

# What ...are the naming code conventions in my application?

## Requirements

- Installed ClassiX® system
- Codewright or another text editor
- knowledge of:

basic ClassiX® design widget	action list statement variable
---------------------------------	--------------------------------------

## Introduction

Every programming language follows certain rules. Here we provide a short overview about the required conventions for the following “cooking recipes”.

### Content

- comments
- identifiers
- modules
- module structure

### Comments

Comments in the C++ style are accepted.

#### Syntax

Single-line comments:

```
// This is a single-line comment
```

```
Window(A, ...) // single-line comments can also be at the end.
```

Multi-line comment:

```
/* this is a multi-line comment.  
Just like in C++ and HTML this comment form gets  
accepted, too.
```

```
*/
```

**Real comments in the source code shall always be written in English!**

## Identification

An identifier is any element, which can be named (in some extend) freely. This includes variable names, message names, module names, names of self-defined events, self-defined classes, functions, etc. An identifier has to start with a letter or a `_`, followed by letters, numbers and `_`.

### General conventions:

- **variables(names), slots and fixed data fields of classes**, start with a lower case character or with `_`.
- **statements to control the program activity** are completely displayed in lower case characters
- **BASIC DATA TYPES, CLASSES, FLAGS, CONSTANTS, ACTIONS, MESSAGES and EVENTS** are displayed in CAPITALS.
- **Widgets, widget names, macro names, class methods** and all remaining **statements** start with a capital letter. The rest is capitals and lower case characters. The first letter of each word is a capital.
- **Modul names** are written in capitals and lower case characters.

**Attention InstantView® is case-sensitive – upper- and lower case needs to be considered!**

## Modules

A module can also be described as a statement aggregation. The entire AppsWh program code written in InstantView® and all customer adjustments are put into modules.

The keyword 'module' determines, that all following definitions belong to this module. The **window objects** that have been defined with InstantView®, their **action lists** and the **variables** and **functions** that are defined in these lists, are always assigned to a module. Many modules (and statements) are followed by parameter lists. These lists are in parentheses. Several parameters are separated by commas.

### *Basic module structure:*

**Module(name, HELP(fileName)): BaseModuleName**

```
[  
// Var  
// Declare  
// Action List  
// Define;]  
// Square brackets contain variable declarations, action lists and  
// function declarations and -definitions.  
// This defined order should be kept.  
//  
// The action lists are followed by the window definitions which are  
// similar to the module definition. At first, here are also action  
lists // inside square brackets, but usually no variables and functions.
```

**Widget(objektName-aliasName, Flags, x, y, w, h, statischerText, bitmap)**

```
[  
// action list  
]  
{  
// child objects  
}  
// Square brackets are only supposed to contain action lists.  
// Curly braces contain widget child objects.  
// For a window, it can be menu items, text fields, lists and buttons. For  
// a list, it can be column definitions.
```

**Example:**

An entry field of a window is its **child object**. Consequently, the window is parent object of the entry field.

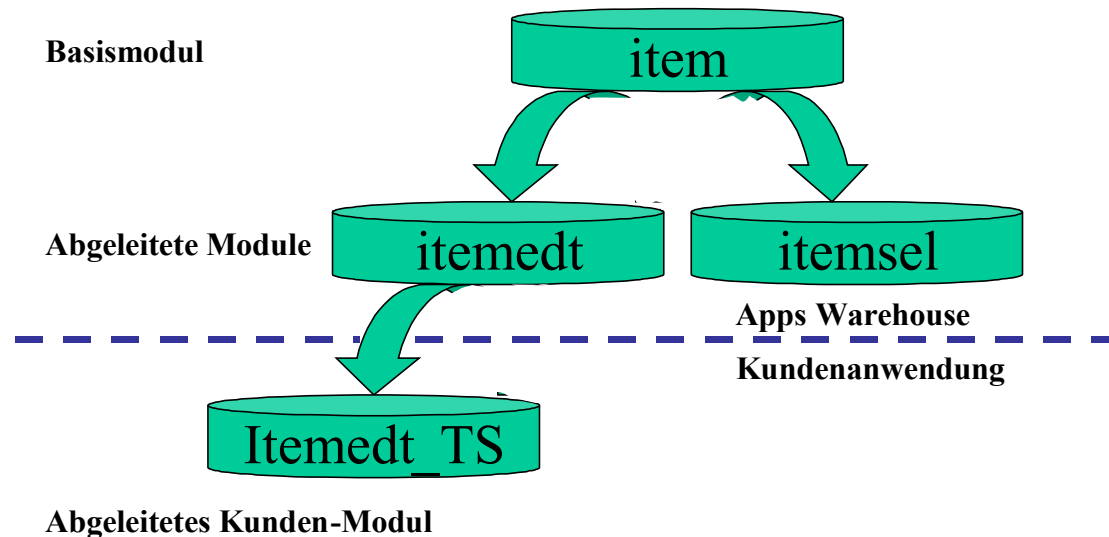
```
Window(A, . . .) //parent of B
{
  String(B, . . .) //entry field: child of A
}
```

Parameter:

Name	a definite name contains the
FileName	HTML-file name for the documentation and <a href="#">online help</a> . A HTML-file should be specified for every module.
BaseModuleName	Name of an already defined module from which this module is derived.

In the module definition, the **:** is – just like in C++ - a character for inheritance. If a BaseModuleName is specified, it is a **derived** module. The naming of derived modules also follows certain conventions:

In the AppsWarehouse®, the name of a derived module always starts with the name of the according basic module. If an individual module has been derived for the customer, a **\_** and the **customer code**, (technically) written in capitals, is attached to the original name of the basic module. All module names are maximum 8 characters + the length of the **customer code**.



## **Abstract**

Now, we know that there are certain conventions, referring to upper and lower case in InstantView<sup>®</sup>, which need to be followed. Program code is aggregated in modules, for which there is a particular structure, as well.

## **Further information**

**How... do I change the code in my application?**