Some examples of custom modules definition

On this page:

- Example #1: create a Module based on a Java package
- Example #2: create a COBOL Module based on a directory
- Example #3: create a DB Module based on object names
- Example #4: create a .NET Module based on .NET projects
- Example #5: create a Module based on .cpp files
- Example #6: create a Module based on SAP Tables
- Example #7: create a Module based on a .Net namespace

Summary: This page gives a few typical examples of how the possibilities of the custom module definition can be used to match a desired module content.

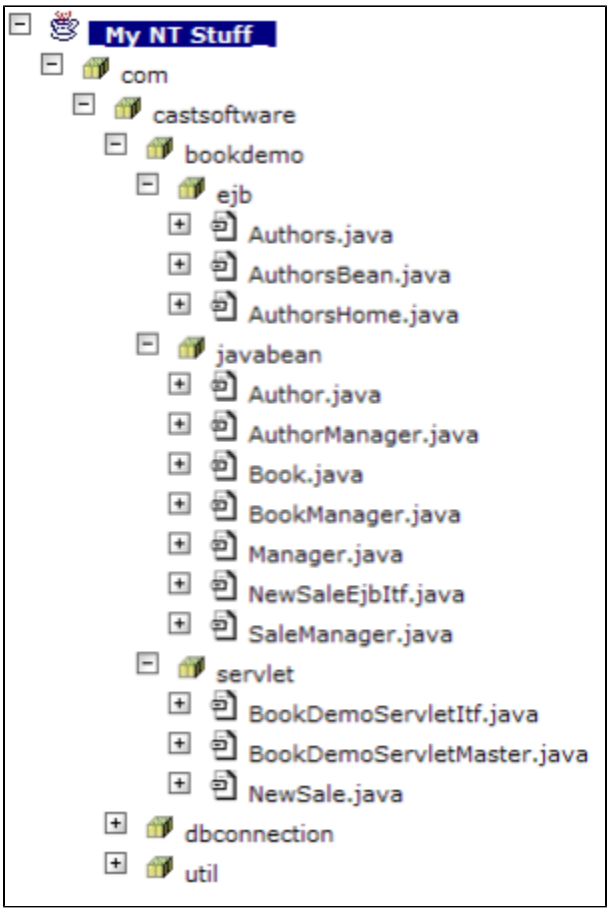
Example #1: create a Module based on a Java package

The following configuration:

- **Full Name Like com.castsoftware.bookdemo**
- **Selected Types: Java Package**

Leads to a Module composed of:

While the full analysis result is:



Explanation:

The filters on both the object type ('Java Package') and the object full name ('like com.castosftware.bookdemo') targeted the actual 'com.castsoftware.bookdemo' Java package; the Module was therefore built with this package and its content (that is, sub-packages, classes, methods and fields).

Use case:

Useful to handle functional area of an application based on the package organization.

> Note that when using the **Object Name** filter based on an **object's path** (as stored in the Analysis Service schema), some objects (notably **Java Packages**) are not saved with a path, therefore these objects will not be included in the module.

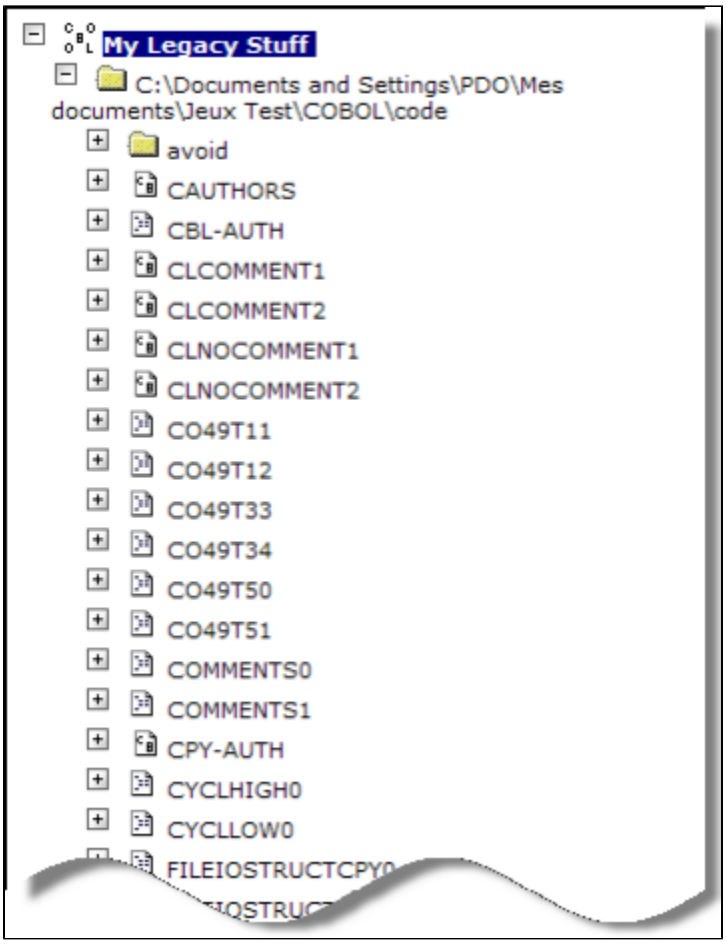
Example #2: create a COBOL Module based on a directory

The following configuration:

- **Full Name Like C:\Documents and Settings\PDO\Mes documents\Jeux Test\COBOL\code\avoid**
- **Selected Types: COBOL Directory**

Leads to a Module composed of:

While the full analysis result is:



Explanation:

The filters on both the object type ('Cobol Directory') and the object path ('like C:\Documents and Settings\PDO\Mes documents\Jeux Test\COBOL\code\avoid\') targeted the actual Cobol Directory 'C:\Documents and Settings\PDO\Mes documents\Jeux Test\COBOL\code\avoid\' directory; the Module was therefore built with this directory and its content (programs, copybooks, sections, paragraphs...).

Use case:

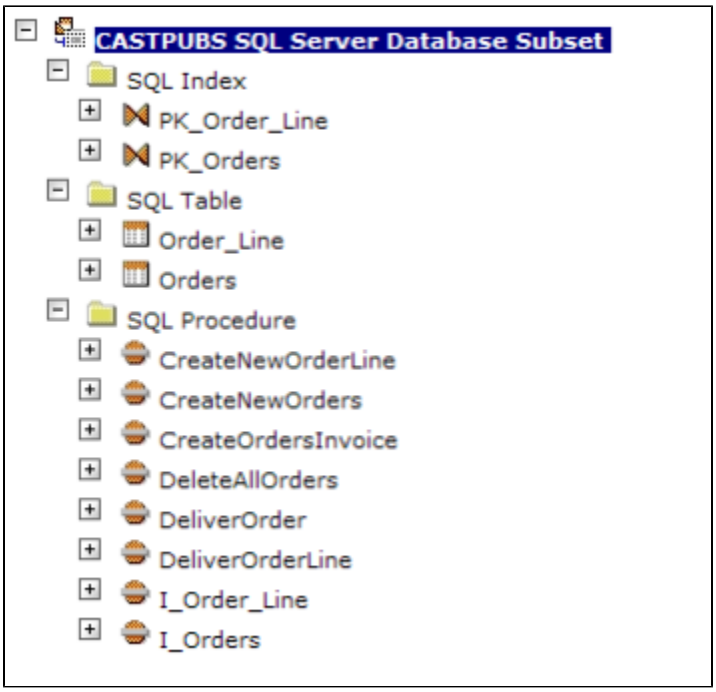
Useful to handle generated code when such source code is stored in a separate directory.

Example #3: create a DB Module based on object names

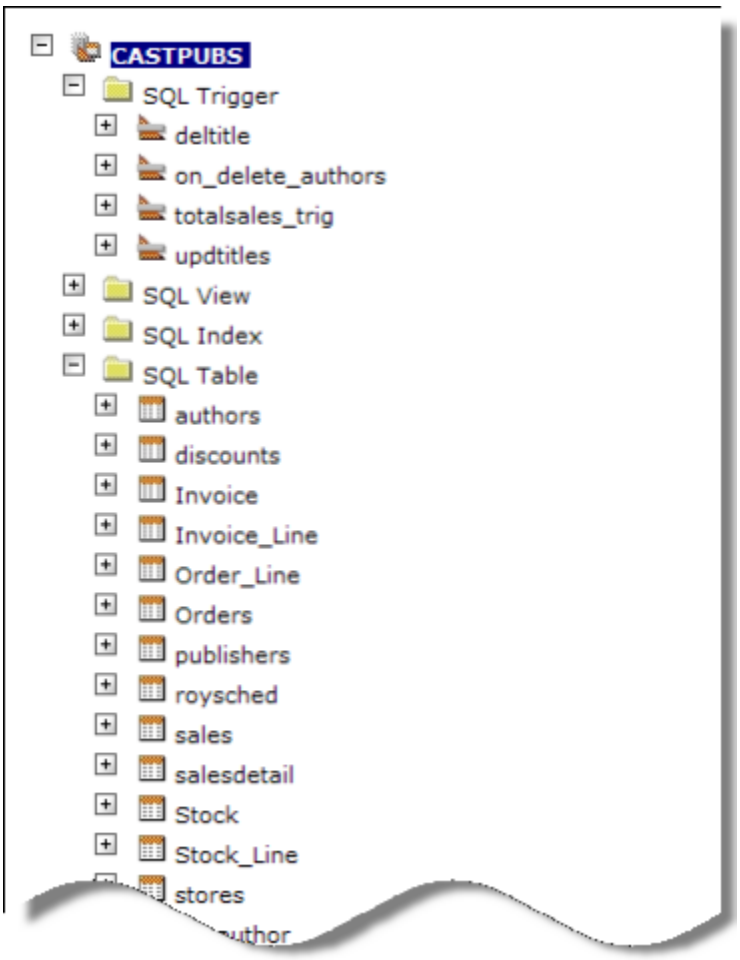
The following configuration:

- **Full Name Like %Order%**

Leads to a Module composed of:



While the full analysis result is:



Explanation:

The filters on the full name ('like %Order%') targeted the actual SQL objects whose name contains 'Order'; the Module was therefore built with these SQL objects and their content (N/A with these types of objects).

Use case:

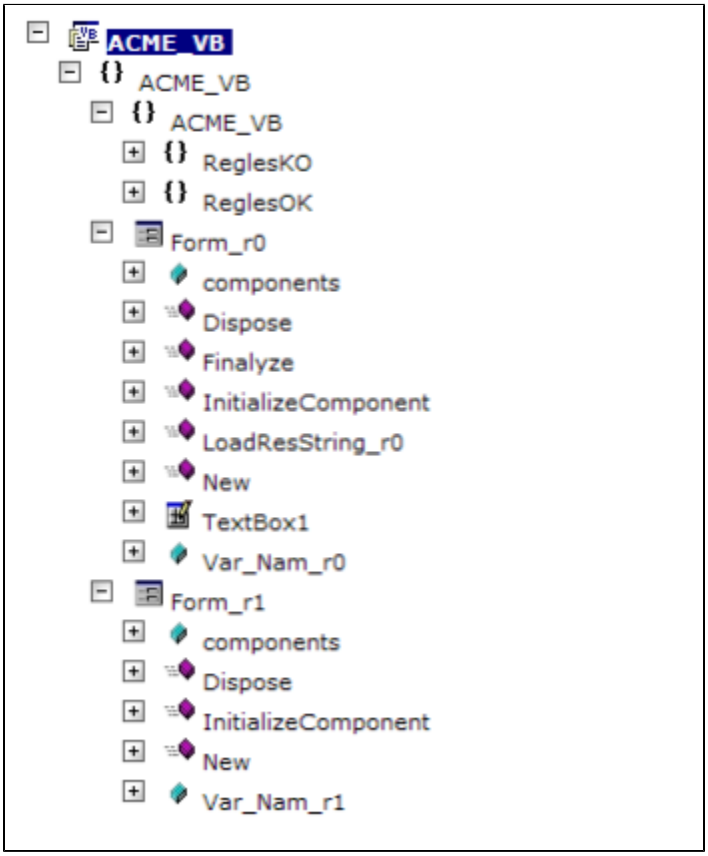
Useful to handle developments on top of packaged applications where the name alone can help discriminate objects.

Example #4: create a .NET Module based on .NET projects

The following configuration:

- **Full Name Like [ACME_VB]**

Leads to a Module composed of:



While the full analysis result is composed of ACME_VB, ACME_CS, and IBM-T42.CASTPUBS used by My .NET Stuff.

Explanation:

The filters on the full name ('like [ACME_VB]') targeted the actual .NET project name '[ACME_VB]'; the Module was therefore built with this project and its content (namespaces, forms, ...).

Use case:

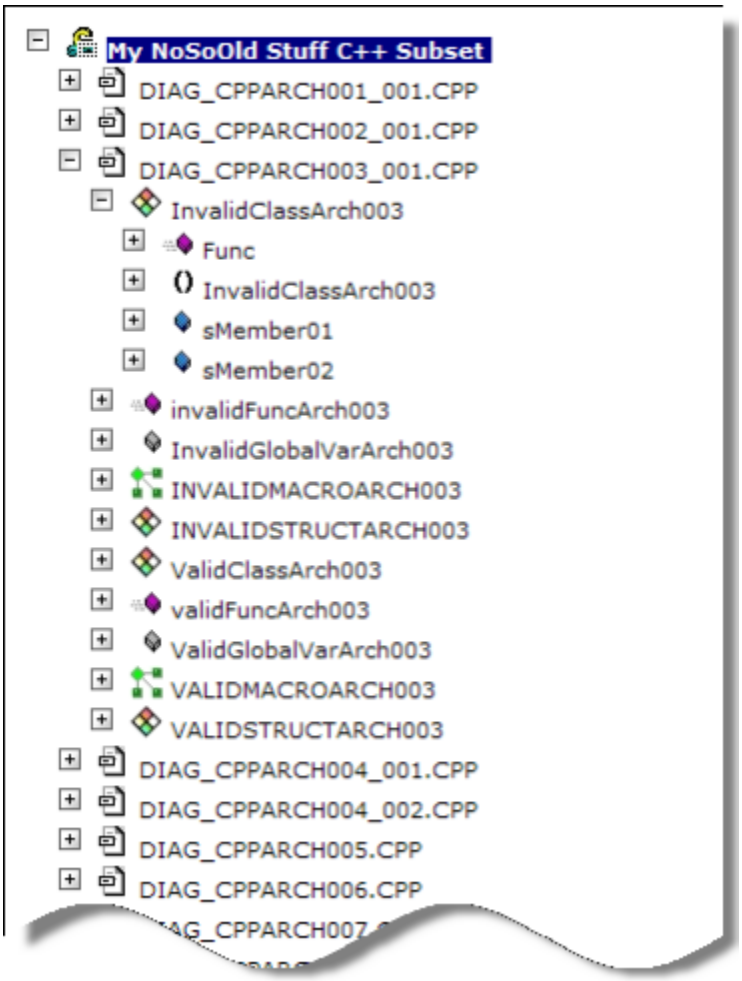
Useful to handle projects within a VB/VB.NET/.NET solution.

Example #5: create a Module based on .cpp files

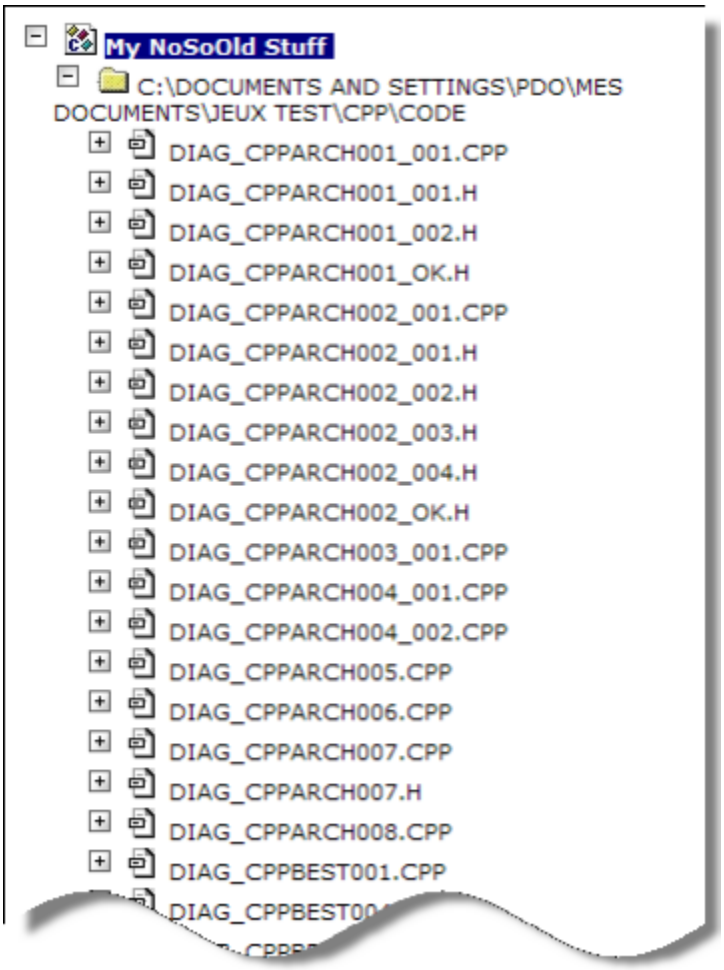
The following configuration:

- **Full Name Like %.cpp**

Leads to a Module composed of:



While the full analysis result is composed of:



Explanation:

The filters on both the full name ('like %.cpp') targeted the actual C++ implementation files and left aside the header files; the Module was therefore built with these files and their content (classes, methods,...).

Use case:

Useful to avoid assessment pollution of batches of header files. Note the directory selection can also be used, depending on the source code organization.

Example #6: create a Module based on SAP Tables

The following configuration:

- **Full Name Like SAP_TABLE/%**

Leads to a Module composed of all SAP Tables from the SAP analysis while the full analysis also contains programs, etc.

Explanation:

The filters on the full name ('like SAP_TABLE/%') targeted all objects whose full name starts with 'SAP_TABLE/' which is the naming convention within the AIP of analyzed SAP tables.

Use case:

This example is valuable with a refined full name to build modules containing some of the tables, according to a naming convention.

Example #7: create a Module based on a .Net namespace

The following configuration:

- **Full Name Like n1**
- **Selected Types: .Net namespace**

Explanation:

The filters on both the object type ('.Net namespace') and the object full name ('like n1') targeted the actual 'n1' .Net namespace; the Module was therefore built with this namespace and its content (that is, sub-namespaces, classes, methods and fields).

> ⚠ In order to distribute source files between modules, the Module "build by namespace" requires that you create filters to distribute them between Modules. As they can contain objects from multiple namespaces, they are not "belonging to" any namespace. For instance, add a filter on "full name like %xyz%.cs".

Use case:

Useful to handle functional area of an application based on the package organization.