

# Wie... lauten die Code Konventionen in meiner Anwendung?

## Voraussetzungen

- Installiertes ClassiX® System
- Codewright oder ein anderer Texteditor
- Kenntnisse über:

Grundlegender Aufbau von ClassiX® Widget	Action List Statement Variable
---	--------------------------------------

## Einführung

Jede Programmiersprache folgt bestimmten Regeln. Hier geben wir eine kurze Übersicht über die Konventionen, die in den folgenden „Kochrezepten“ benötigt werden.

### Inhalt

- Kommentare
- Identifikatoren
- Module
- Modulstruktur

### Kommentare

Kommentare im C++ Stil werden akzeptiert.

#### Syntax

Einzeiliger Kommentar:

```
//          Dies ist ein einzeiliger Kommentar
```

Window(A, ...) // Einzeilige Kommentare dürfen auch am Ende stehen.

Mehrzeiliger Kommentar:

```
/*          Dies ist ein mehrzeiliger Kommentar.  
           Wie in C++ und HTML wird diese Form  
           des Kommentars ebenfalls akzeptiert.
```

```
*/
```

**Echte Kommentare im Sourcecode sollen immer in Englisch geschrieben werden!**

## Identifikatoren

Unter einem Identifikator versteht man ein beliebiges Code-Element, welches vom Anwender (teilweise) frei benannt werden kann.

Darunter sind zu verstehen: Variablennamen, Messengernamen, Modulnamen, Namen selbstdefinierter Events, selbstdefinierte Klassen, Funktionen, etc.

Ein Identifikator muss mit einem Buchstaben oder `_` beginnen, danach können Buchstaben, Zahlen sowie `_` folgen.

### Allgemein gelten folgende Konventionen:

- **variablen(namen), slots und feste Datenfelder von Klassen**, beginnen mit einem Kleinbuchstaben oder mit `_`.
- **Anweisungen zur Steuerung des Programmablaufs** beginnen mit einem Kleinbuchstaben und werden kleingeschrieben.
- **ELEMENTARE DATENTYPEN, KLASSEN, FLAGS, KONSTANTEN, AKTIONEN, MESSAGES und EVENTS** werden in GROSSBUCHSTABEN dargestellt.
- **Widgets, Widgetnamen, Makronamen, Methoden von Klassen** und alle übrigen **Anweisungen** beginnen mit einem Grossbuchstaben und werden in Klein- und Großbuchstaben weiterschrieben. Der erste Buchstabe jedes Wortes wird großgeschrieben.
- **Modulnamen** werden in Klein- und Großbuchstaben geschrieben.

**Achtung InstantView<sup>®</sup> ist casesensitiv, d.h. es muss die Groß- und Kleinschreibung beachtet werden.**

## Module

Grob kann man ein Modul als eine Zusammenfassung von Statements (Anweisungen) bezeichnen. Im wesentlichen ist der ganze, in der Sprache InstantView<sup>®</sup> geschriebene, Programmcode des AppsWh und aller Kundenanpassungen in Modulen untergebracht.

Das Schlüsselwort 'Module' bestimmt, dass alle folgenden Definitionen zu diesem Modul gehören. Die mit InstantView<sup>®</sup> definierten **Windowobjekte**, ihre **Aktionslisten**, die darin definierten **Variablen** und **Funktionen** sind immer einem Modul zugeordnet.

Vielen Modulen (und Anweisungen) folgen Parameterlisten. Diese sind in runde Klammern eingeschlossen. Mehrere Parameter werden durch Kommata getrennt.

### *Grundstruktur eines Moduls:*

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

```
[  
// Var  
// Declare  
// Action List  
// Define;]  
// Eckige Klammern schließen Variablendeklarationen, Aktionslisten und  
// Funktionsdeklarationen und -definitionen ein.  
// Die Reihenfolge in diesem Definitionsteil sollte eingehalten werden.  
//  
// Nach der Aktionsliste folgen die Fensterdefinitionen, welche in sich  
// der Moduldefinition ähnlich sind. Auch hier gibt es zunächst in  
// eckige // Klammern eingeschlossene Aktionslisten, jedoch im Normalfall  
// keine // Variablen und Funktionen.
```

**Widget(objectName~aliasName, Flags, x, y, w, h, statischerText, bitmap)**

```
[  
// Action List  
]  
{  
// Child Objekte  
}  
// Eckige Klammern sollen nur Aktionslisten einschließen.  
// Geschweifte Klammern schließen die Child Objekte eines Widget ein.  
// Für ein Fenster sind dies z.B. dessen Menüpunkte, Textfelder, Listen  
// und Schaltflächen. Für eine Liste sind dies z.B. dessen  
// Spaltendefinitionen.
```

**Beispiel:**

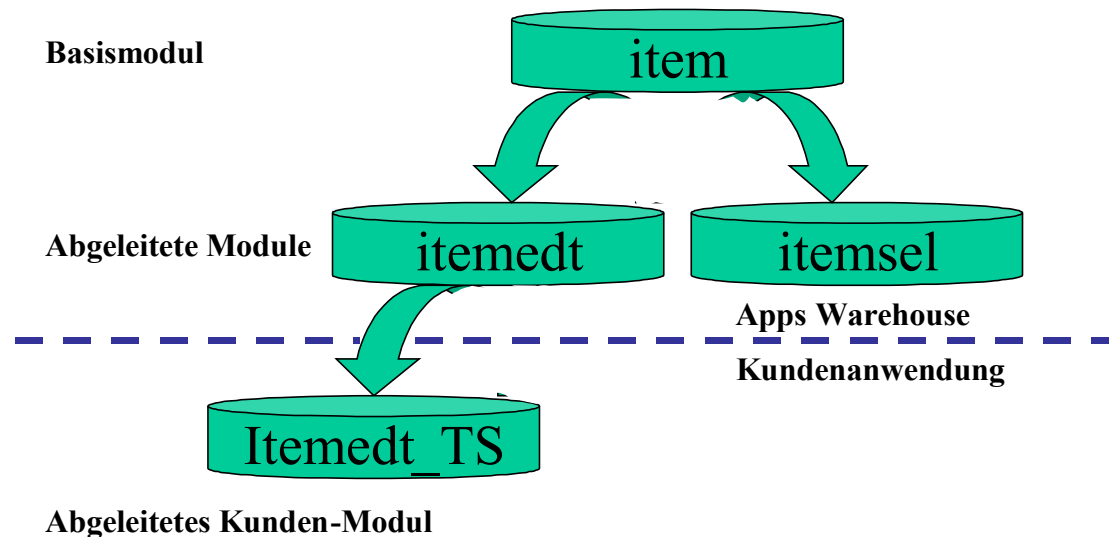
Ein Eingabefeld auf einem Window ist das **Child-Objekt** des Windows, und umgekehrt verhält sich das Window zum Eingabefeld als dessen Parent-Objekt.

```
Window(A, . . .) //Parent zu B
{
  String(B, . . .) //Eingabefeld: Child zu A
}
```

Parameter:

Name	ein eindeutiger Name
FileName	enthält den Namen einer HTML-Datei für die Dokumentation und <b>Online-Hilfe</b> . Zu jedem Modul sollte eine HTML-Datei angegeben werden
BaseModuleName	Name eines bereits definierten Moduls von welchem dieses Modul abgeleitet ist.

Der **:** ist bei der Moduldefinition wie in C++ das Zeichen für Vererbung. Ist also ein BaseModuleName angegeben, so handelt es sich um ein **abgeleitetes** Modul. Die Benennung abgeleiteter Module folgt ebenfalls bestimmten Konventionen: Innerhalb des AppsWarehouse® beginnt der Name eines abgeleiteten Moduls immer mit dem Namen des jeweiligen Basismoduls. Wurde für den Kunden ein individuelles Modul abgeleitet, so wird an den Namen des Basismoduls ein **\_** und das (Theoretisch) großgeschriebene Kundenkürzel angehängt. Für alle Modulnamen gilt die maximale Länge von 8 Zeichen + Länge des Kundenkürzels.



## **Zusammenfassung**

Wir wissen jetzt, dass in InstantView® bestimmte Konvention bezüglich der Groß- und Kleinschreibung herrschen, die eingehalten werden müssen. Programmcode wird in Modulen zusammengefasst, bei denen eine bestimmte Struktur ebenfalls vorgegeben ist.

## **Weiterführendes**

**Wie... verändere ich den Code meiner Anwendung?**