* Plasma Data Studio

TerraMeta Software, Inc.

PlasmaSDO® and PlasmaQuery® are registered

Trademarks of TerraMeta Software, Inc.

# **Introduction**

Plasma is a data store agnostic, object mapping and object query framework written in a Java™ with Maven tools for metadata ingestion and conversion. At its core, Plasma contains a directed graph or digraph model and a set of metadata driven graph traversal algorithms. Data Objects under Plasma form a digraph transparently as a client manipulates the SDO API, graph edges or links being automatically created and used internally to manage associations between Data Object nodes. Data store provider implementations are available for Apache HBase, MAPR-DB (M7) and various relational database vendors.

The Plasma metadata extensions are described in a UML profile which contains UML stereotypes, enumerations and other elements used to enrich UML models for use within the Plasma core as well as third party Data Access Service (DAS) providers. Particular design consideration has been focused on leaving each stereotype granular with only a few tightly related attributes/tags, rather than more monolithic stereotype groupings. This approach lets each stereotype convey far more meaning and maps well to metadata oriented extensions in various target languages, such as Java™ annotations. This granular approach can however have the effect of making UML diagrams more cluttered depending on the presentation settings of the UML diagramming tool.

A single UML logical model fully enriched or annotated with the Plasma UML profile provides enough context specific information to support various technology-specific runtime environments and the generation of numerous context or platform-specific models as well as many other related source-code and other artifacts.

# **Plasma Data Studio**

Plasma Data Studio is an Eclipse based IDE for annotating and integrating UML models with Maven (M2E) based Java projects. The UML editor is derived from Eclipse Papyrus components. See <https://www.eclipse.org/papyrus>. It is centered around a custom Eclipse perspective (Plasma perspective), capability to create projects and diagrams, and a custom UML palette. UML Models are annotated using a custom UML profile and datatypes package, both of which are fully integrated or registered in the IDE. The below screen shot pictures a UML model being edited in-line with a standard Java Eclipse project.

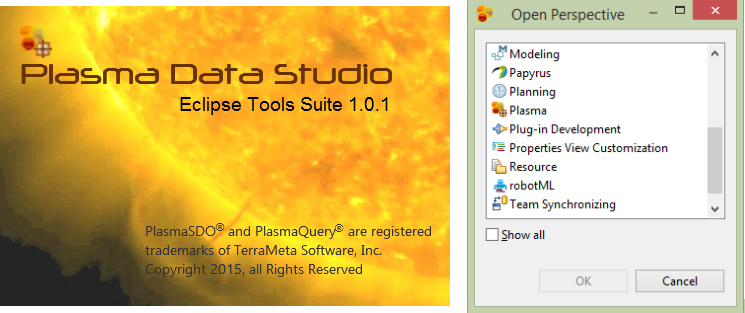


Figure 1 - Plasma Perspective Dialog

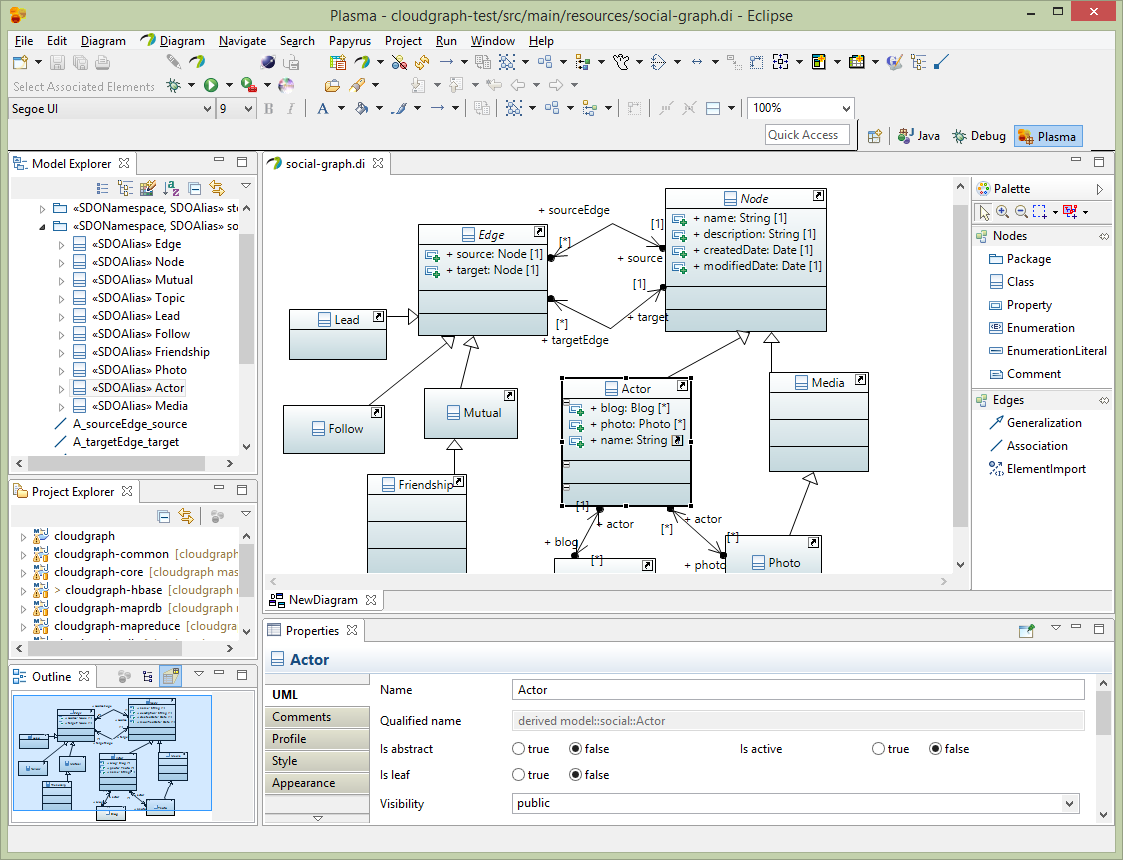
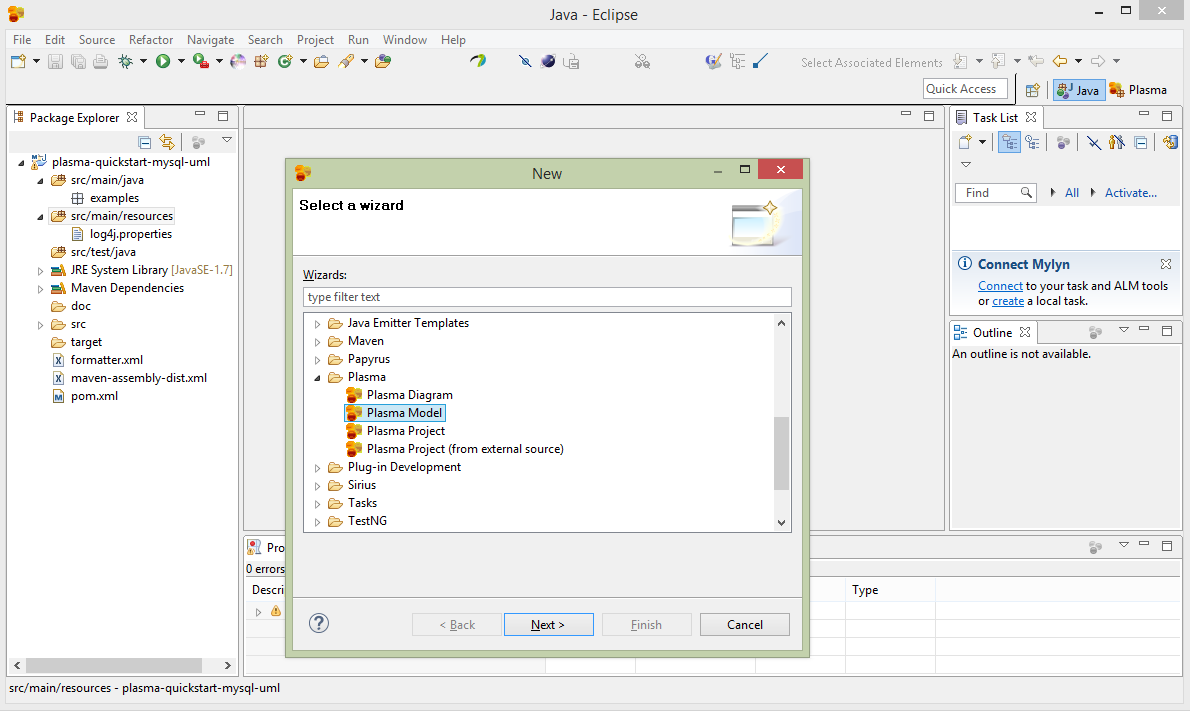


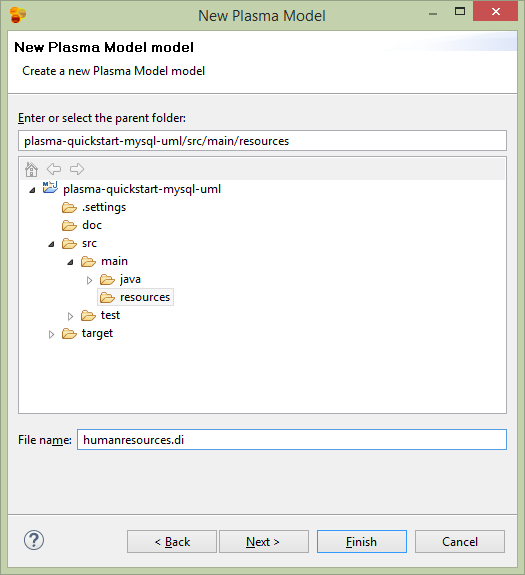
Figure 2 - Plasma Data Studio

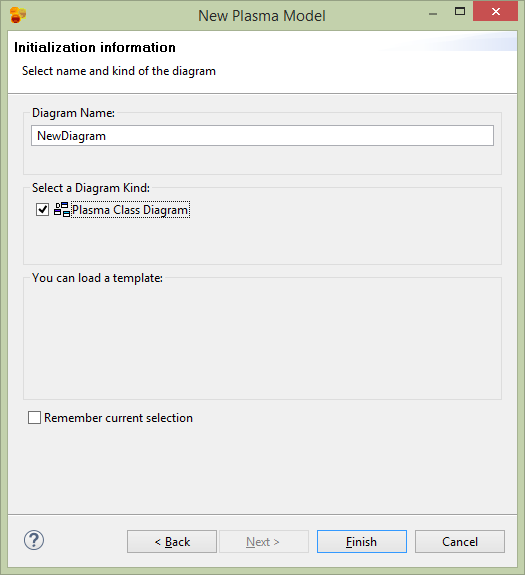
# **Creating a Plasma Model**

From Plasma Data Studio create a new UML model and diagram into the src/main/resources directory of your existing maven project.

1. Right click on the resources directory, then to New->Other->Plasma as below.
2. Save the model (it will have a .di extension, for diagram) in the resources dir of the project.
3. Select Plasma Class Diagram for the diagram type. Then Finish the wizard.







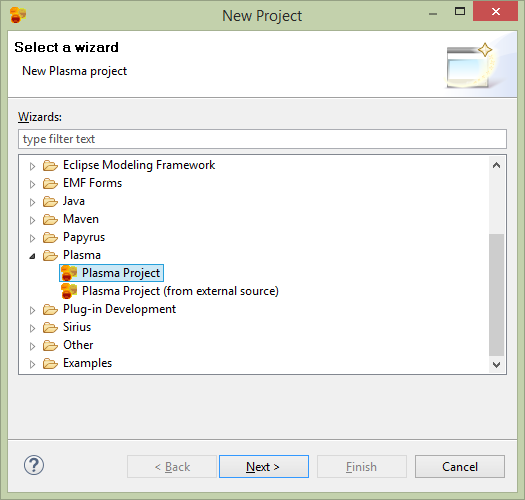
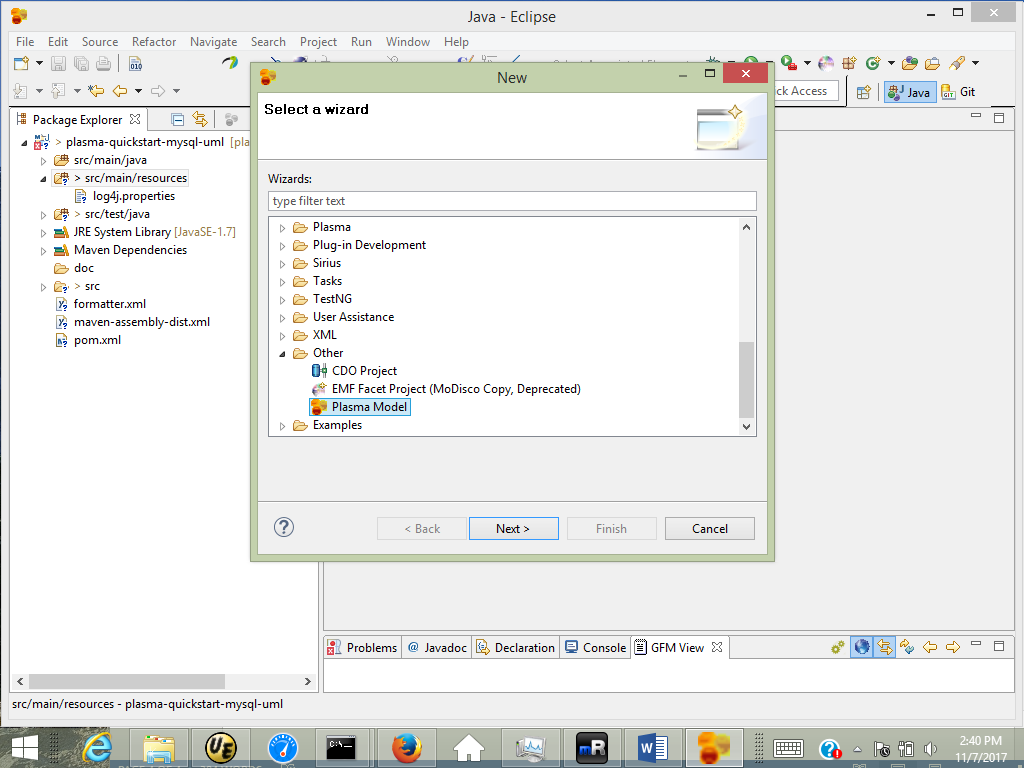


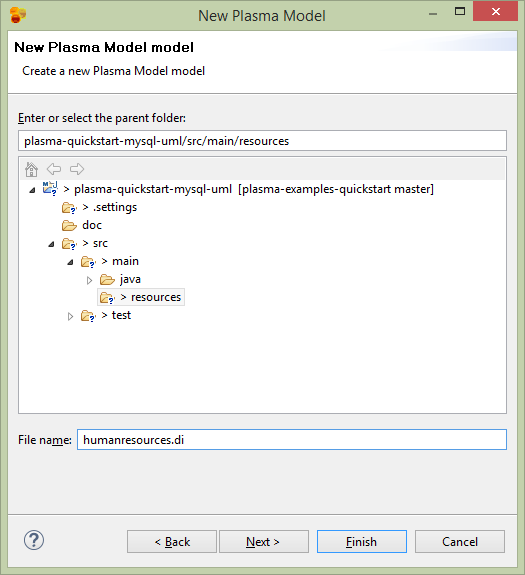
Figure 3 - Create Project Wizard

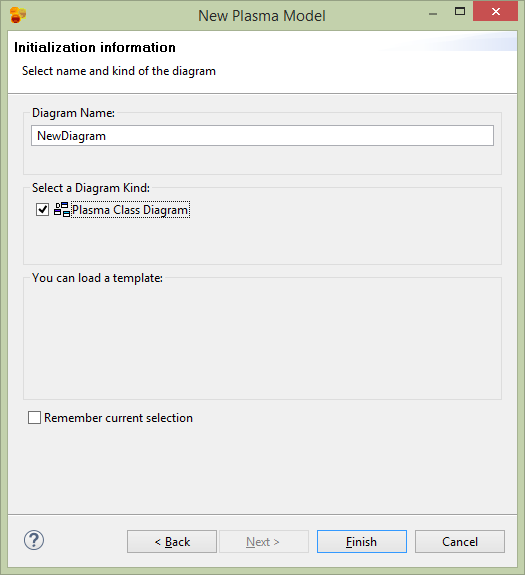
# **Creating a Plasma UML Model**

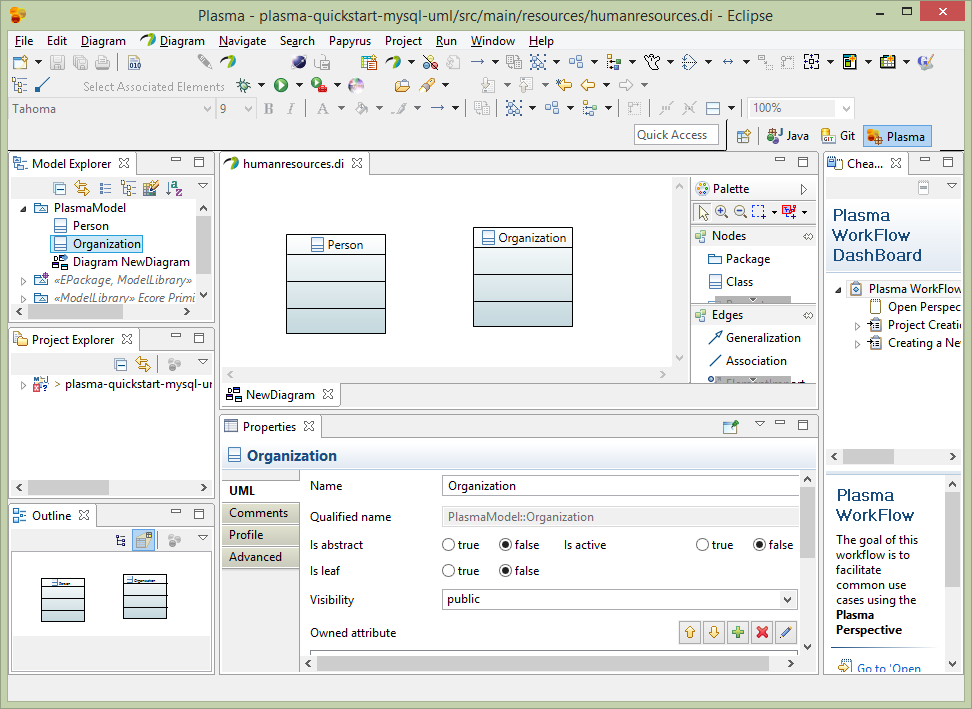
Create a new UML model and diagram into the src/main/resources directory of your existing maven project.

1. Right click on the resources directory, then to New->Other->Plasma as below.
2. Save the model (it will have a .di extension, for diagram) in the resources dir of the project.
3. Select Plasma Class Diagram for the diagram type. Then Finish the wizard.



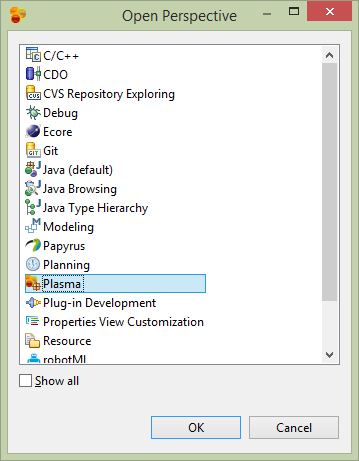


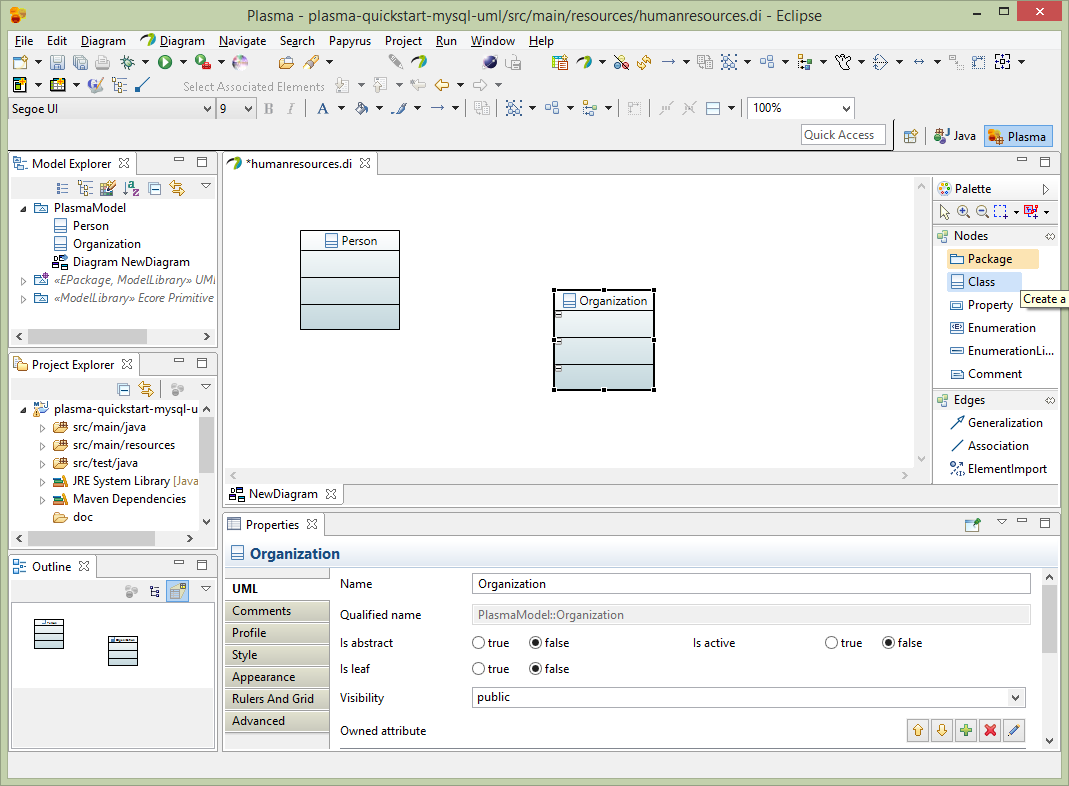




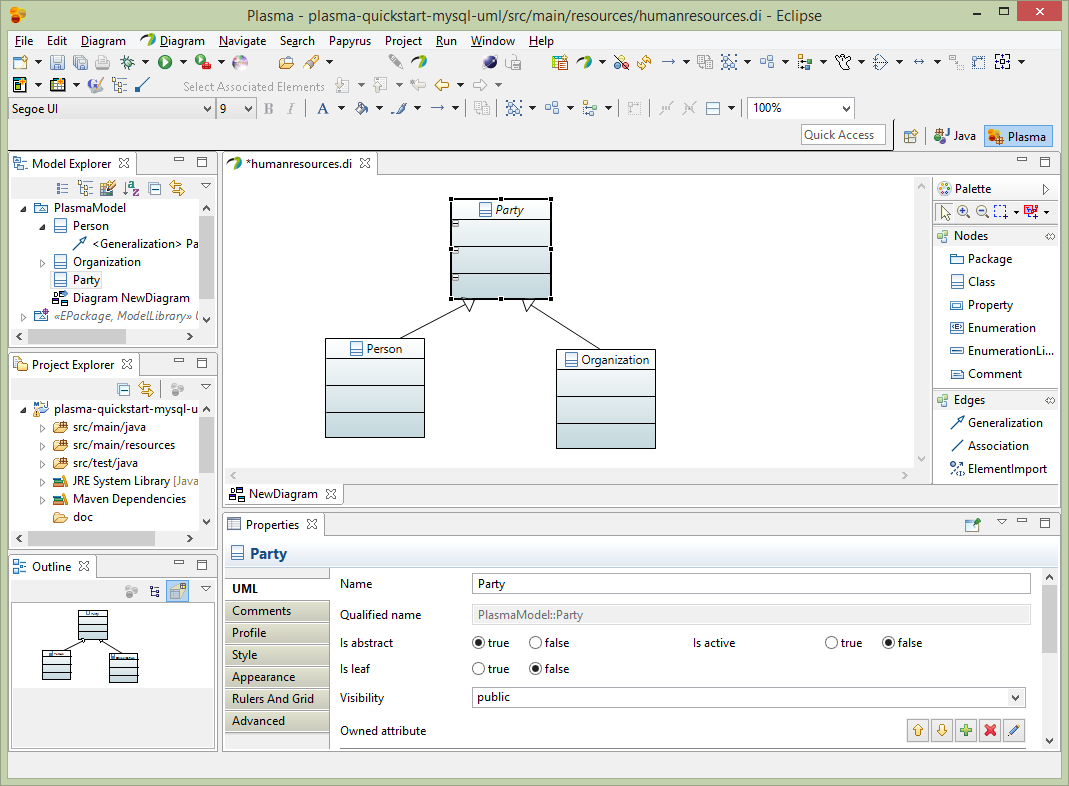
# **Editing a Plasma UML Model**

First open the Plasma Perspective. The new model can be edited and annotated as any Papyrus model. See <https://www.eclipse.org/papyrus/> for detailed documentation on Eclipse Papyrus. Plasma Data Studio however has a custom pallete which restricts the available UML elements to those applicable to Plasma structural models. Double click or drag 2 class elements to the diagram and name one Person and one Organization.

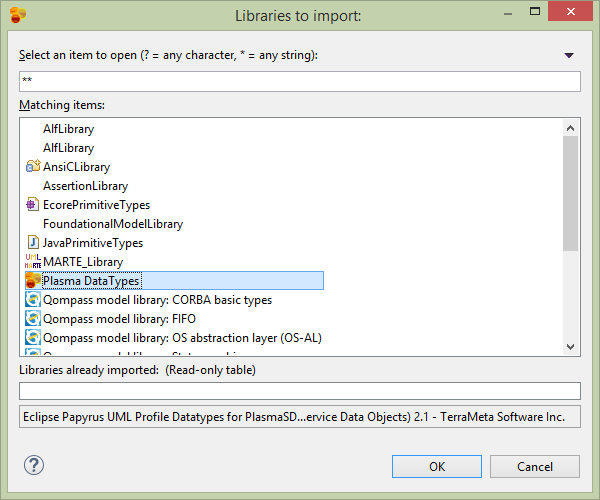




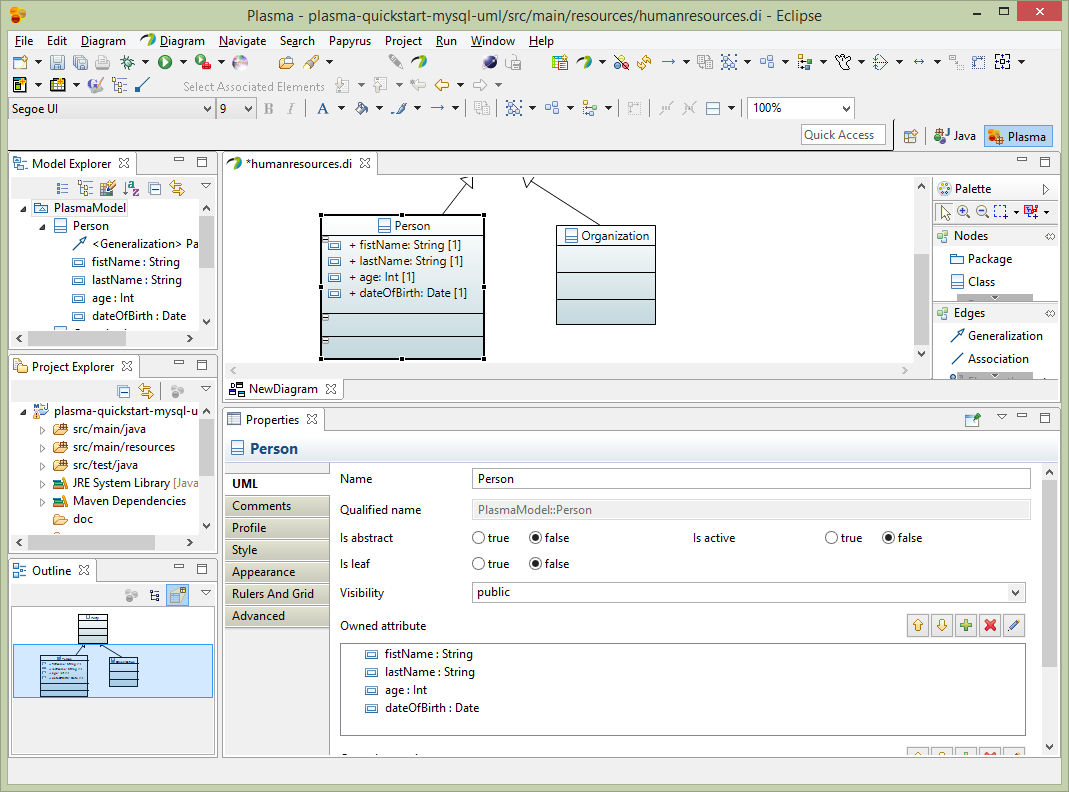
Next add a class called Party and make it anstract, then use generalization edges to make the Person and Organization classes inherit from Party.



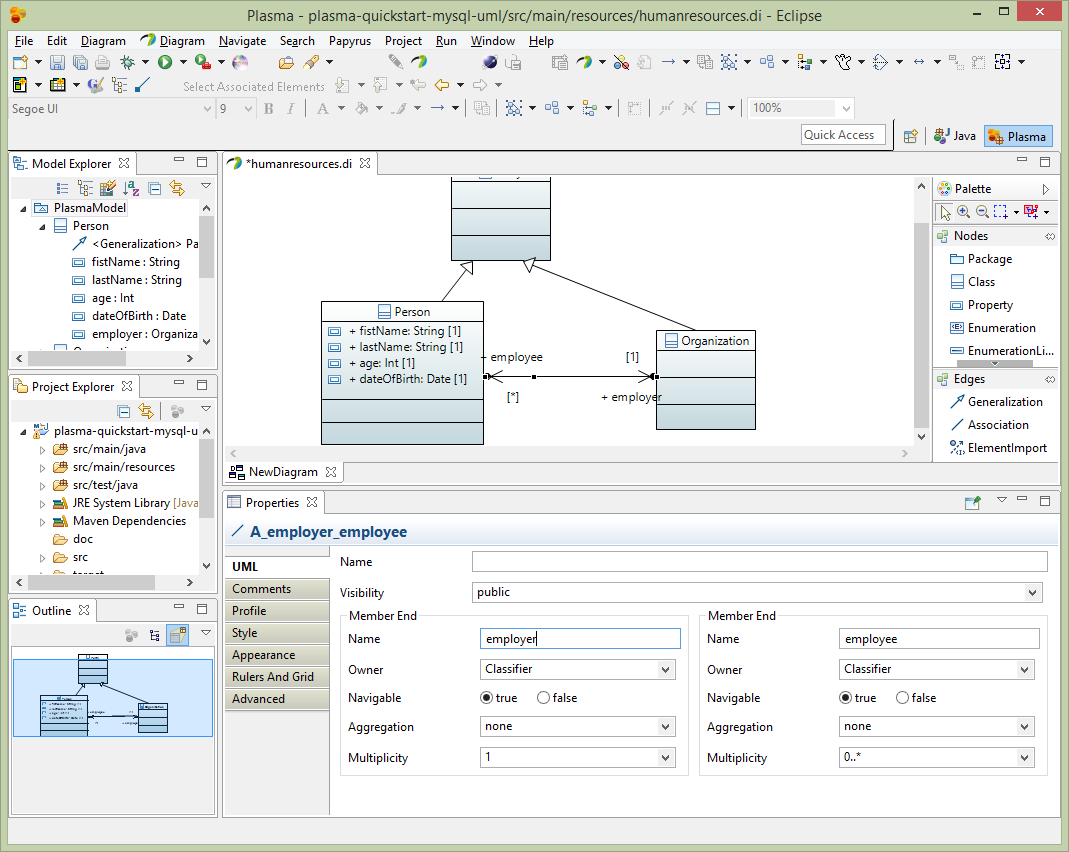
Next add properties to the Person entity. Before adding properties we need to import datatypes to use for the property datatypes. Select the root element of the project in the Model Explorer, then select Import->Import Registered Package. Then select Plasma Data Types and load the package into your model.



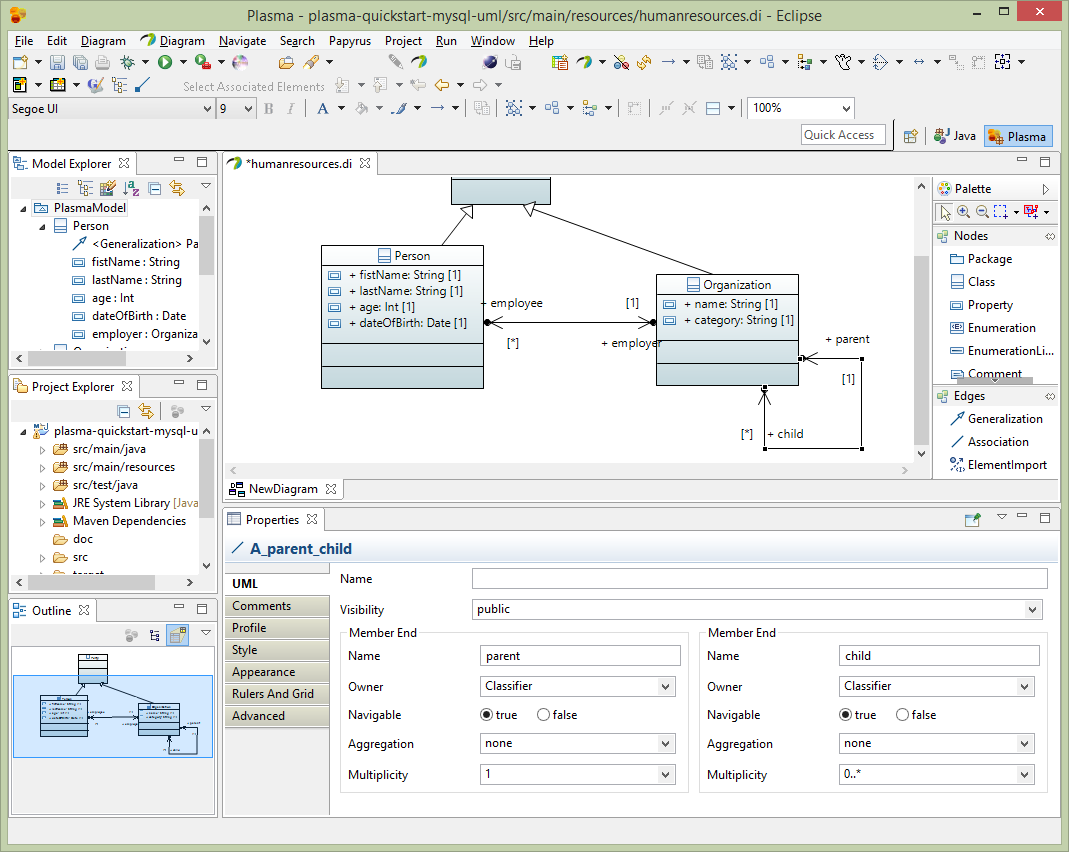
Next using the plus (+) sign in the properties editor, add 4 properties firstName, lastName, age and dateOfBirth with datatypes (String, String, Int, Date) respectively. Note that the Papyres Filters must be used to make the properties appear in the entity. Right click on the entity and select Filters->Show/Hide Contents.



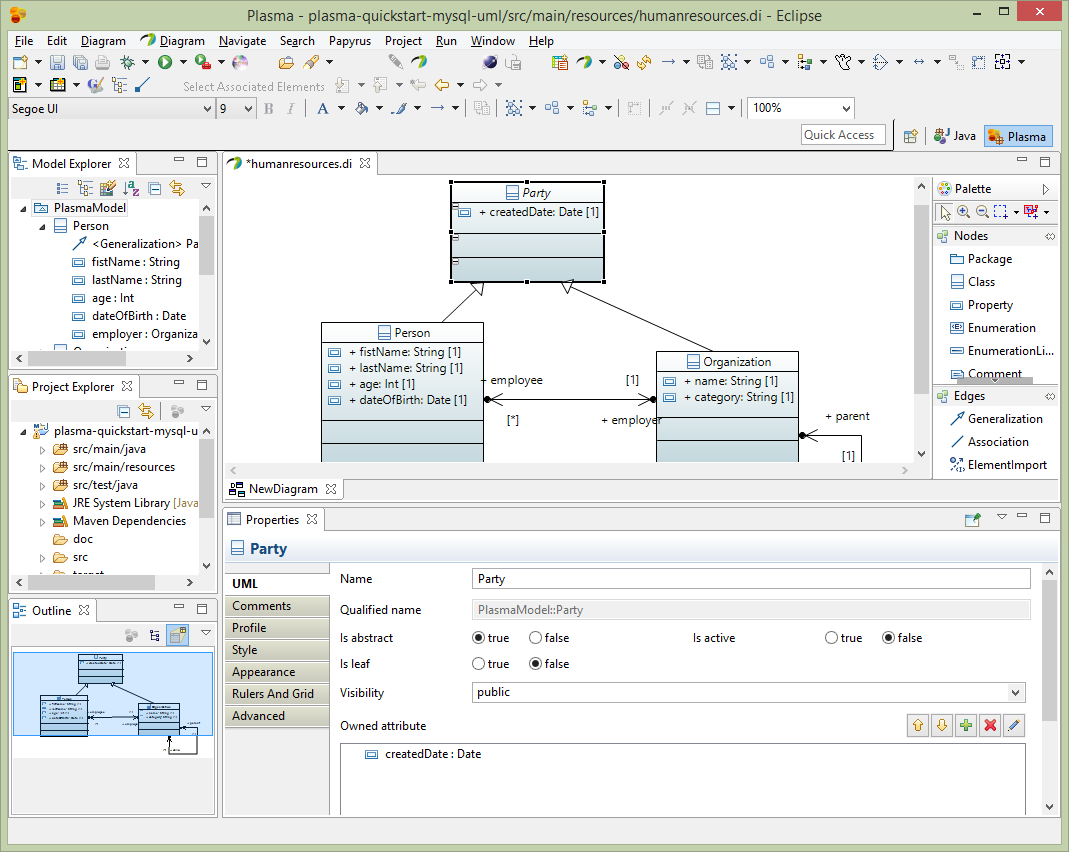
Next we will link the Person and Organization entities with an association. Select Association from the edges palette and link the Person and Organization entities. Make both ends of the association navigable and owned by the respective Classifier and make the Multiplicity on the Organization side zero-to-many as below. And instead of leaving the default names for the association ends (person, organization) make these more specific to a Human Resources model such as employer and employee, as below.



Next add 2 properties to the Organization entity name and category, both with String datatypes from the Plasma Data Types package. We will enhance the category property with an enumeration restriction later. Then add a recursive association to the Organization entity as below, with both sides of the association owned by the classifier. Note that the Papyres Filters must be used to make the properties appear in the entity. Right click on the entity and select Filters->Show/Hide Contents.

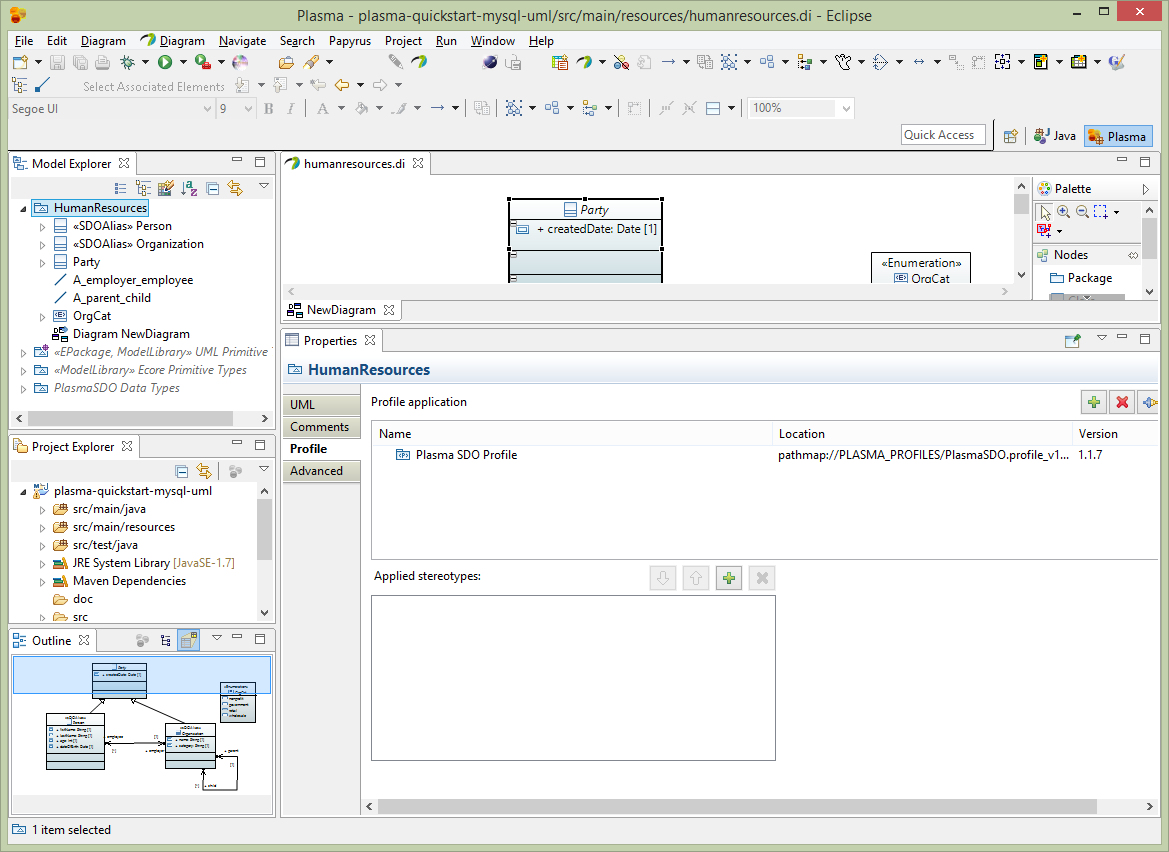


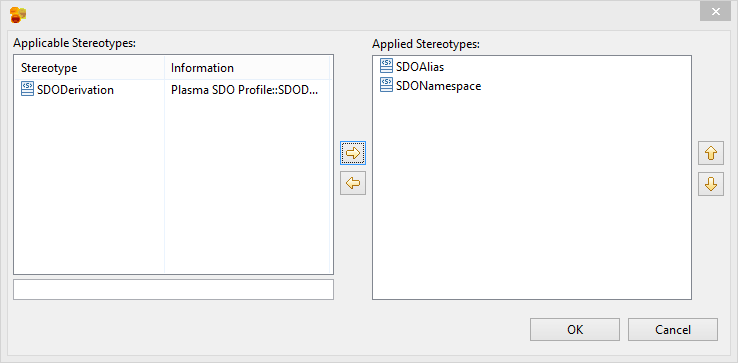
Nexte add a property to the shared entity we created previously called Party. Just add 1 property createdDate of data type Date. This property will be inherited by both the Person and Organization entities.

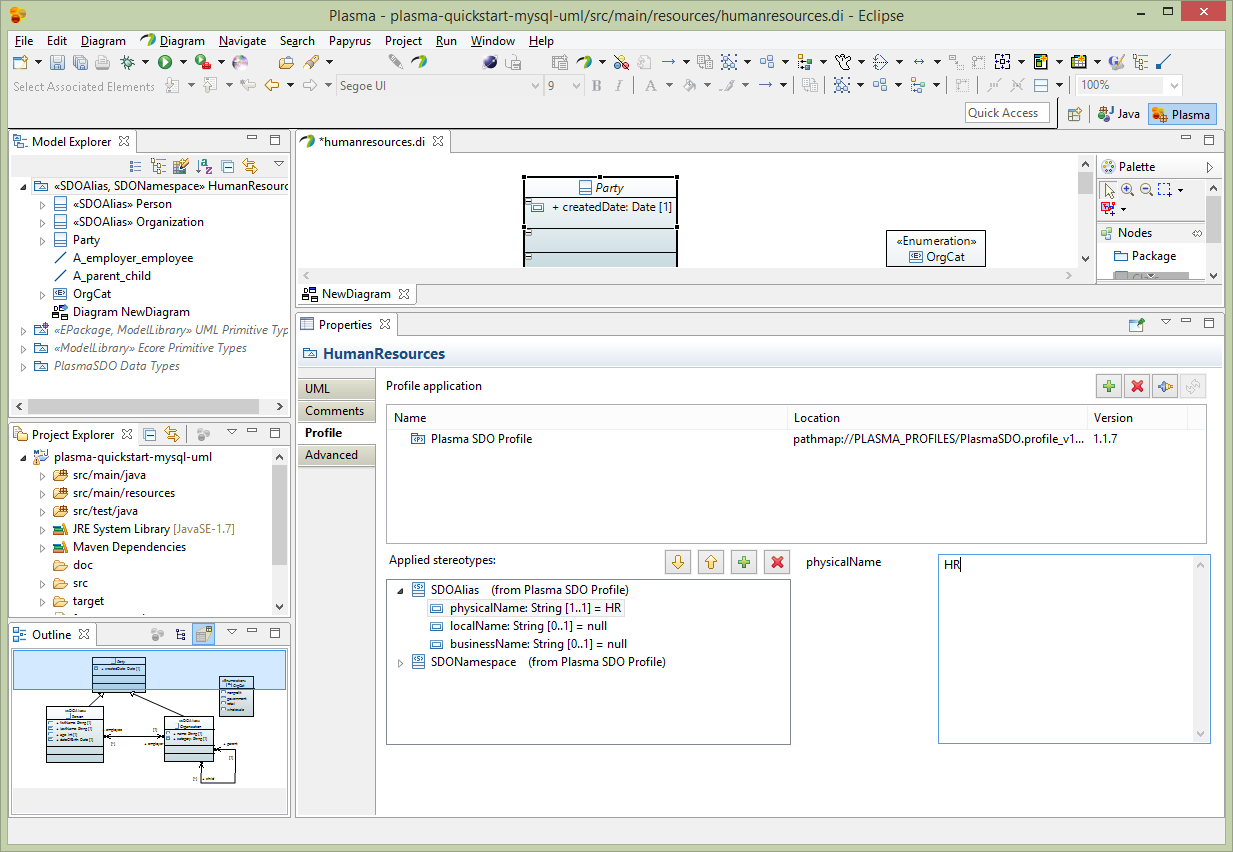


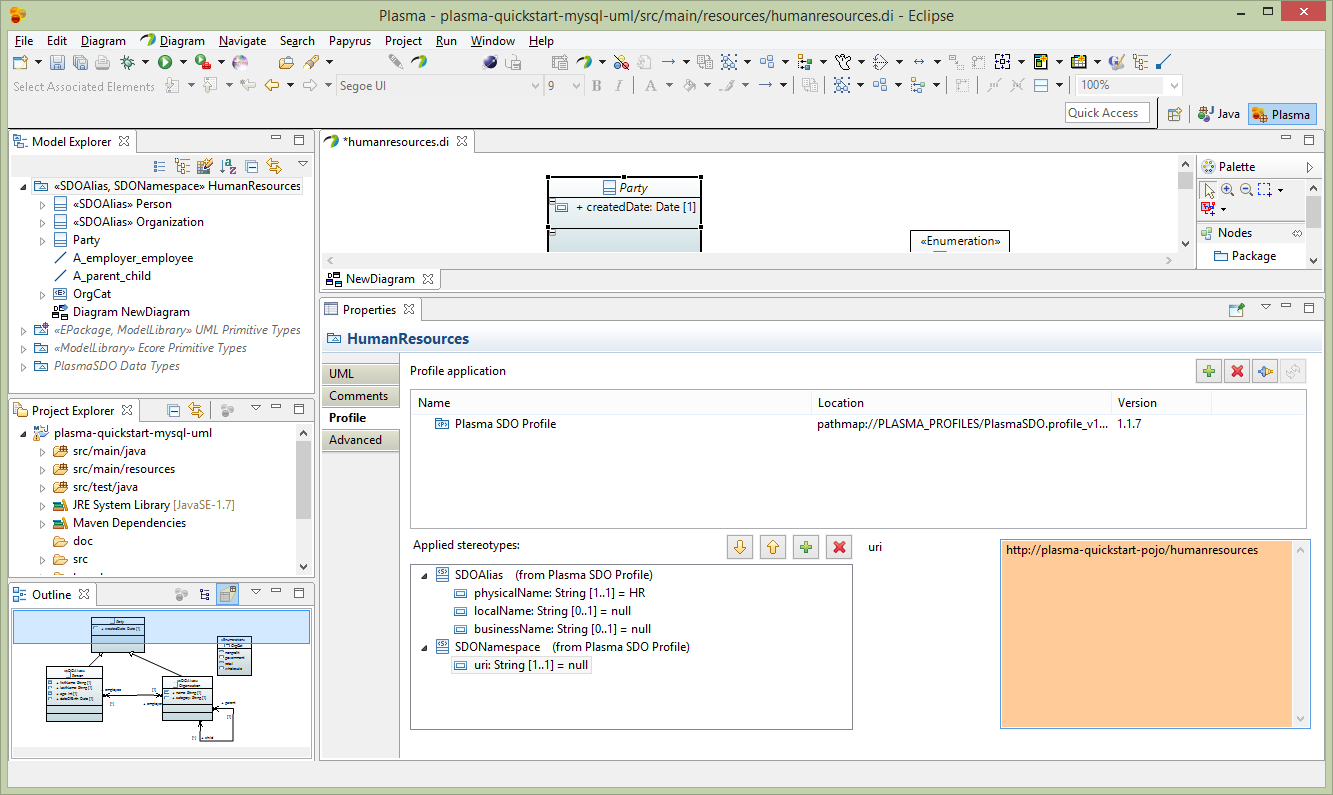
# **Enhancing a Plasma UML Model**

After creating a basic model we need to add enhancements using standard UML Profile mechanisms supported by Eclipse Papyrus, which add “facets” to various elements. These facets describe restrictions such as String property length restrictions enumeration restrictions and several others which provide a complete structural model description. The final model has enough information to allow us to generate code and persist the entities we created in one or more physical data stores. (*Note that for big-data stores such as HBase the size of the physical table is greatly influenced by the size of column names, and the logical/physical name isolation provided by Plasma is critical for controlling the size of the physical store while maintaining a readable data model and generated code*) First select the root package node on the Model Explorer and rename the package to HumanResources. Then select the Profile editor section and add the SDOAlias and SDONamespace stereotypes. Assign the SDOAlias physicalName = ‘’HR’ for the model package and the SDONamespace uri = ‘http://plasma-quickstart-pojo/humanresources’. The URI is critical and is used to link the model to a specific data store. A large model can of course have numerous packages and each package can be associated the the same or several data store providers.

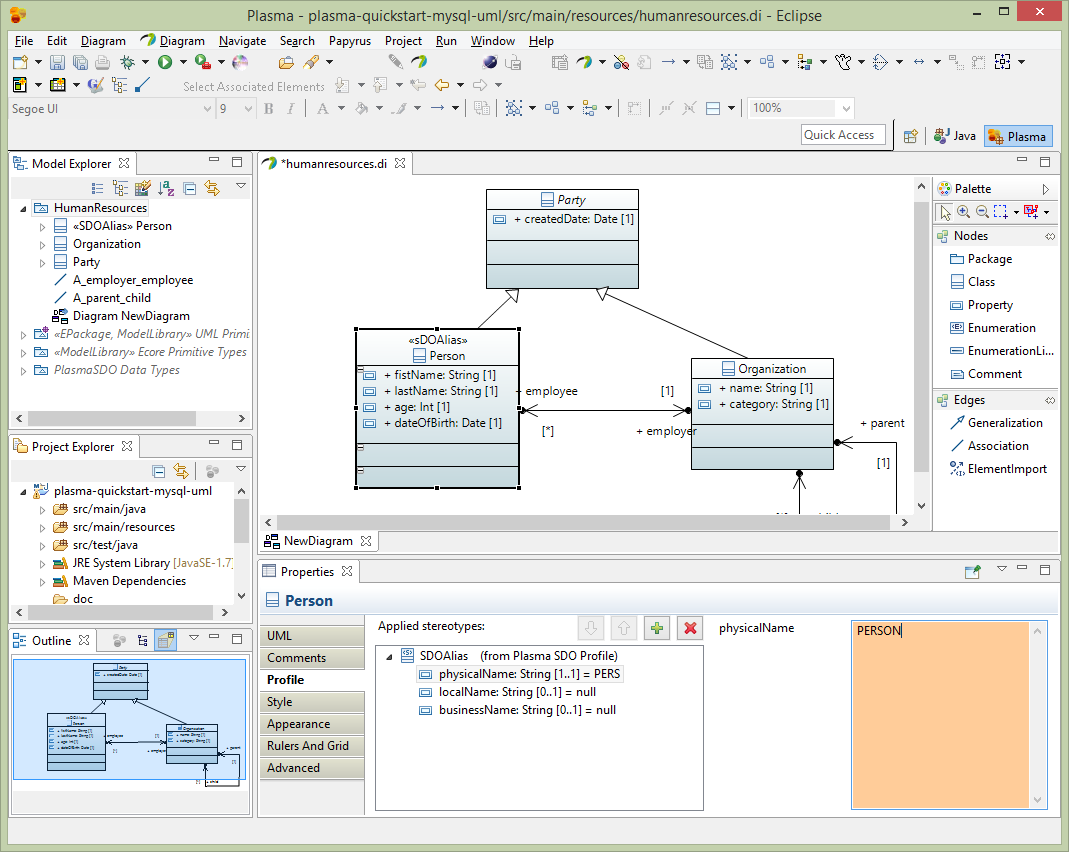


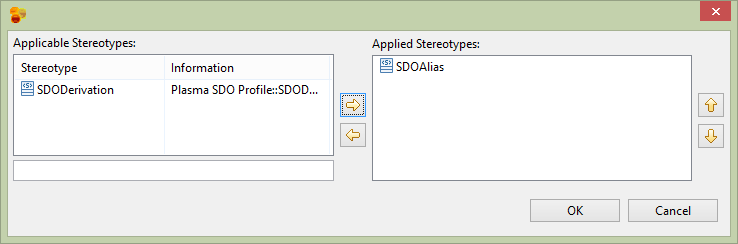




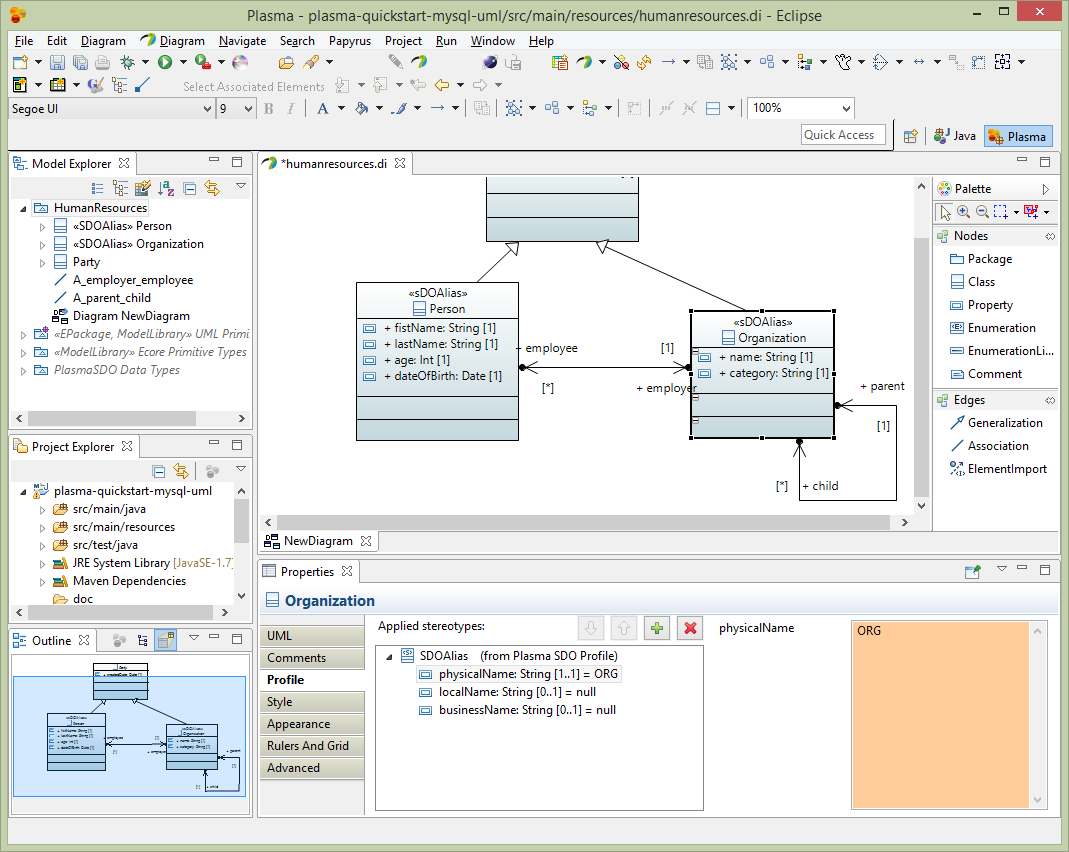


Next select the Person entity and edit the Profile section of the property editor as below. Add the SDOAlias stereotype and assign the physicalName as ‘PERSON’.

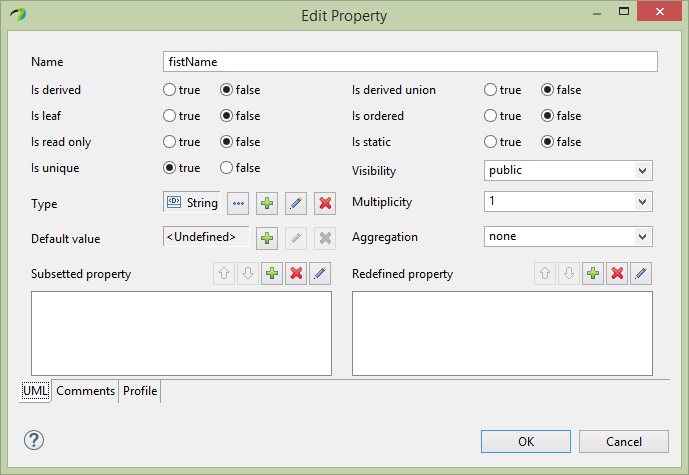


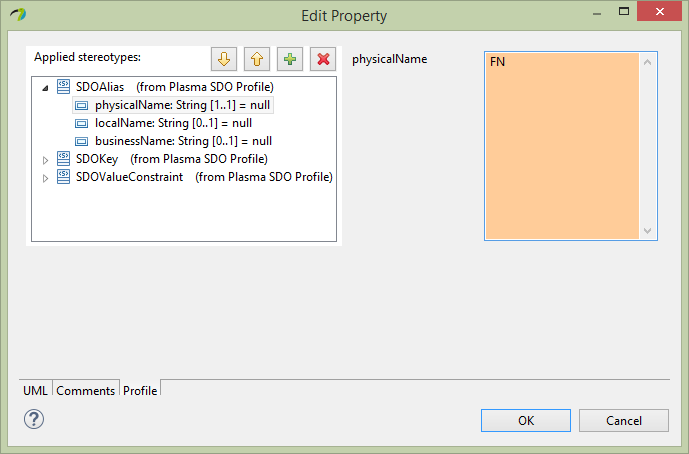


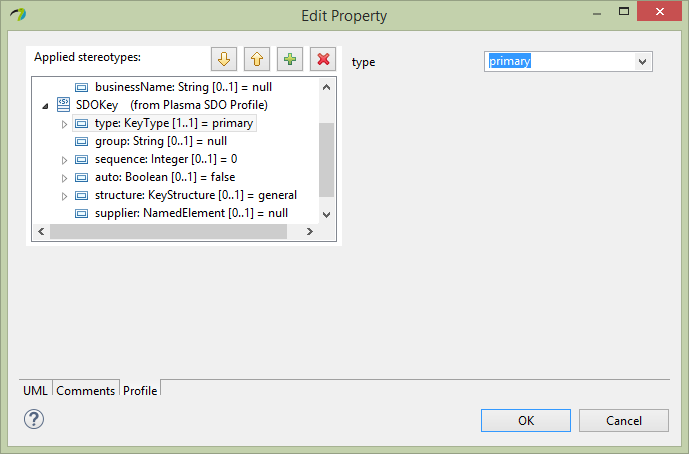
Next assign the Orgaization entity the physicalName of ‘ORG’.

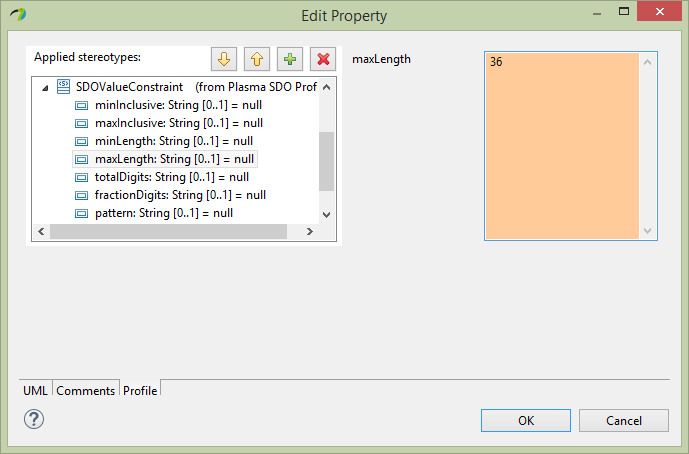


Next for the Person entity firstName property, select the Profile section of the editor and assign the SDOAlias, SDOKey and SDOValueConstraint stereotypes. Then assign SDOAlias physicalName = “FN” and SDOKey type=’primary’ and SDOValueConstraint maxLength=36 as below.



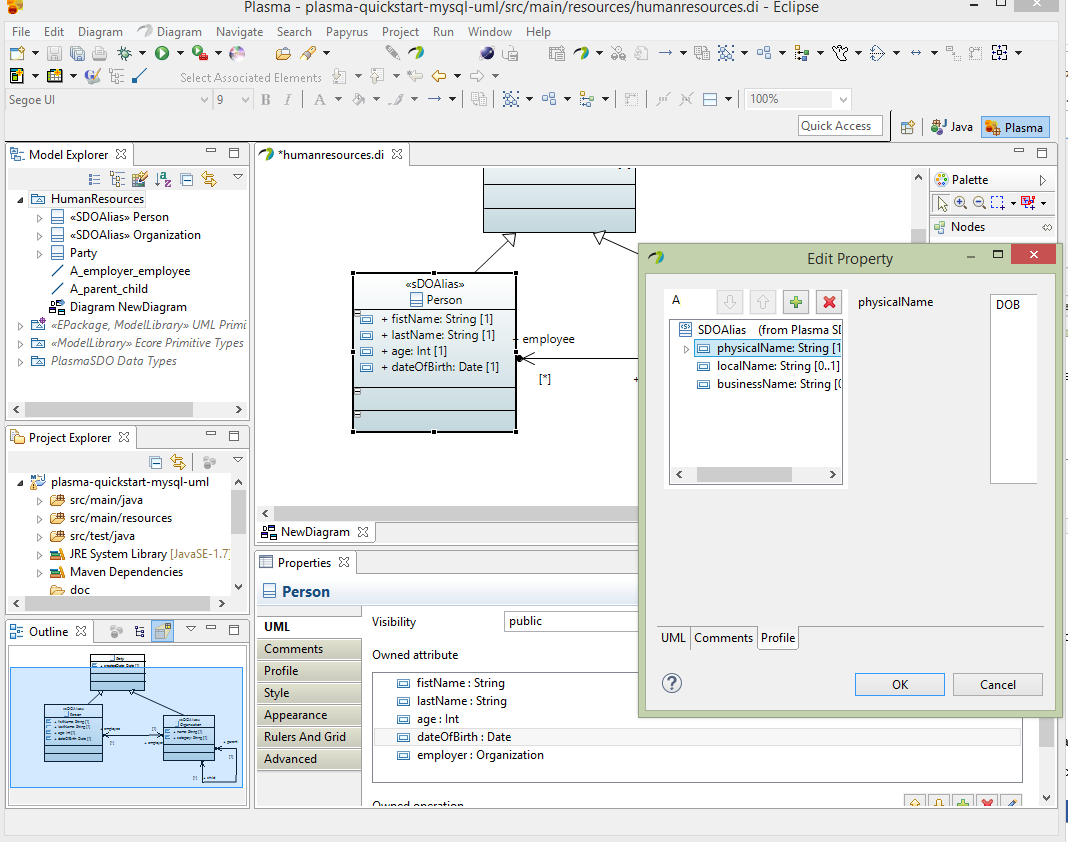




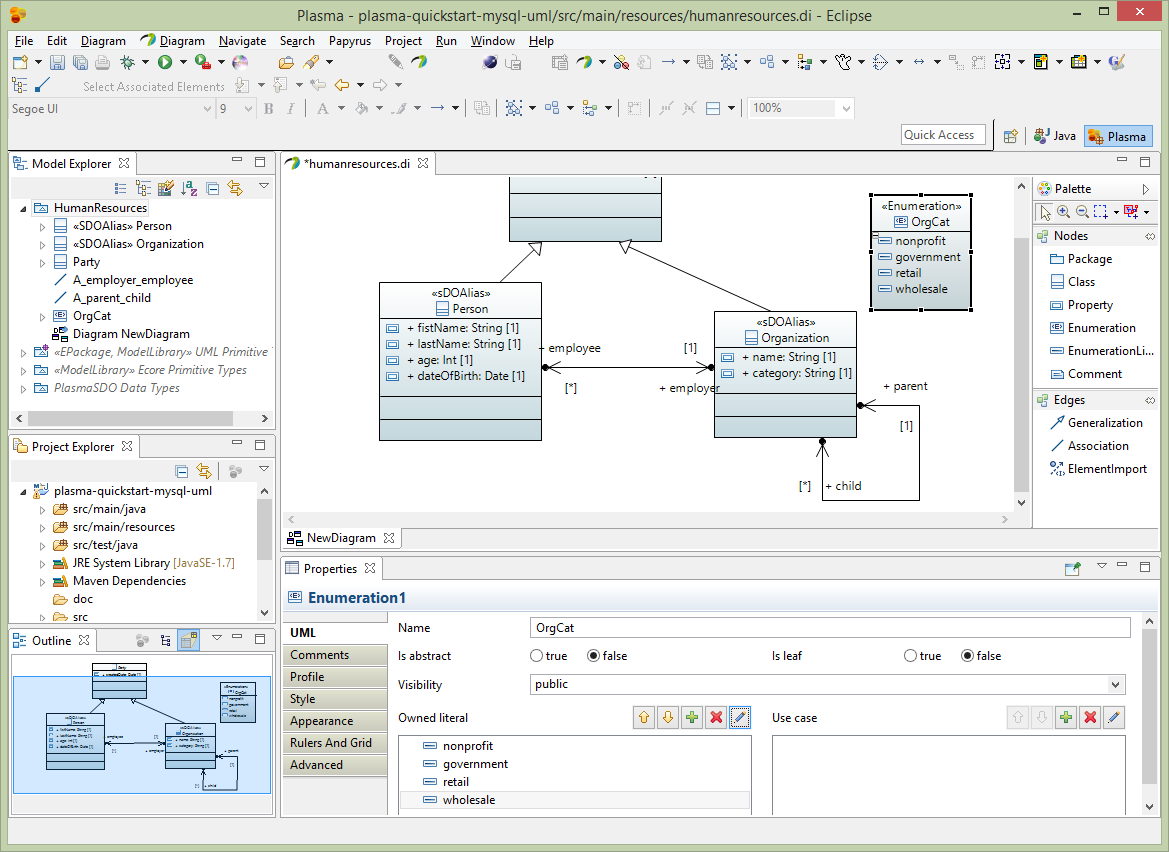


Next similarly for the Person entity lastName property, select the Profile section of the editor and assign the SDOAlias, SDOKey and SDOValueConstraint stereotypes. Then assign SDOAlias physicalName = “LN” and SDOKey type=’primary’ and SDOValueConstraint maxLength=36.

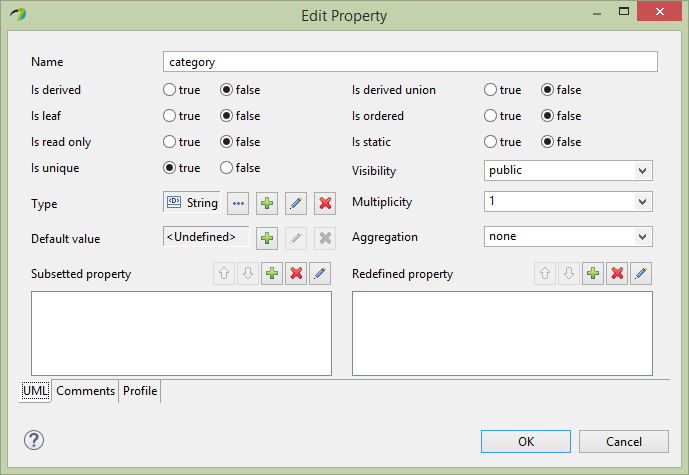
Next similarly for the Person entity dateOfBirth property, select the Profile section of the editor and assign the SDOAlias stereotype. Then assign SDOAlias physicalName = “DOB”.

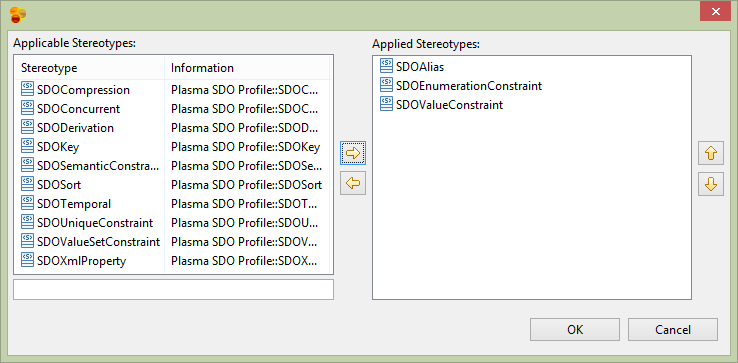


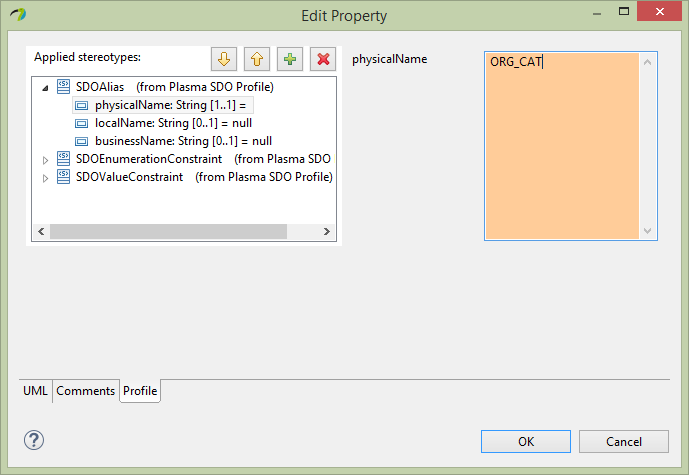
Next to enhance the Organization entity, we will first create an Enumeration ‘OrgCat’ which will be used to restrict the values of the category property we added earlier. Add four owned literals to the OrgCat enumeration ‘nonprofit’, ‘government’, ‘retail’, “wholesale” and assign each literal a physicalName which is the fitst letter i.e. ‘N’, ‘G’, ‘R’ and ‘W’ respectively as below.

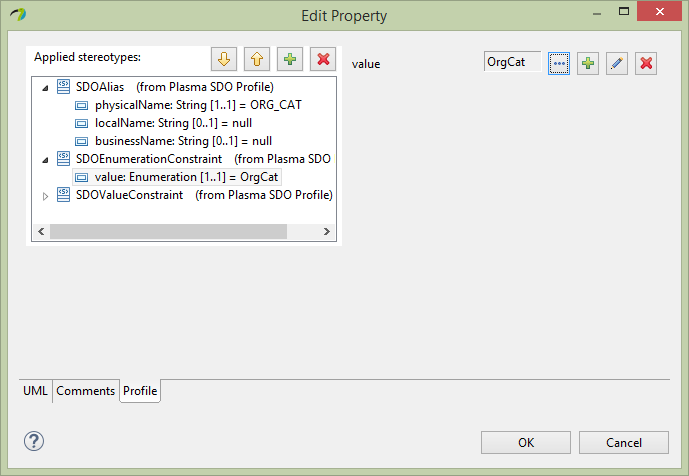


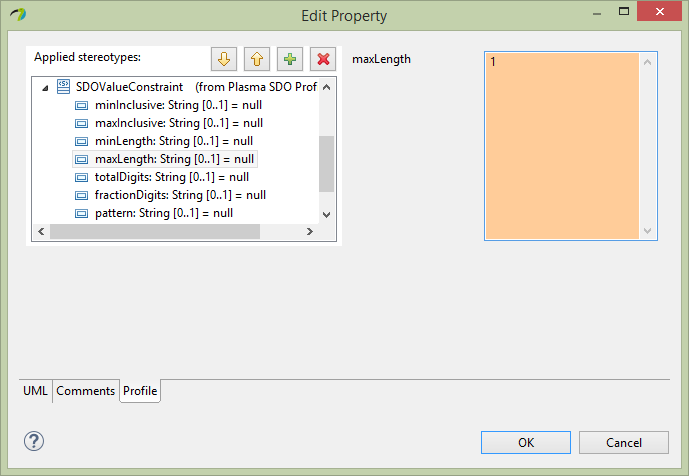
Next edit the Organization category property Profile section adding the SDOAlias, SDOEnumerationConstraint and SDOValueConstraint stereotypes. Assign the SDOAlias and SDOValueConstraint stereotypes as below and assign the SDOEnumerationConstraint to the OrgCat enumeration we created previously.











Next for the Organization entity name property, select the Profile section of the editor and assign the SDOAlias, SDOKey and SDOValueConstraint stereotypes. Then assign SDOAlias physicalName = “NAME” and SDOKey type=’primary’ and SDOValueConstraint maxLength=36 similarly.

And finally for the Party entity createdDate property select the Profile section of the editor and assign the SDOAlias stereotypes Then assign SDOAlias physicalName = “CRTD\_DT” similarly as below.

