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

April 2019

The 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

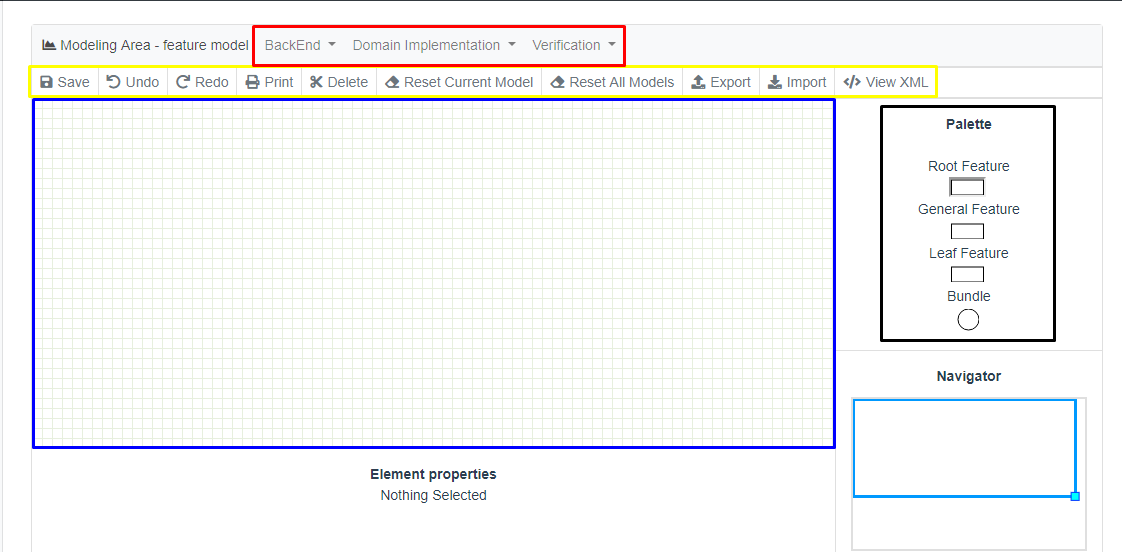
**Understanding the modeling area.**

*Before to start the design and implementation of a product line, we need to understand where and how we should model it.*

VariaMos Web only supports three different types of models(For now):

* Feature model
* Component model
* Binding model

Before using the modeling area, we need to know what are the parts which compose it, look at the following image:



In the picture, we can see 5 important areas.

* **In black** is the Palette, the palette defines what are the elements allowed to create your model, in this case, for example, the palette has 4 elements
  + *Root feature* represents what is the product we are modeling, and from this detaches all the other elements. In a feature model only exists one root feature since you can model only one product at a time.
  + *Abstract feature*.
  + *Concrete feature*.
  + *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 things you need to actually create a product line in Variamos, in this case, both Variamos Services and Variamos Web.*

* 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 following guides there will be a more 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**

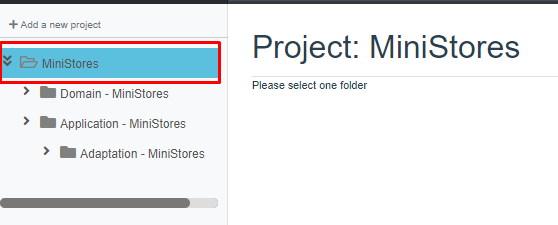
*The project itself is the product line, and where you will derive and create new products.*

Once you’ve started VariaMos-Web in localhost:

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



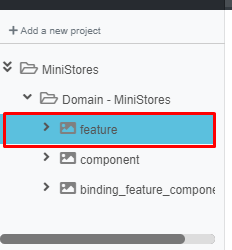
* Now you’ve created a new project, double click on its name and it will detach a tree-like structure which contains **“Domain - PrName”** folder, **“Application - PrName”** folder**,** and nested in the Application folder, **“Adaptation - Prname”** folder.



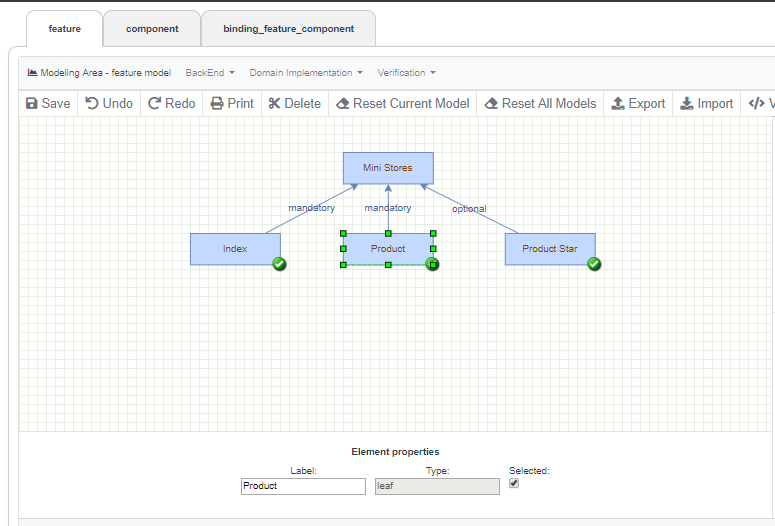
**Stage 2 - Modeling product line requirements**

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

* Double click in **“Domain - MiniStores”** and select in the modeling menu “**Feature”**

****

* Once we are in the modeling area we are going to design the product line: **Mini Stores**



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

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

**Mandatory** relations means that **the existence of that feature in a derivation is mandatory**

**Optional** relations means that **the existence of that feature in the derivation process is not mandatory** and you can select or deselect them**.**

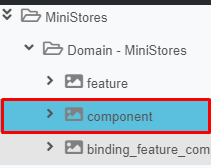


In the element properties panel you can change the **Feature’s label** and if it’s optional, and **determine if it will be selected** or not to be integrated.

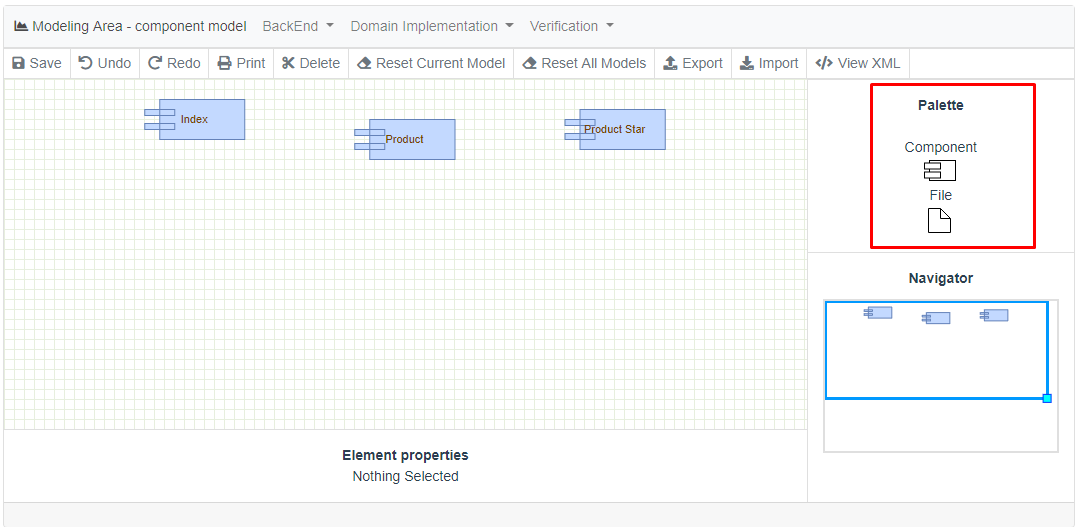
**Stage 3 - Modeling domain components**

*To model the domain components we are going to use the “Component model”*

* Double click in **“Domain - MiniStores”** and select in the modeling menu “**Components”**

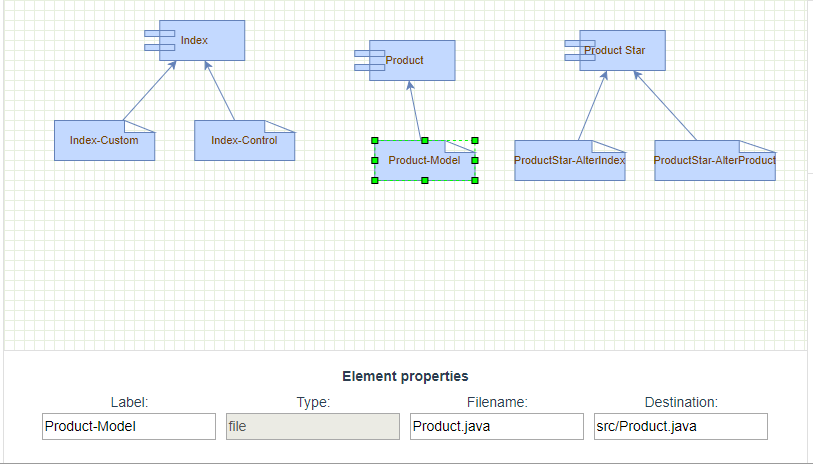
****

* Once in the modeling area, we can see some differences from the feature model, in the components model we only have two elements available, components and files, components represent a functional component of the product while the file is where it actually is physically (In the directory).
* So we are going to create a component **“Index”**, **“Product”** and **“Product Star”** before going in the creation of the files.

**

* As this tutorial is intended to show how to design and derive a product line, we will use an already made small project. In this case, we already know what the files are, but in real life what really happens is that once you have the project created, you have to add the files in the component view.
* As we already know what the files are, we are going to create the following **“Files”** with this table, modifying the specified fields for every column in the **“File”** fields.

|  |  |  |
| --- | --- | --- |
| **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 |  |

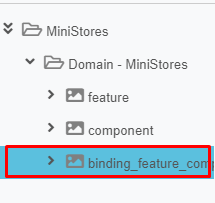


* It should look like this if everything was ok.

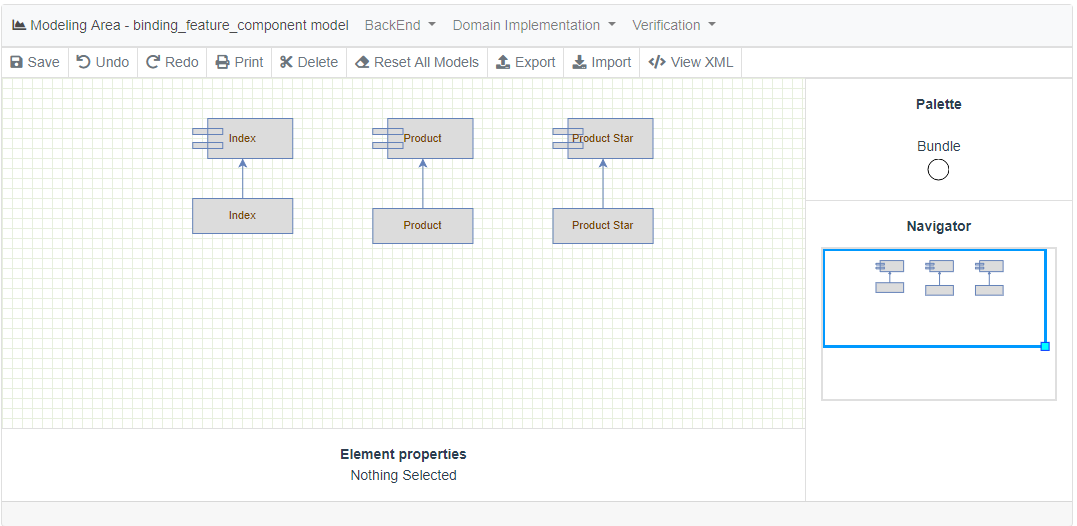
**Stage 4 - Binding domain requirements to domain components**

*To model the binding we will use the “Binding model”*

* Double click in **“Domain - MiniStores”** and select in the modeling menu “**Components”**

****

* For now, binding it’s pretty simple since it only supports one-to-one relationships, so, all you have to do is connect the respective component to the concrete feature.
* Once connected it should look like this.

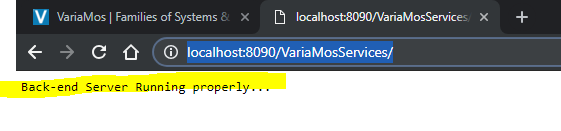


* This is needed since the **binding model** connects the features made in the **feature model** to the components with the physical view in the **component model**.

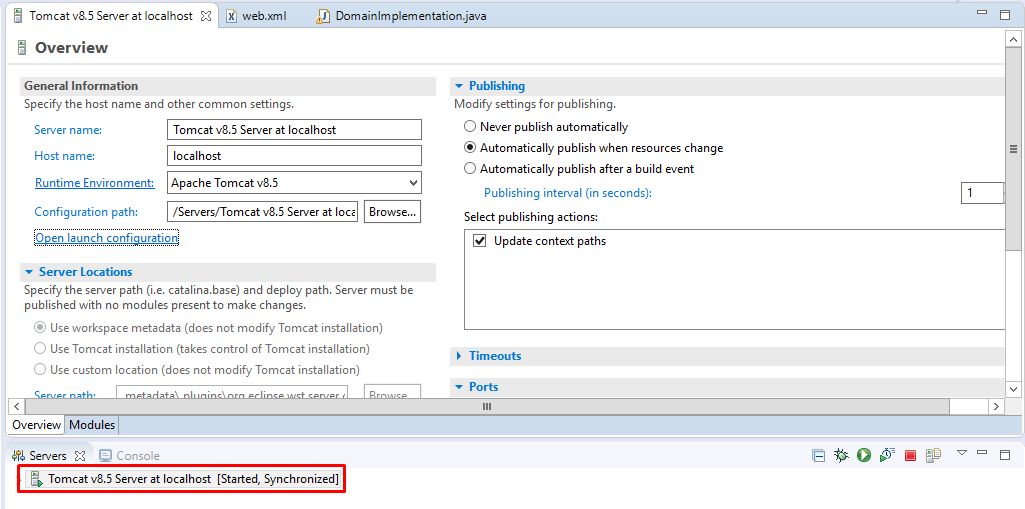
**Stage 5 - Implementing domain components**

*In this step, we are going to define the component pool, from where we will take the files needed to derive the product.*

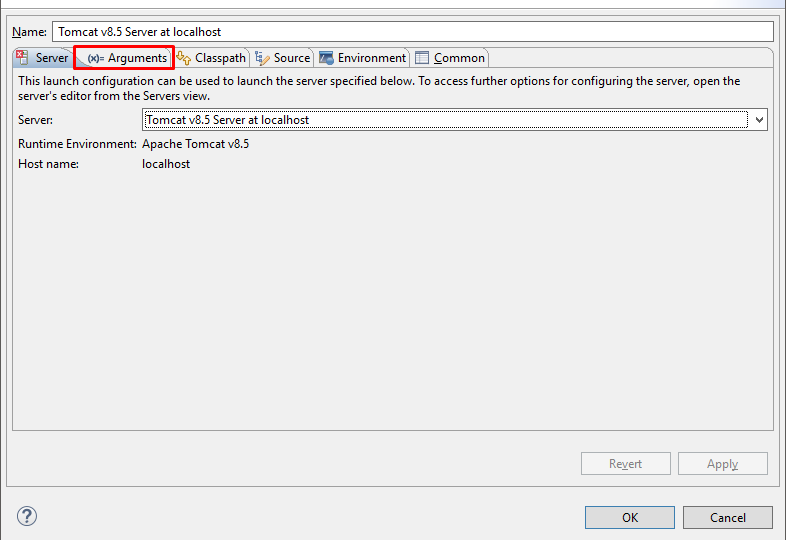
* In order to be able to make the derivation, we need to Set-Up the BackEnd services of VariaMos Web. (**If you don´t know how, review this link** [**https://goo.gl/g55aQW**](https://goo.gl/g55aQW)***)***
* Run the VariaMos services project and if you get the following message it means that VariaMos Services is working properly **if don’t, return to step 1 of this stage.**



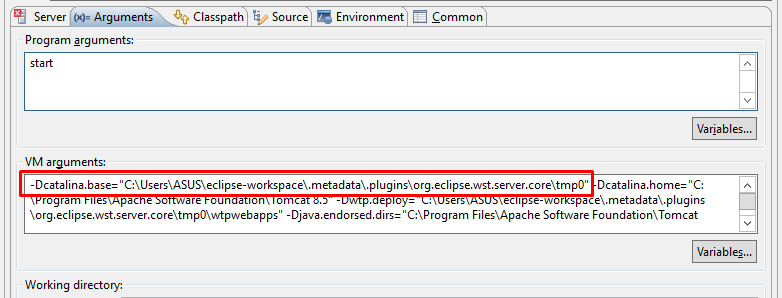
* Go to the VariaMos services server path, in eclipse you can look it double clicking in the **server**



* Click on **“Open Launch Configuration”** and “**Arguments”**

****

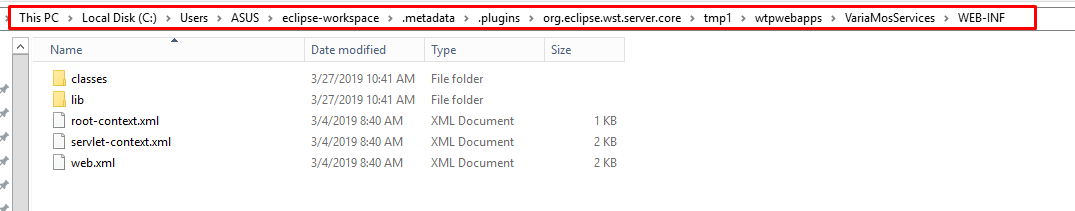
* Now save this **Path,**

****

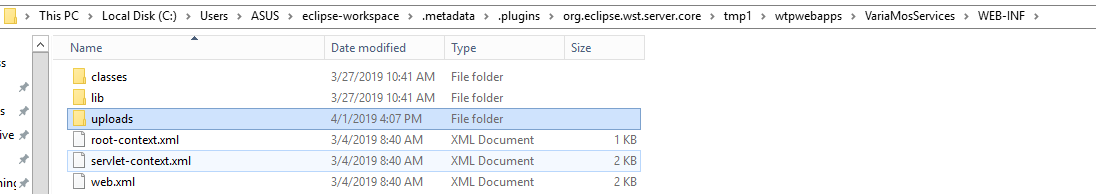
* You have to search for the following directory: **Workspace**/.metadata/.plugins/org.eclipse.wst.server.core/**tmp\***/wtpwebapps/VariaMosServices/WEB-INF/

where **“Workspace”** is the location of **Eclipse workspace**, and **tmp\*** is a folder called **“tmp”** with any number next. (You could have some, select which your server arguments show)

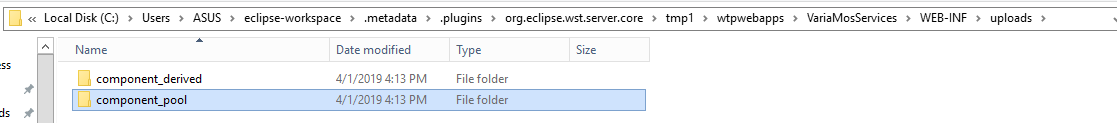
* It could be something like this:

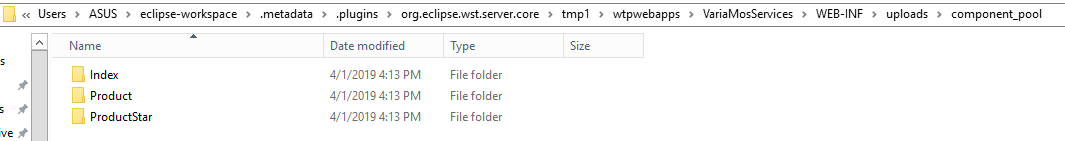


* Create a folder there called **“uploads”.**

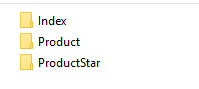
******

* Copy the folder **“MiniStores”** from this GitHub <https://github.com/eechava6/MiniStores> ***(Clone or Download as Zip)*** inside the **“uploads”** folder
* Rename “**MiniStores”** to **“component\_pool”** andcreate **“component\_derived”** folder inside **“uploads”.**
* It should end up like this:

**

**

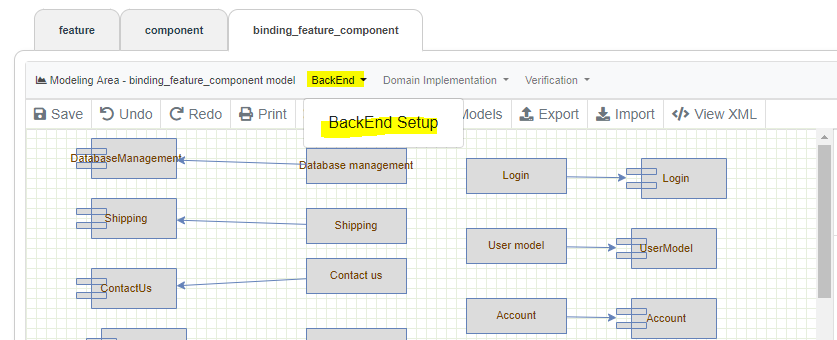
* **MiniStores** contains these 3 folders. (You may note that every file in components model is represented physically in these folders with the exact name and source)

**

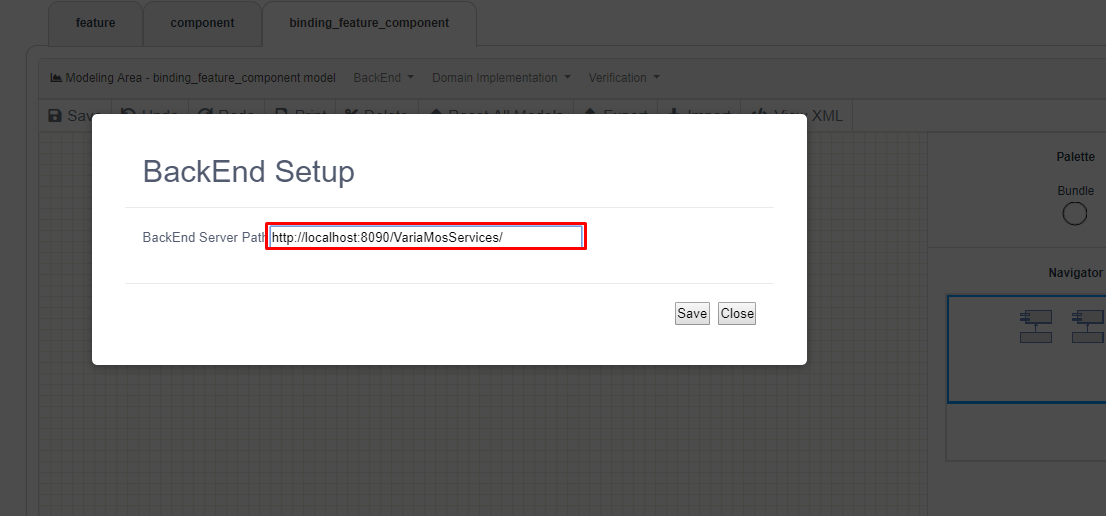
**Stage 6 - VariaMos front-end link**

*In this step, we are going to link and SetUp the front-end with the back-end used, in this case VariaMos Services*

* To Set Up and verify if BackEnd is working properly, we are going to click on “**BackEnd**”and then “**BackEnd Setup**”

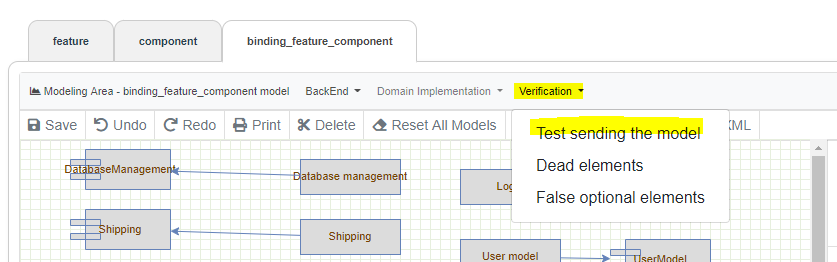


* Write VariaMos Services main path (i.e. <http://localhost:8090/VariaMosServices/>) in “**BackEnd Server Path”** and save it.

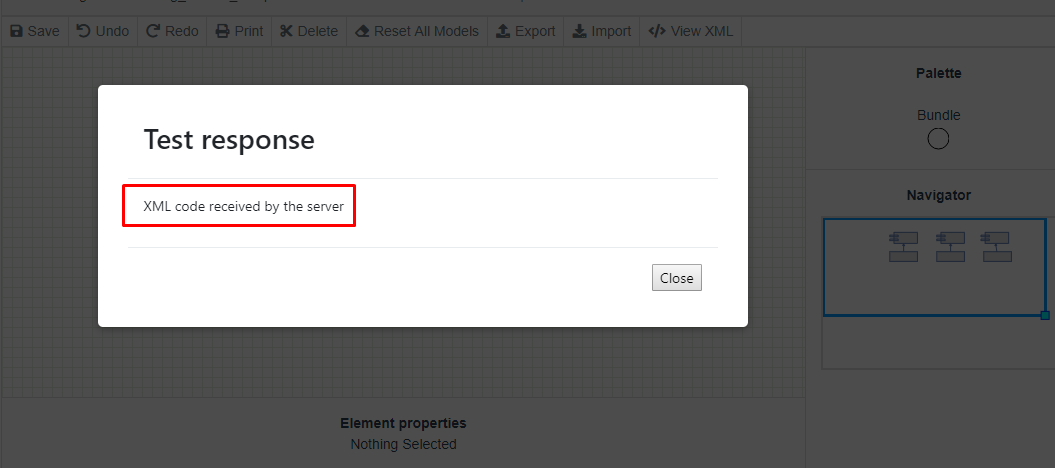


**Note:** sometimes your BackEnd would be running on a port different to **8090**, you have to verify in **Eclipse** where is it actually running, you can do that clicking the server and verifying its ports, since if you don't use the right port the communication between FrontEnd and BackEnd won’t work.

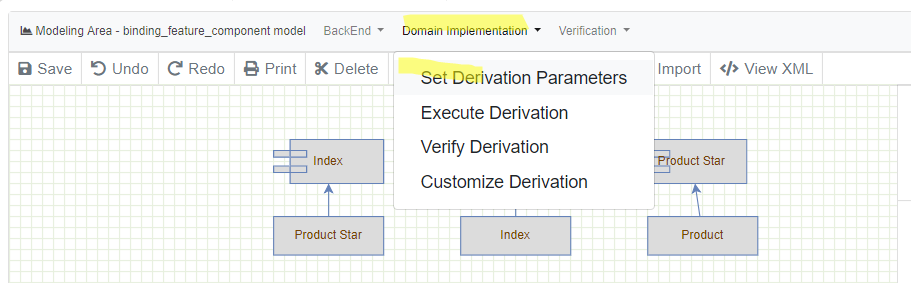
* Once the BackEnd was set up, we are going to click on “**Verification**”and then “**Test sending the model**” to verify if its working properly.

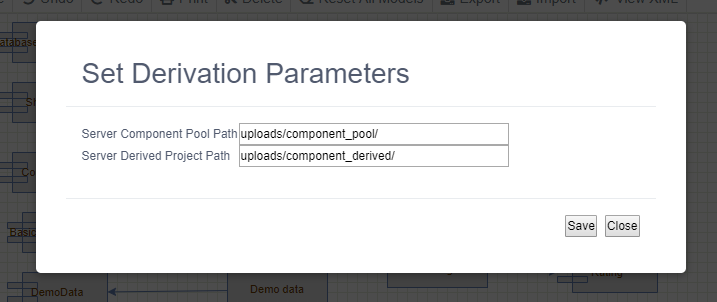


* If it was ok, you will get the following screen:



* Otherwise, it means you have a BackEnd problem, you can take a look here to know how to set up 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 implementation” -> “Set derivation parameters”**. And put the next content and save it.

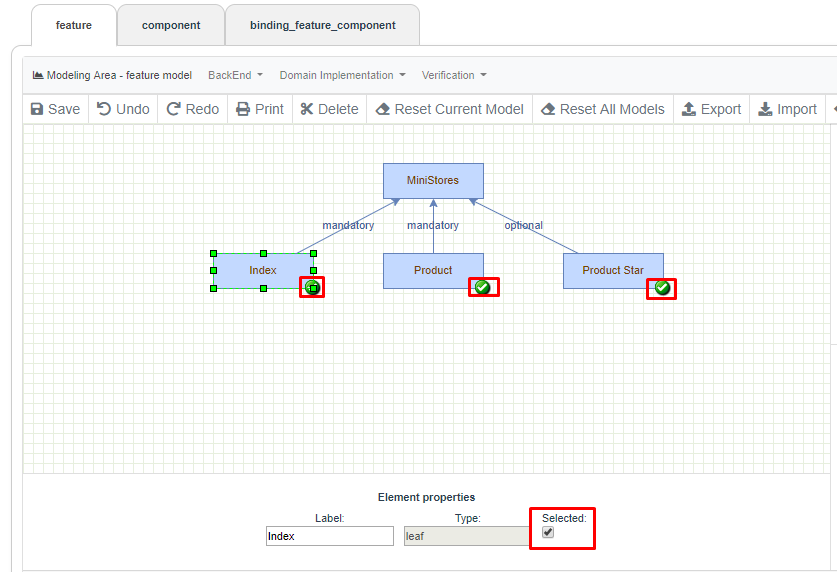




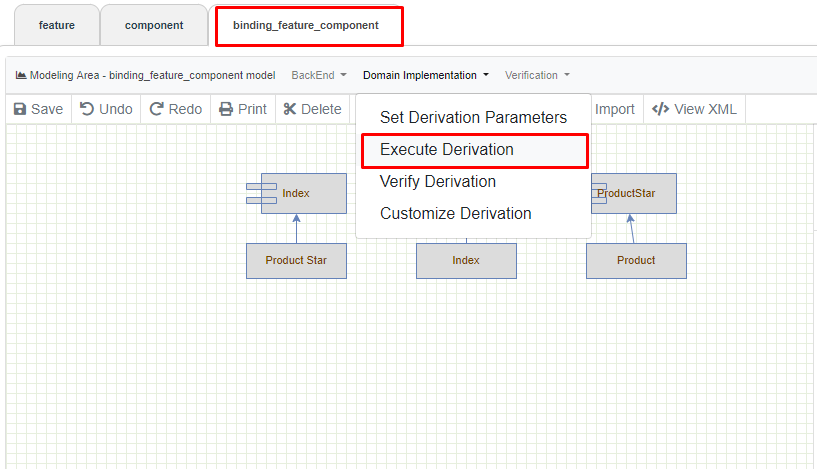
* **Component Pool Path** is where you have the **Source Files** for the derivation. **Derived Project Path** is where the derived product will be **generated.**

**Stage 6 - Deriving a product**

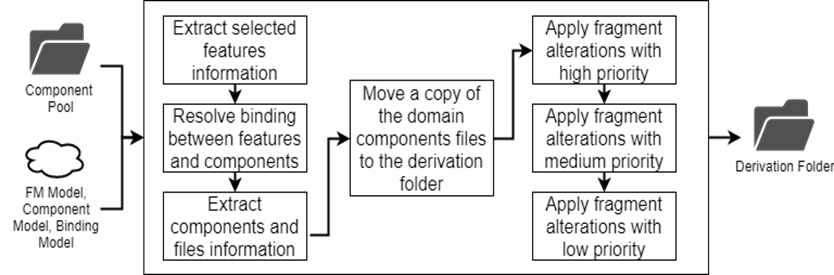
* Open the “**Feature”** model and select the “concrete features” you want to include in the derivation

****

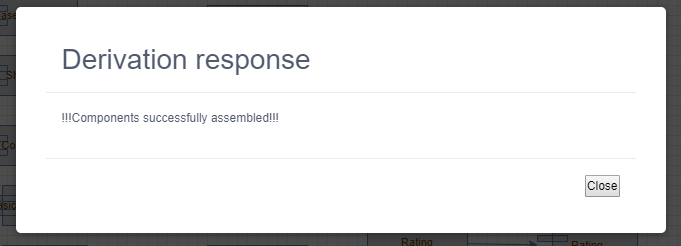
* Click **“Domain implementation”** -> **“Execute derivation”**



* The FragOP process to derive a product is the following:

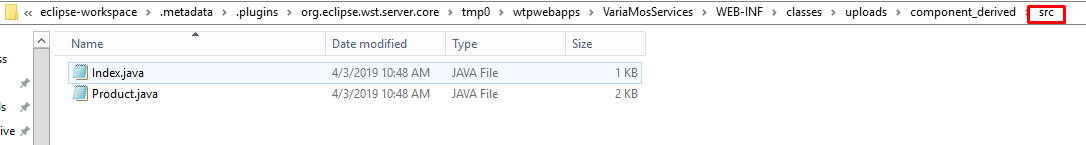


* If everything is ok, it should display the following message:



* Now, you can go to: **WorkSpacePath**/.metadata/.plugins/org.eclipse.wst.server.core/**tmp\***/wtpwebapps/VariaMosServices/WEB-INF/classes/uploads/component\_derived/

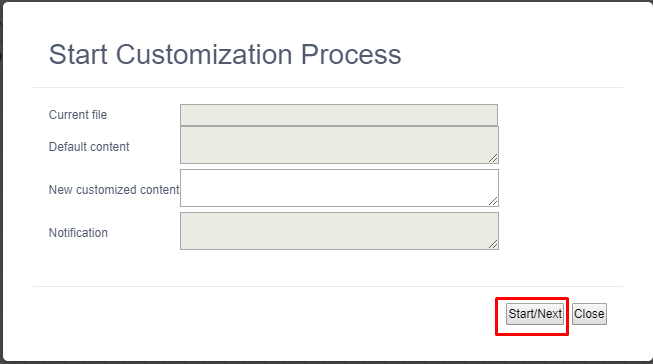
and verify the derived project.

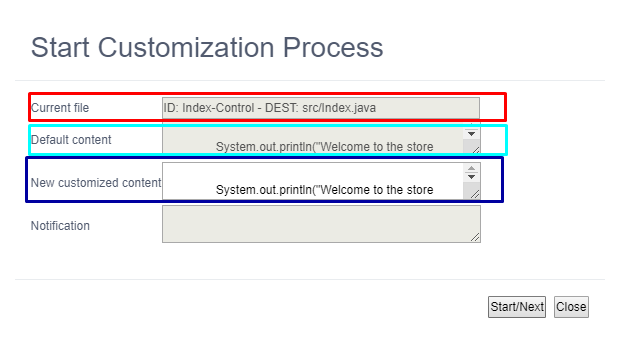


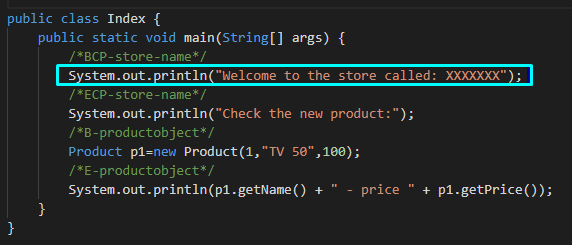
**Stage 7 - Customizing a product**

*In this step, we are going to configure the product itself, since the “master” product has a general content, in this step we are going to make it the product we want to create.*

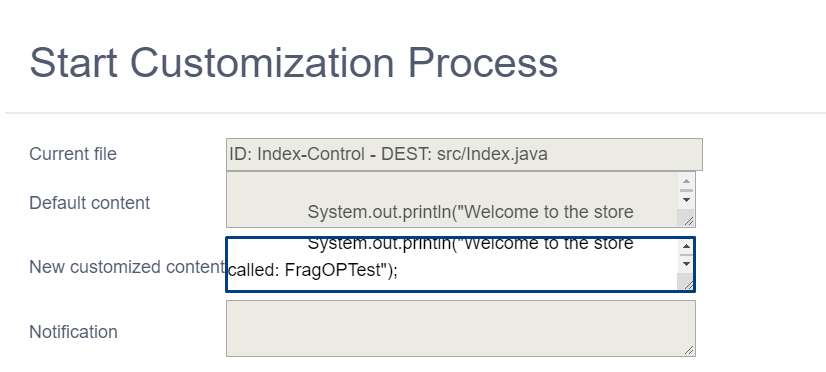
* Click **“Domain implementation”** -> “**Customize derivation”**. Then, **“start”** to customize the derived product.



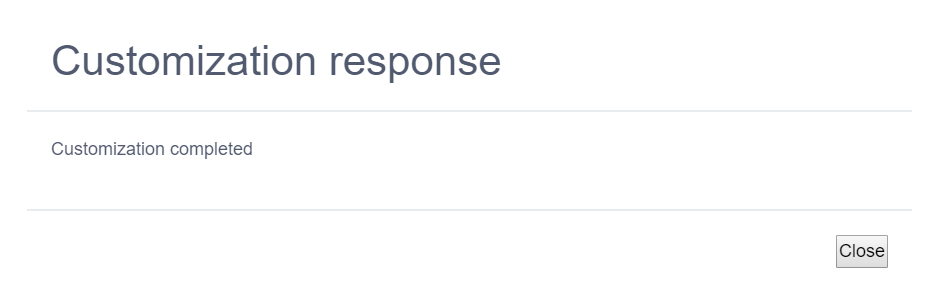
* The customization window has 3 important areas plus an extra area for notifications:
* Where the Red area is the File that you are actually modifying and where will be that File created in the destination; in this case the file is **index.java**.
* The Light-Blue area is the default content of the customization point in this case it



* The Dark-Blue area is the content I want in that **customization point**, so I want my store title to be: **FragOPTest**



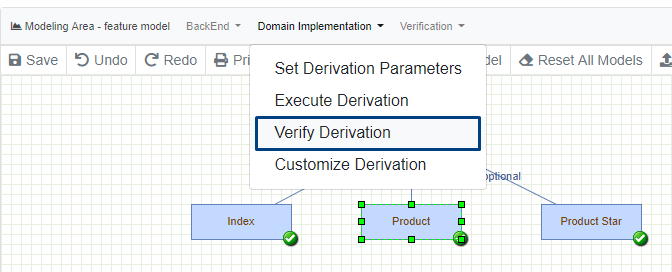
* Once I customized all my components with customization points, I will get the following window telling me that the customization was completed.



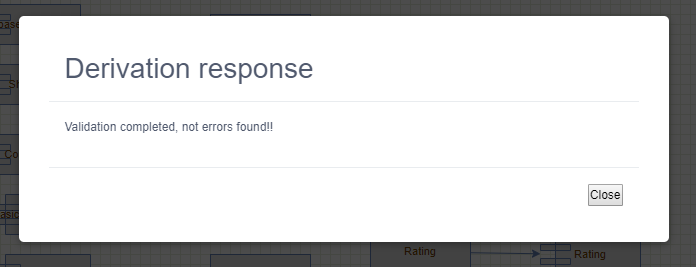
**Stage 8 - Verify derivation**

*In this step, we are going to verify if the product created is ok, in this step we verify the syntax of the result code.*

* In this step, we are going to verify through the use of some compilers depending in the programming language you’re using (Java Syntax Compiler in this case), to do this, Click **“Domain implementation”** -> **“Verify derivation”.**



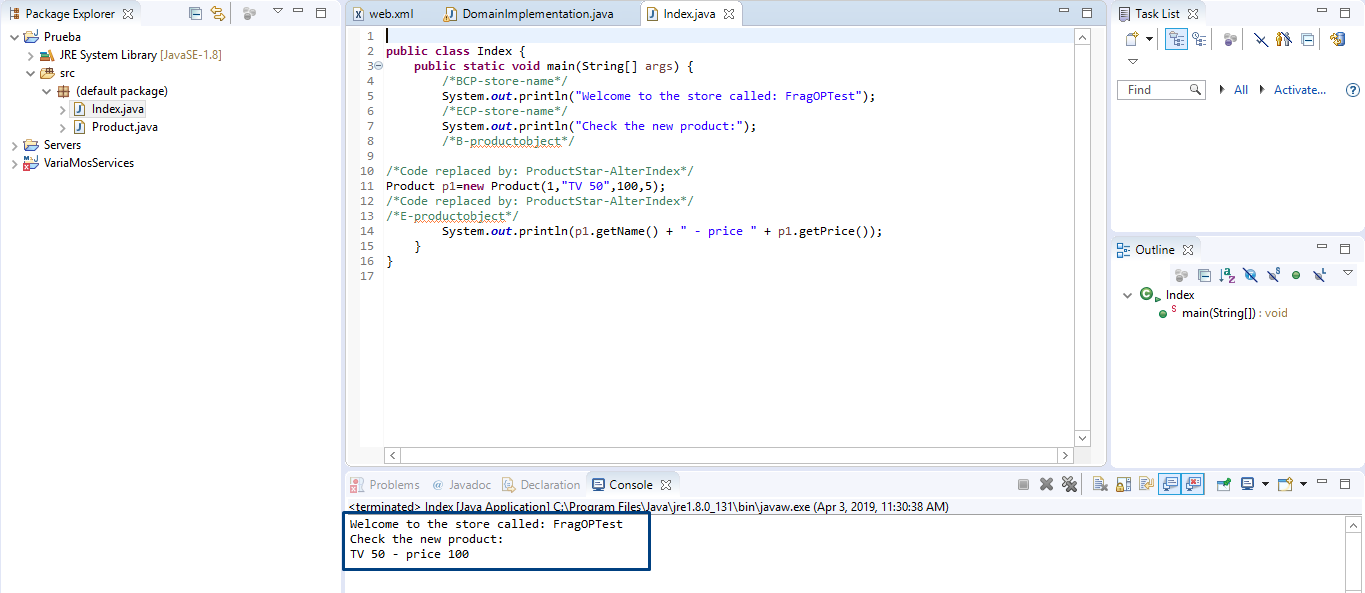
* It could take a while (15-20 seconds) since it is making use of the programming language compiler and it could be an exhaustive task. If everything was ok, it should display the following image, which means that the validation was ok and there were no errors found.

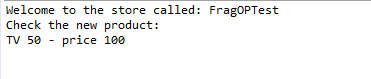


**Stage 8 - Testing the product**

*In this step, we will see how the final product looks and test it.*

* Now, use your favorite IDE (In this case Eclipse) to import the **“Src”** folder with the derived product, execute it and verify the results





* As you can see, the customization and the results were as expected. In the code, you will have these comments indicating where the code was replaced.

