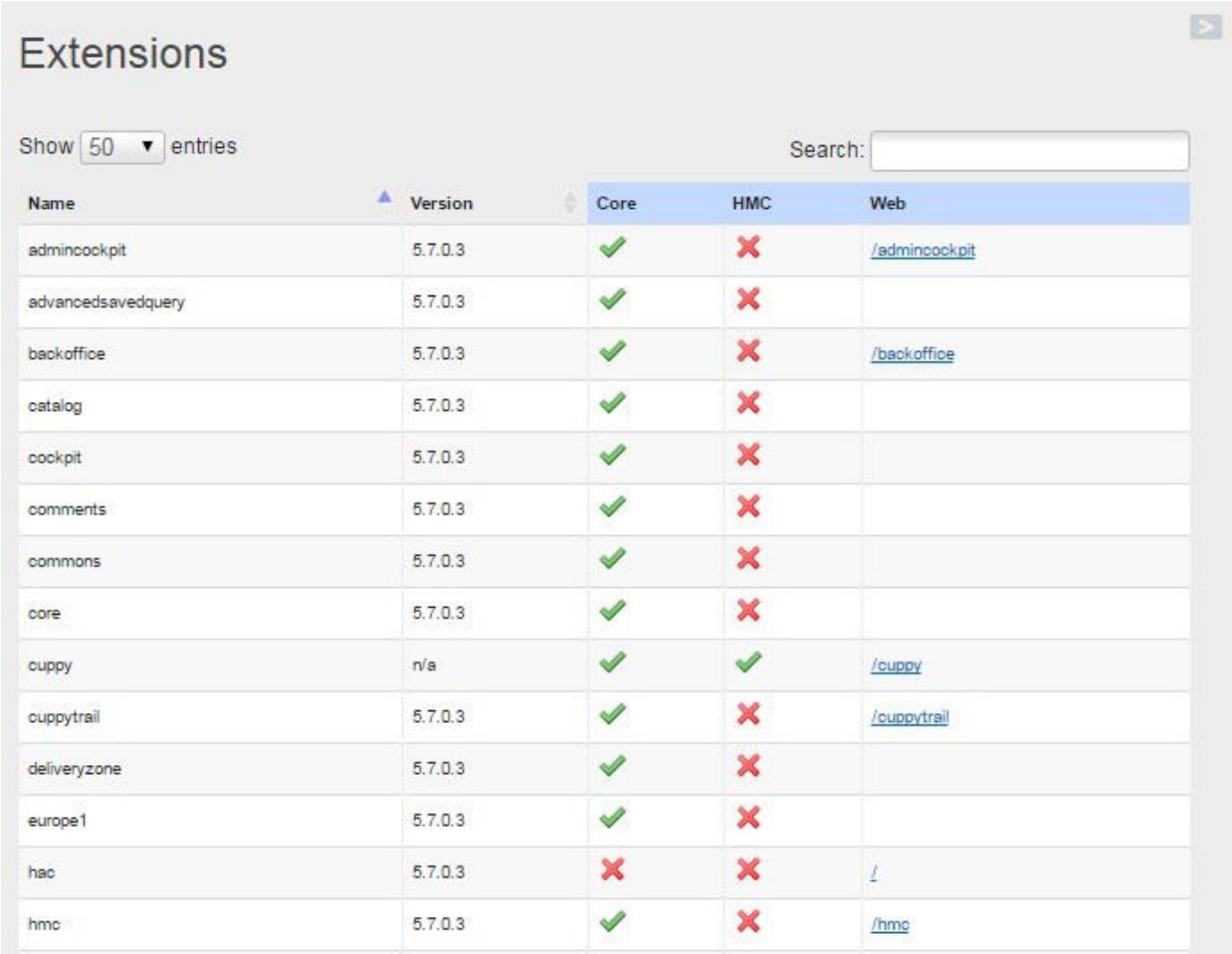
```
<extensions>
    ...
    <extension name="lycatrail" />
</extensions>
```

- Run **ant all**
- Start the application server

- Import the generated extension into your Eclipse environment

1. Right-click in package explorer and select Import
2. Select **General|Existing Projects** into Workspace and browse to the new extension `YOURPATH/bin/custom/cuppytrail`
3. Make sure that the **Copy projects into workspace** check box is not checked before clicking on the **Finish** button
4. You should now see the extension in your Eclipse Package Explorer

- Open the hybris administration console http://localhost:9001/platform/extensions in a web browser and log in as *admin nimda*. You will get an overview on the available extensions and, if the extension comes with a core extension module, an hmc extension module, a webmodule depicted by the webroot and a core + extension module. As you can see, **lycatrail** contains a core module and a web module.



About AddOn

AddOns are built on top of the existing hybris Commerce Suite to extend the functionality of the hybris Commerce Accelerator. AddOns are a type of extension that allow you to add front-end files (such **.jsp**, **.html**, **.css**, and javascript files, as well as images) from within your own AddOn, instead of modifying the storefront front-end files directly.

The overall extensibility of the hybris Commerce Accelerator is also improved in the following ways:

- You can generate your facade Data Transfer Objects (DTOs) and extend them, even if they are not defined in your own AddOn.
- You can plug populators into existing converters without having to redefine them.
- Some facades and services have been refactored so that they can be easily plugged in and their behavior can be easily modified.

Benefits of the AddOn Approach

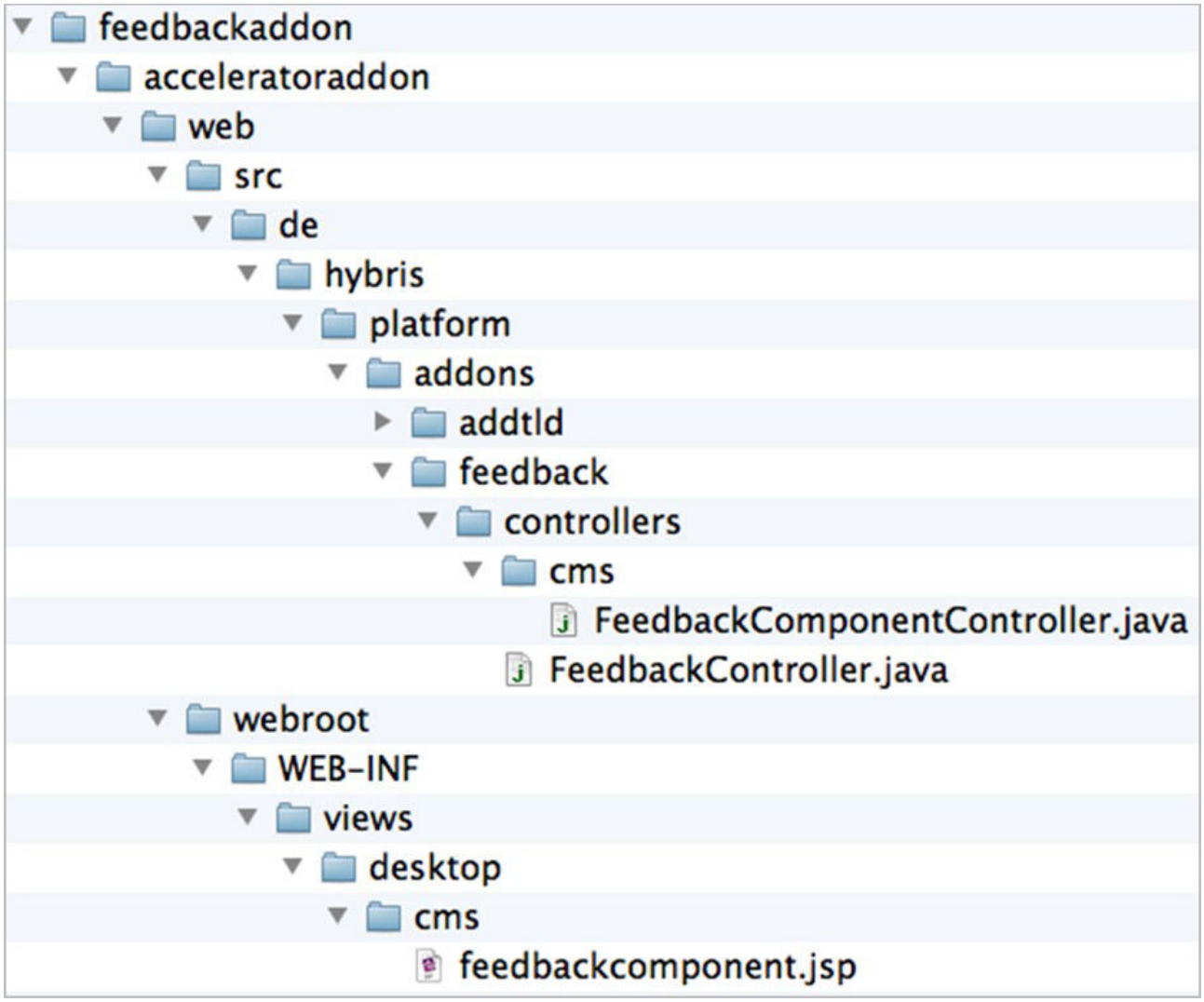
The benefits of the AddOn Concept approach are as follows:

- 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.

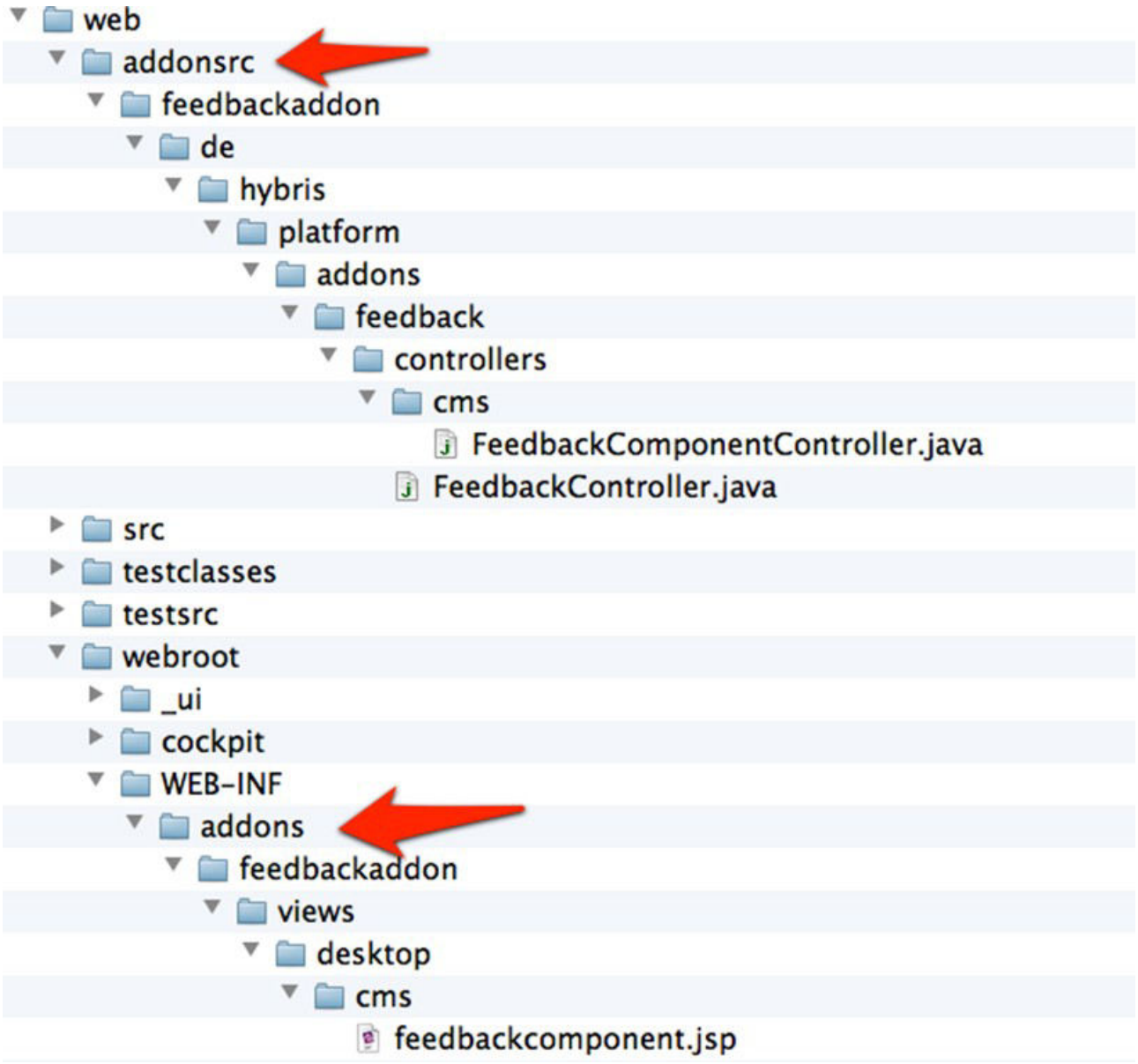
The AddOn Approach

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.

Continuing with this example, the AddOn would have an additional folder called **acceleratoraddon**, and the structure of the folders within it mirrors the structure of the folders containing the front-end components of a regular extension, as follows:



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 the imported content:



Installing an AddOn for a Specific Storefront

The addoninstall tool allows you to configure an AddOn for a storefront. It also adds the AddOn to the extensioninfo.xml file of the storefront, and generates the relevant project.properties file for the AddOn

Format of the addoninstall command
`ant addoninstall -Daddonnames="AddOnName1,AddOnName2" -DaddonStorefront.<storefrontTemplateName>="Storefront1,Storefront2"`

Parameters	Notes
addonnames	This parameter is mandatory. If more than one AddOn name is specified, the names must be separated by commas. If you do not include the **`addonnames`** parameter when you run the **`addoninstall`** command, you are asked to provide it.
addonStorefront.*storefrontTemplateName*	This parameter is used to indicate which storefronts you are installing the AddOn for. If more than one storefront is specified, the storefront names must be separated by commas. The *storefrontTemplateName* variable indicates the storefront template you wish to use for storefront generation. By default, the two possible values are **`yacceleratorstorefront`** and **`yb2bacceleratorstorefront`**.

The following is an example of the **ant addoninstall** command, where two AddOns (**NewAddOn1** and **NewAddOn2**) are installed for **B2CStorefront1** and **B2CStorefront2**, as well as for **B2BStorefront1** and **B2BStorefront2**:

Example of the addoninstall command

```
ant addoninstall -Daddonnames="NewAddOn1,NewAddOn2"
-DaddonStorefront.yacceleratorstorefront="B2CStorefront1,B2CSto
```

Running the ant addoninstall Command

The following procedure describes how to run the ant addoninstall command.

- Before you run the ant addoninstall command, make sure the addonsupport extension is listed in localextensions.xml file, as follows:
- Also, ensure that the AddOn and storefront extension that you want to install are listed in localextensions.xml file.
- Open the command prompt and navigate to the hybris/bin/platform directory.
- Run the addoninstall command. The command has the following format:

```
ant addoninstall -Daddonnames="AddOnName1,AddOnName2"
```

```
-DaddonStorefront.<storefrontTemplateName>="Storefront1,Storefront2"
```

- When addoninstall has finished running successfully, rebuild the hybris system by running ant clean all from the hybris/bin/platform directory.

UnInstalling an AddOn for a Specific Storefront

The addonuninstall tool can be used to remove an AddOn from the storefront.

Format of the addonuninstall command

```
ant addonuninstall -Daddonnames="AddOnName1,AddOnName2"
-DaddonStorefront.<storefrontTemplateName>="Storefront1,Sto
refront2"
```

Note :Same like installing addon Except antinstall with antuninstall

Task2:-Create lycaaddon

1. Navigate to *${HYBRIS_HOME}*/**hybris/bin/platform** and open a command prompt.
2. Run the appropriate command:
   - **setantenv.bat** on Microsoft Windows systems.
   - **. ./setantenv.sh** on Unix-related systems (such as Linux or Mac OS).
   Do not close the command prompt window.
3. In the command prompt, navigate to *$ {HYBRIS_HOME}*/**hybris/bin/platform/extgen** .
4. Run the following command to create a new extension named **myextension**. The new extension is created under *${HYBRIS_HOME}*/**hybris/bin/custom** .

```
ant -Dinput.template=yaddon
-Dinput.name=lycaaddon
-Dinput.package=com.techouts.lycamobile
```

Navigate to *${HYBRIS_HOME}*/hybris/config/localextensions.xml file and add your new extension.

localextensions.xml
. . . . <extension name="lycaaddon" /> . . . .

Write the your Own logic in addon then it make availbule to StoreFront by installing addon to Strore Front

```
ant addoninstall -Daddonnames="lycaaddon"
-DaddonStorefront.yacceleratorstorefront="lycamobilestorefront"
```

Now Run the **Ant clean all** and refresh the Eclipse now see the bellow path your addon is availbule to lycamobiestorfront
/lycamobilestorefront/web/addonsrc/

Now you change or write the code in addon when it build that automatically reflect in Storefront