**Designing product lines and deriving new products with VariaMos web from scratch (using Fragment-oriented programming)**

August, 2019

VariaMos team

**Fragment-oriented programming.**

Fragment-oriented programming (FragOP) is a framework used to design, implement and reuse domain components in the context of an SPL (Software Product line).

FragOP is a mix between compositional and annotative approaches.

FragOP is based on:

* Domain components - Customization points
* Fragmentation points - Customization files
* Fragments

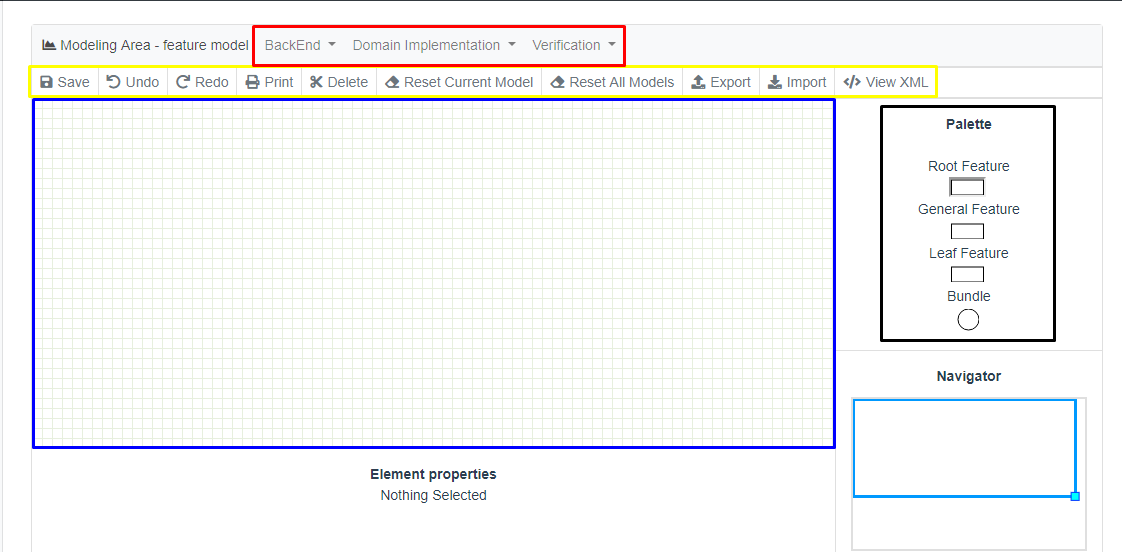
**Understanding the modeling area.**

*Before starting the design and implementation of a product line, we need to understand the modeling area.*

VariaMos Web supports three different types of models:

* Feature model
* Component model
* Binding model

Before using the modeling area, we need to know what are the parts which compose it, check next image:



In the previous picture we can see five important sections.

* **In black** is the Palette. The palette defines the available elements used to create the current model, in this case, for example, the feature model palette has 4 elements
  + *Root feature* represents the root of the SPL. Commonly the root feature is named as the SPL name.
  + *Abstract features* represents features that are not leaf features (features in the last model level).
  + *Concrete features* are the leaf features that will be connected to the domain components. These are the features that will be operationalized.
  + *Bundle*. An elements that allows to group features.
* **In blue** is the Modeling area. This is where you can actually drop the elements from the palette which are going to compose the **Feature Model** and define the **Functional requirements** of your product line.
* **In yellow** is the toolbar. The toolbar has some functionalities such as import or export the model, with the toolbar we will be saving the model or erasing everything in the modeling area.
* **In red** is the verification and derivation area, this area needs the BackEnd of Variamos web, and it’s used to derive a product from a product line or verify the integrity of its design.

**Stage 0 - Pre-requirements**

*The pre-requirements are the programs you need to create a product line in VariaMos and derive a specific software product. In this case, both Variamos Services and Variamos Web projects are required.*

* Download, install and run VariaMos web **“development”** version.

<https://github.com/SPLA/VARIAMOS-WEB/tree/development>

* Download, install and run VariaMos services.

<https://github.com/SPLA/VARIAMOS-SERVICES>

**Note:** in the next guides there is a detailed explanation on how to set up **VariaMos-Web**, and **VariaMos-Services**.

* VariaMos Services: <https://docs.google.com/document/d/1UtLtJ2XxPDLYxlLr7V4LCaV7m1A5w24V__LdUfMwcfc/edit>
* VariaMos Front-end ***(Remember to install the “development” version):*** <https://docs.google.com/document/d/1gU18SYM6A7M61e54O7L6jpvQpdQAhTZahMaPlWxrejk/edit>

**Stage 1 - Create a new project**

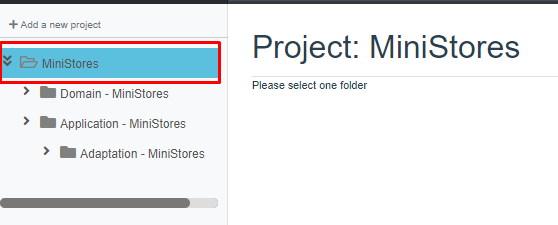
*The project itself is the software product line, and it will be used to derive and create new products.*

Once you’ve started VariaMos-Web in localhost:

* Go to VariaMos home (i.e. <http://localhost:8080/variamosweb/#/>), and click **“Add a new project”**, then write **“MiniStores”** as the project name.



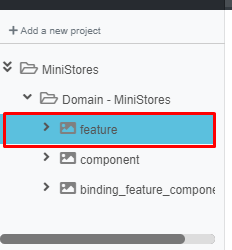
* Now you’ve created a new project, double click on its name and it will detach a tree-like structure which contains **“Domain - PrName”** folder, **“Application - PrName”** folder**,** and the **“Adaptation - Prname”** folder.



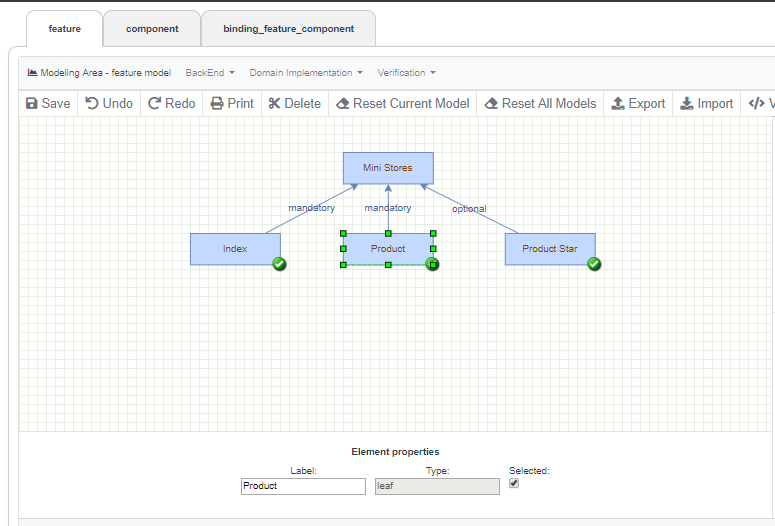
**Stage 2 - Modeling product line requirements**

*To model the product line requirements we are going to use the “Feature model”*

* Make a double click in **“Domain - MiniStores”** and select the “**Feature”** model.

****

* Once we are in the modeling area, we should design the product line: **Mini Stores**



Mini Stores is composed of **4** features:

* **Root Feature:**
  + MiniStores
* **Concrete features:**
  + Index
  + Product
  + Product Star

A **mandatory** relation means that **the existence of that feature in the derivation process is required.**

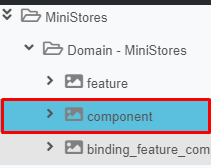
An **optional** relation means that **the existence of that feature in the derivation process is not mandatory** and you can select or deselect them (to be part of a specific product)**.**



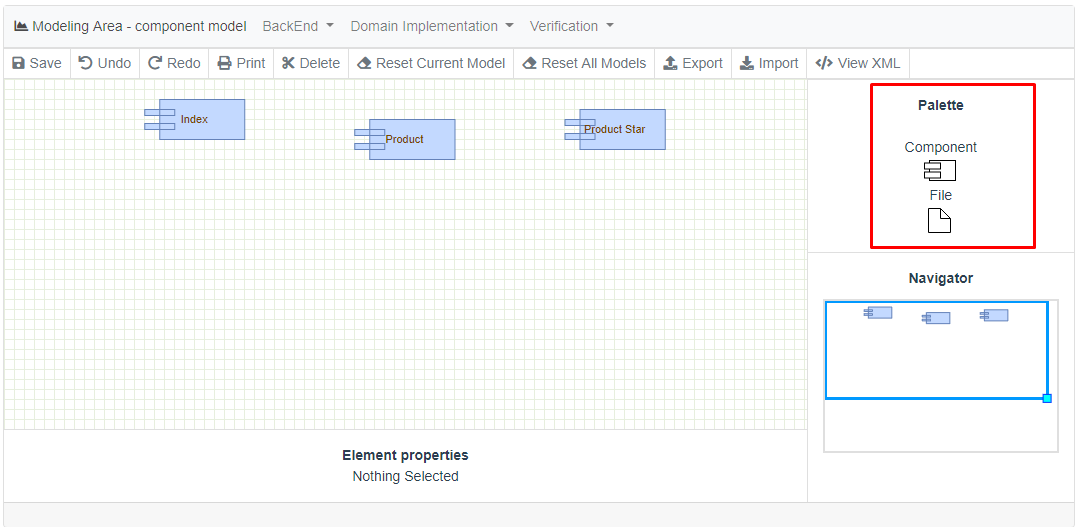
The element properties panel allows you to change the **feature label**,and if the feature is optional, you can **determine if it will be selected** or not to be derived in a specific product.

**Stage 3 - Modeling domain components**

* Make a double click in **“Domain - MiniStores”** and select the **“Component”** model

****

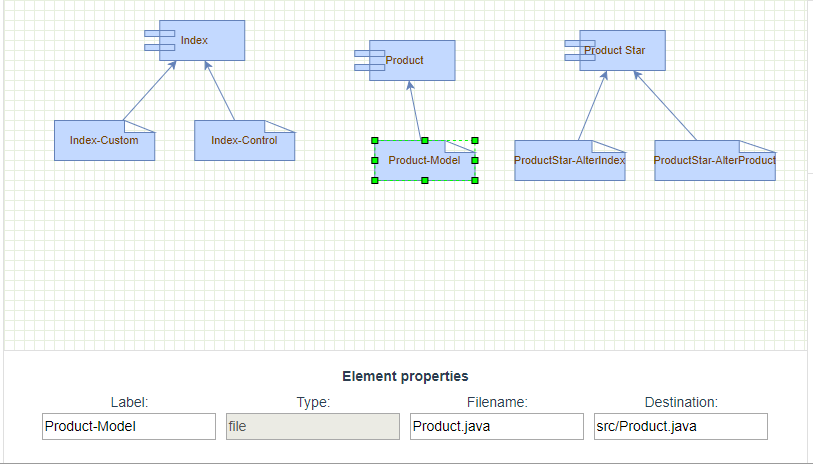
* The component model only have two elements available, components and files. A component represents a physical folder, in which are stored the component files. The files represent the physical files.
* So, for this example, we are going to create three components **“Index”**, **“Product”**, and **“Product Star”**.

**

* Then, we are going to create the following **“Files”** with the use of the information presented in the next table. You have to copy-paste and the next information, to complete the element properties of each file.

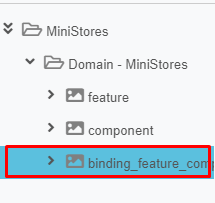
|  |  |  |
| --- | --- | --- |
| **Label** | **Filename** | **Destination** |
| Index-Control | Index.java | src/Index.java |
| Product-Model | Product.java | src/Product.java |
| Index-Custom | customization.json |  |
| ProductStar-AlterIndex | alterIndex.frag |  |
| ProductStar-AlterProduct | alterProduct.frag |  |

* The next image present the final component model.

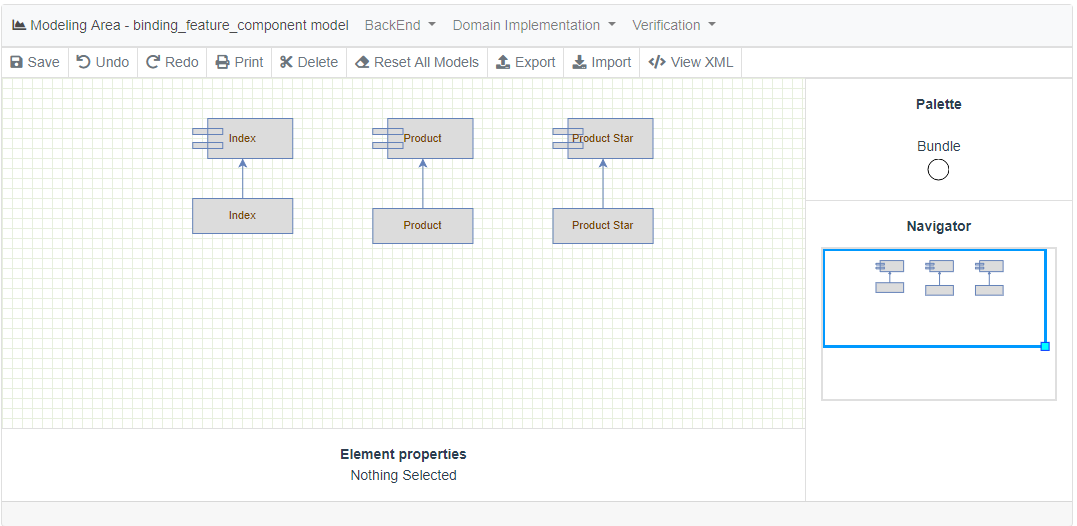


**Stage 4 - Binding domain requirements to domain components**

* Make a double click in **“Domain - MiniStores”** and select the “**binding\_feature\_component”** model

****

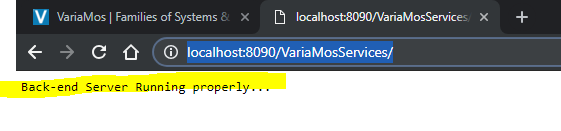
* In this example the binding is very simple. So, all you have to do is connect the respective component to the respective concrete feature.
* Once connected, the final binding model should look like this.



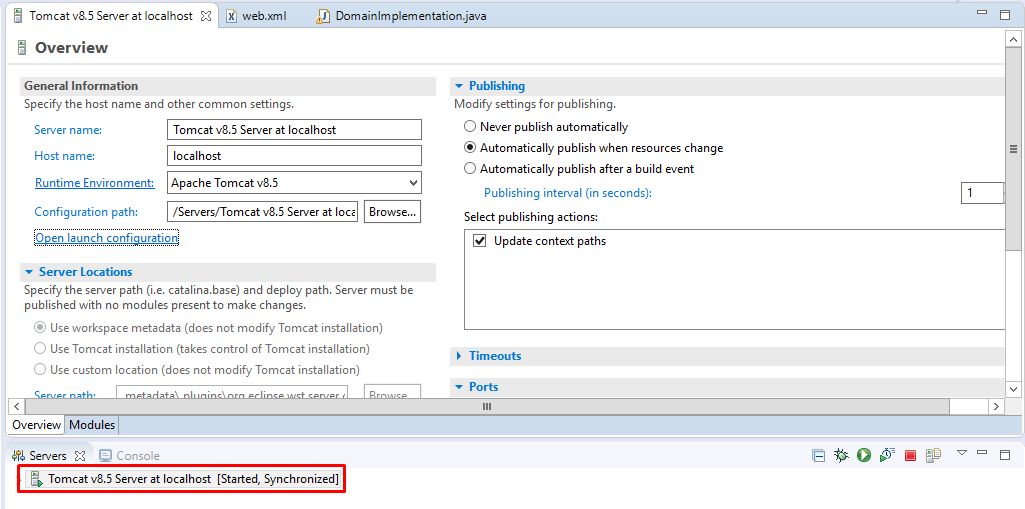
**Stage 5 - Implementing domain components**

*In this step, we are going to define the component pool that contains the real folders and files needed to derive different products.*

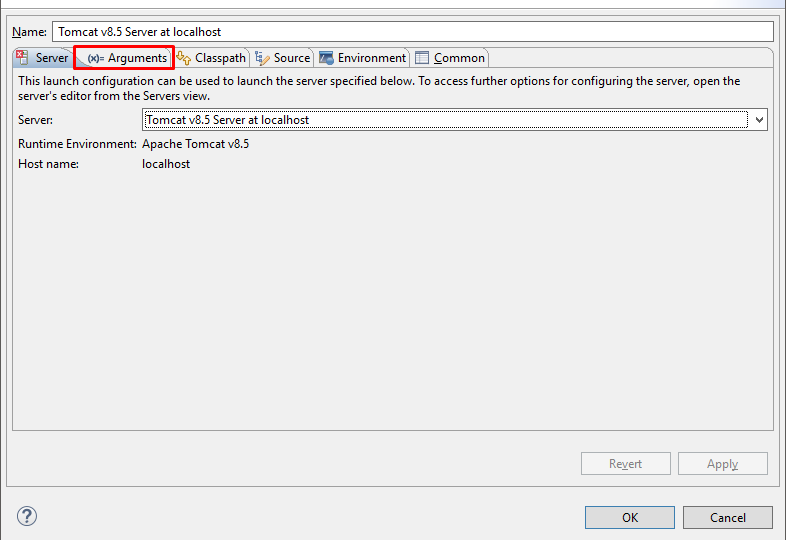
* In order to be able to make a product derivation, we need to Set-Up the BackEnd (the VariaMos Services project). If you don’t know how to do it, you should review this link <https://goo.gl/g55aQW>.
* Once you have the VariaMos Services project running, go to the next address, and it will show you the next message.



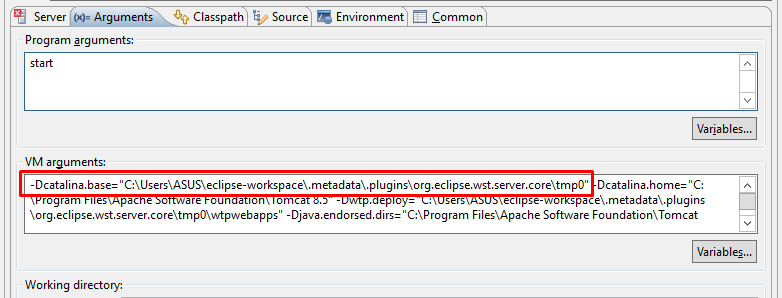
* Now, go to the VariaMos services server path. In eclipse you can find it, making a double clicking in the **server** view



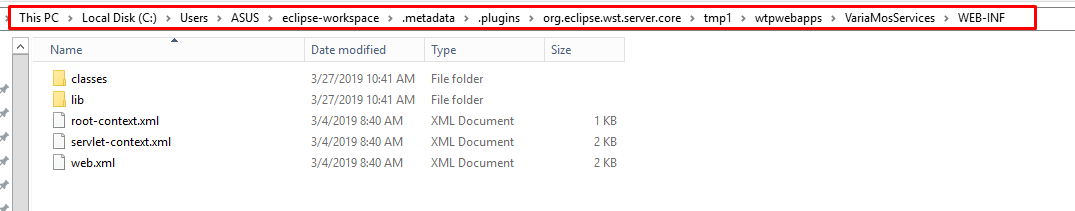
* Then, click on **“Open Launch Configuration”** and “**Arguments”**

****

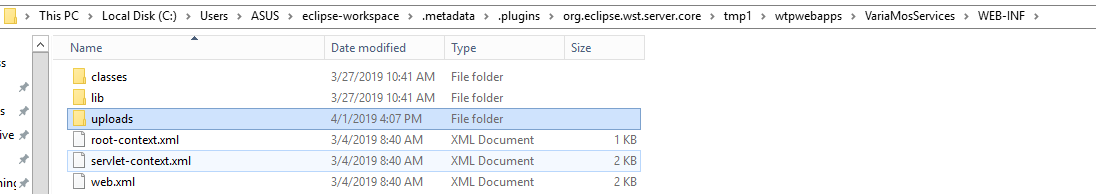
* Now save this **Path,**

****

* Now, you have to search the following directory and open it in a new window: **Workspace**/.metadata/.plugins/org.eclipse.wst.server.core/**tmp\***/wtpwebapps/VariaMosServices/WEB-INF/
* Where **“Workspace”** is your location of the **Eclipse workspace**, and **tmp\*** is a folder called **“tmp”** with a random number.
* The path could be something like this:



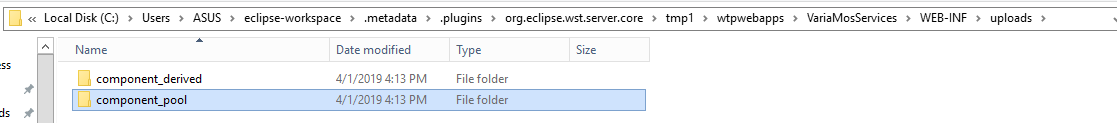
* Then, you have to create a folder there called **“uploads”**.

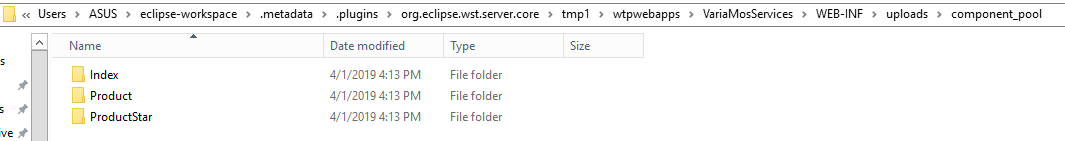
******

* Then, download and copy the folder **“MiniStores”** from this GitHub

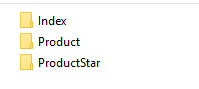
<https://github.com/eechava6/MiniStores> (Clone it or Download it as Zip) and put it inside the “uploads” folder.

* Rename “**MiniStores”** to **“component\_pool”**, andcreate the **“component\_derived”** folder inside **“uploads”**.
* It should look like this:

**

**

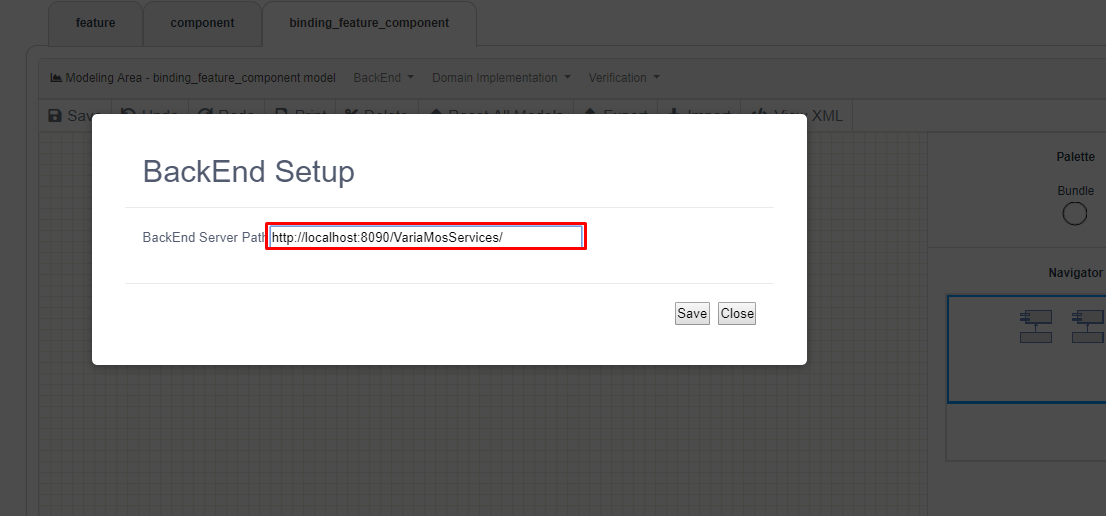
* **MiniStores** contains these 3 folders. You may note that these folders are named exactly the same as the components in the component model.

**

**Stage 6 - VariaMos front-end link**

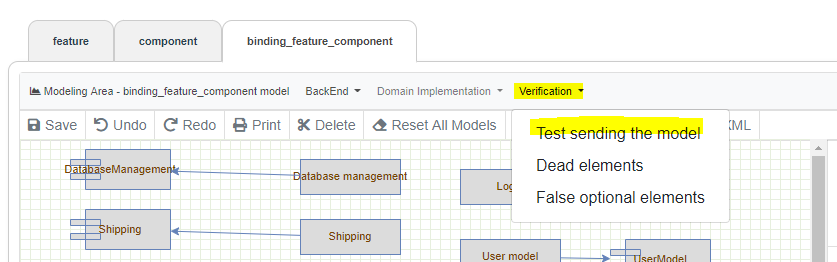
*In this step, we are going to link the VariaMos web (front-end) with the VariaMos Service (back-end) project.*

* To setup and verify if the connection between the two projects are working properly, we are going to click on “**BackEnd**”and then “**BackEnd Setup**”
* Then, write the VariaMos Services main URL (i.e. <http://localhost:8090/VariaMosServices/>) in “**BackEnd Server Path”** and save it.

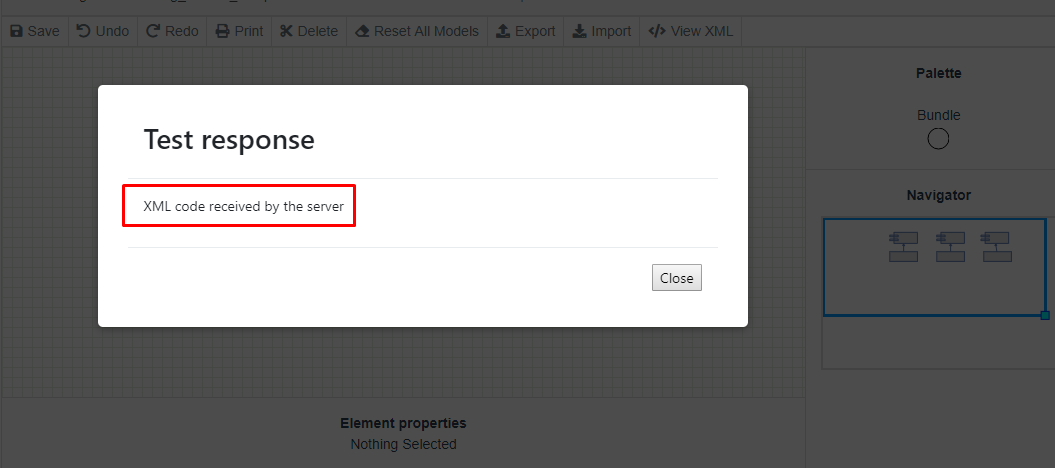


**Note:** sometimes your back-end would be running on a different port to **8090**. So, you have to verify in **Eclipse** where the VariaMos service project is actually running. You can do that clicking the server and verifying its port. If you don't use the right port the communication between the front-end and the back-end projects will fail.

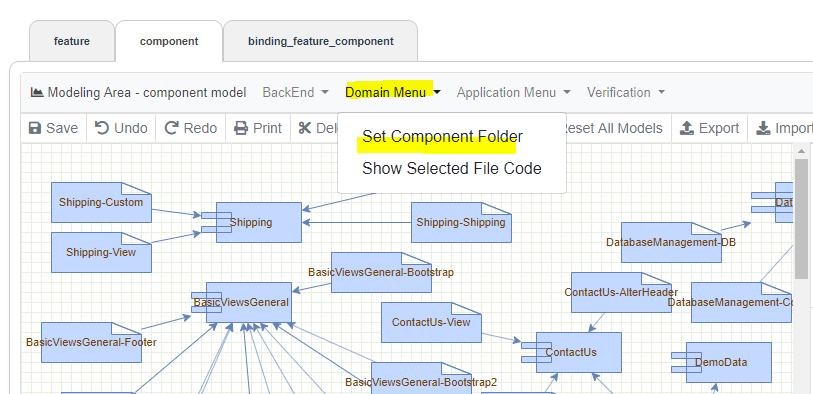
* Once the back-end was configured and run, we are going to click on “**Verification**”,and then “**Test sending the model**” to verify if its working properly.

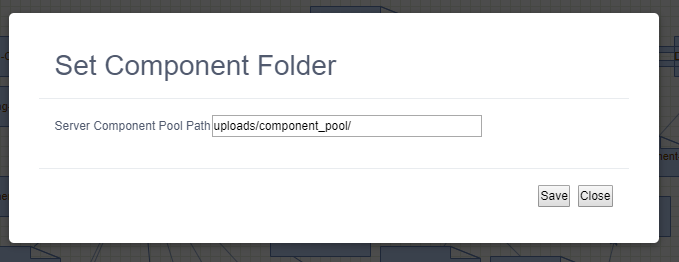


* If the link is ok, you will get the next message:

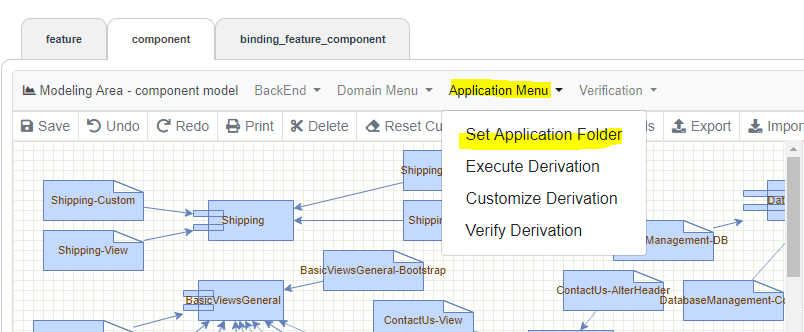


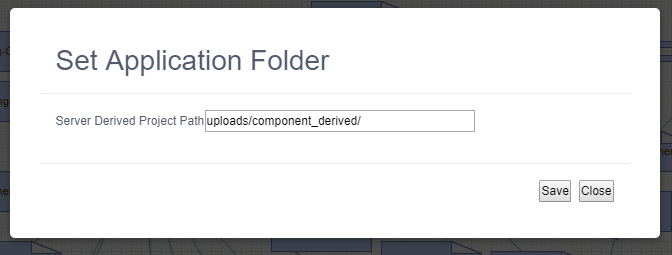
* Otherwise, it means you have a back-end config problem. In that case, you can take a look here to know how to set up properly the VariaMos Services. [**https://goo.gl/g55aQW**](https://goo.gl/g55aQW)
* Now you have to define the Derivation Parameters (Source folder and target folder).
* Click **“Domain Menu” -> “Set Component Folder”**. And put the next content and save it.





* Click **“Application Menu” -> “Set Application Folder”**. And put the next content and save it.

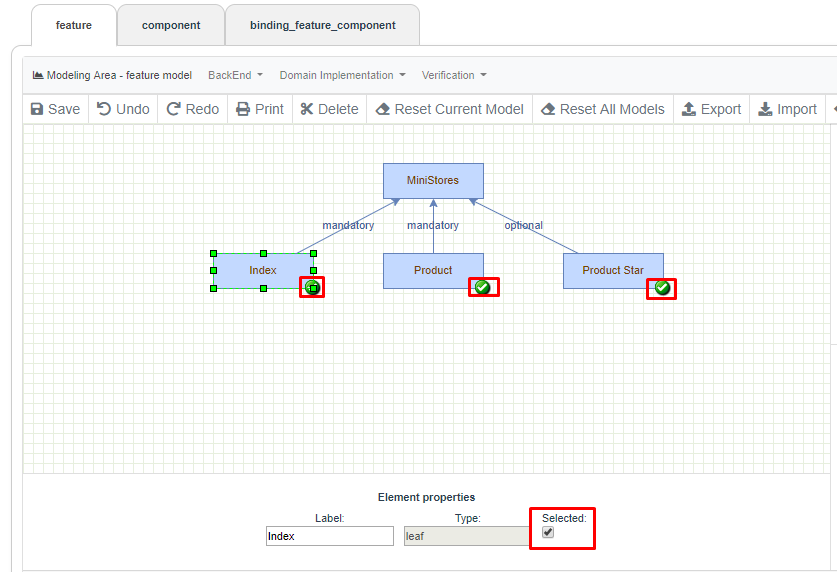




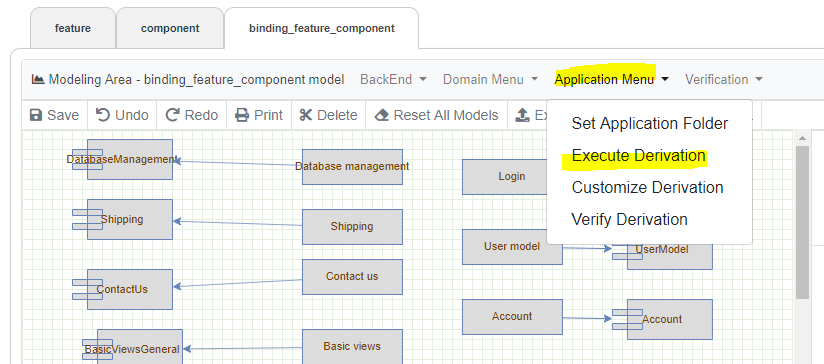
* **Component Pool Path** is where you have the **Source Files** for the derivation.
* **Derived Project Path** is where the derived product will be **generated.**

**Stage 6 - Deriving a product**

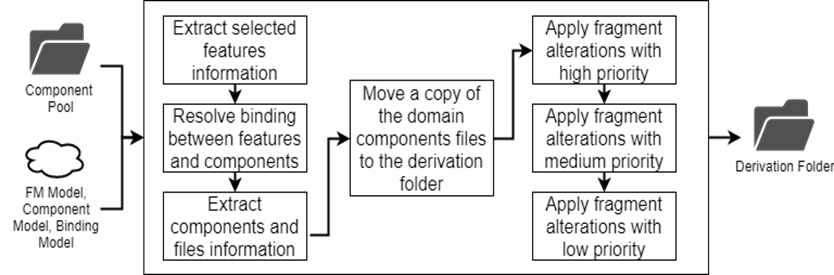
* Open the “**Feature”** model and select the “concrete features” you want to include in the derivation process.

****

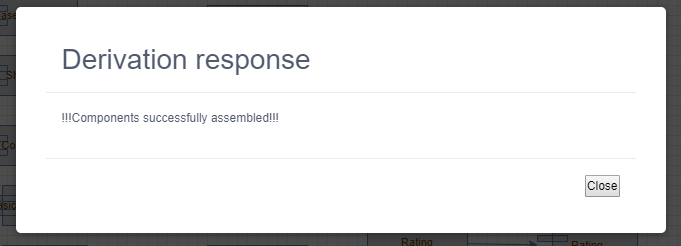
* Click **“Application Menu”** -> **“Execute derivation”**



* Internally, the FragOP process to derive a product is the next one:



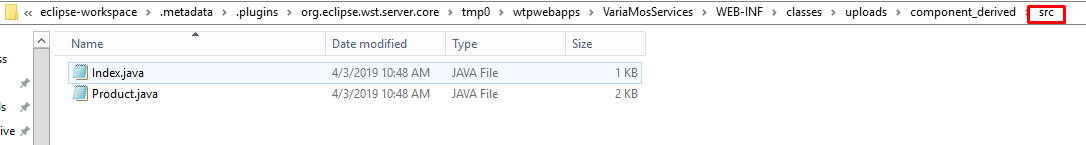
* If everything is ok, it should display the following message:



* Now, you can go to:

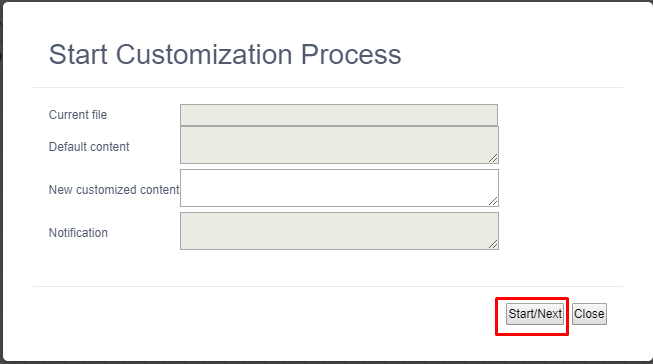
**WorkSpacePath**/.metadata/.plugins/org.eclipse.wst.server.core/**tmp\***/wtpwebapps/VariaMosServices/WEB-INF/classes/uploads/component\_derived/

Then, you should verify that new folders and files were created. If that is the case, the derivation was a success.

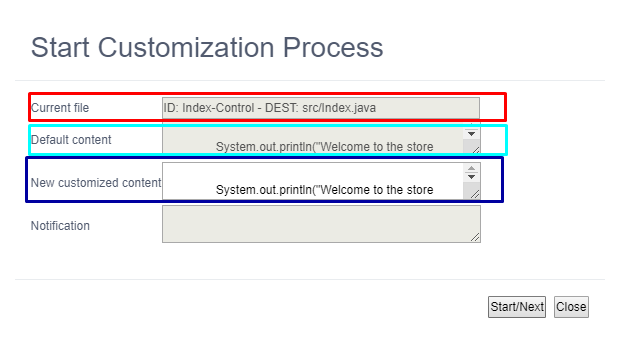


**Stage 7 - Customizing a product**

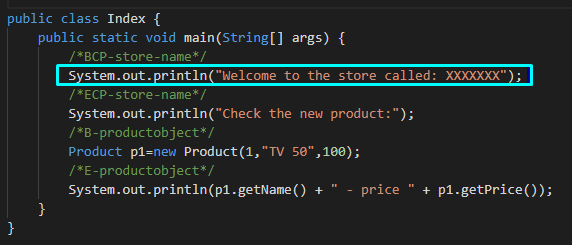
* Click **“Application Menu”** -> “**Customize derivation”**. Then, **“start”** to customize the derived product.



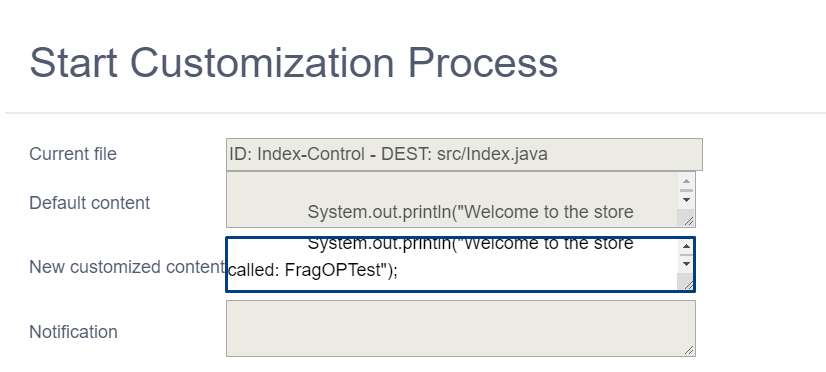
* The customization window has 3 important areas plus an extra area for notifications:}



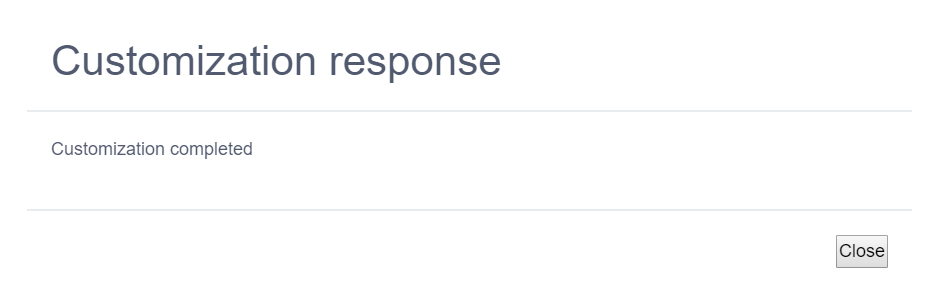
* The Red area presents information about the current File that you are modifying and it shows the destination in which the file is being modified.
* The Light-Blue area is the default content before being customized.



* The Dark-Blue area is the new content that you can customize.

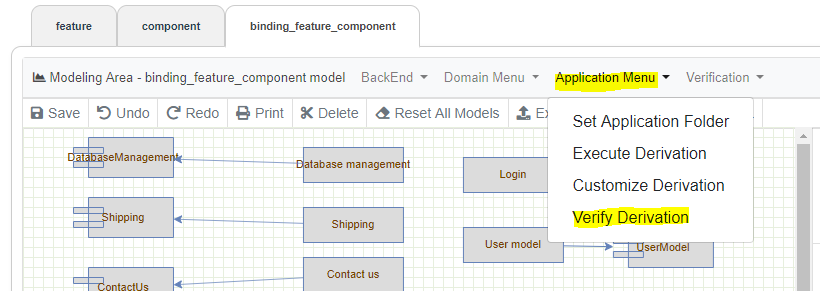


* Once you finish the customization process, you will get the following window telling you that the customization process was completed.

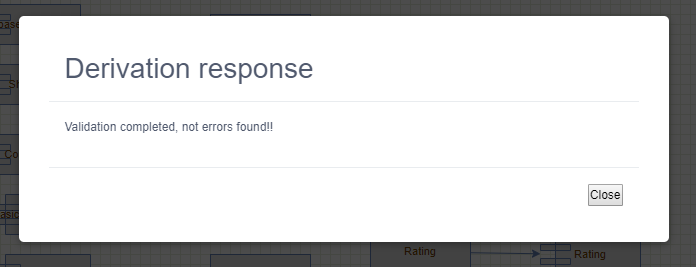


**Stage 8 - Verify derivation**

* In this step, we are going to verify the new product files, through the use of some syntax validators (Java Syntax Checker in this case). To do this, you must click **“Application Menu”** -> **“Verify derivation”.**

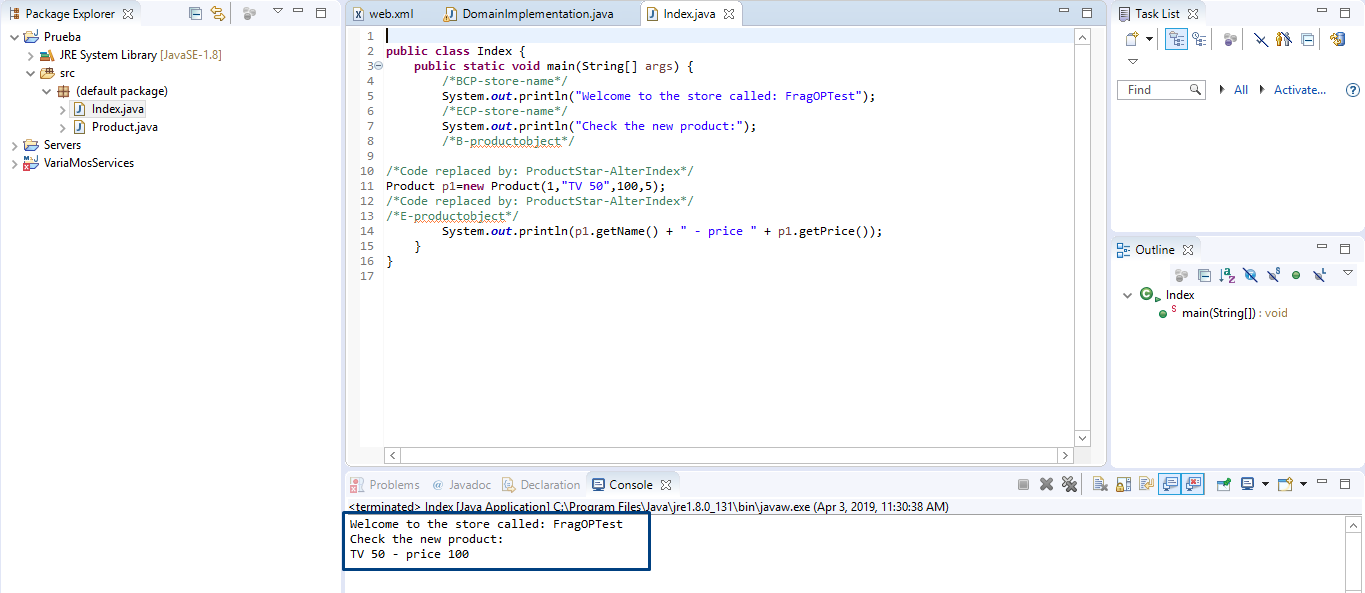


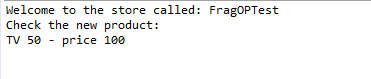
* It could take a while (15-20 seconds). If everything was ok, it should display the following image, which means that the validation was ok, and there were no errors.



**Stage 8 - Testing the product**

* Now, you can use your favorite IDE (i.e. Eclipse). You have to create a new Java project and import the **“component\_derived/src”** folder inside the Eclipse new project (inside the src folder). Then, you can execute it and test it.





* As you can see, the customization and the results were expected. The final product code will have some comments that indicates where some fragments or customizations were executed.

