

# Wie... wird eine Spalte in eine Listbox eingefügt?

Teil 1: wie funktioniert Vererbung in ClassiX®?

## Voraussetzungen

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

<a href="#">Grundlegender Aufbau von ClassiX®</a>	<a href="#">Widget</a>
<a href="#">Konventionen</a>	<a href="#">Action List</a>
<a href="#">Klassen</a>	<a href="#">Event</a>
<a href="#">Vererbung (Inheritance)</a>	<a href="#">Message</a>
<a href="#">Modul</a>	<a href="#">Variablen</a>

## Aufgabenstellung:

In dem letzten „Kochrezept“ lernten wir wie das Monitor-Fenster aufgerufen wird und ein Modul verändert werden kann. Jetzt soll eine Spalte in eine Listbox der Anwendung eingefügt werden.

### Inhalt

- ObjectListView
- Praktische Anwendung
- Was ist passiert?
- Wie funktioniert Vererbung
- Wo darf Code geändert werden.

ObjectListView, OListView, OLView

Eine Listbox wird mit **ObjectListView** in InstantView® definiert. OlistView und OLView sind akzeptierte Kürzel.

### Syntax

ObjectListView(name~aliasName, flags, x, y, w, h)

Parameter:

Name	*	Indentifikator oder KLASSE::ausdr
aliasName		ein zusätzlicher Identifikator
Flags		Flags

X	*	Position X (in Minicells)
Y	*	Position Y (in Minicells)
W	*	Breite (in Minicells)
H	*	Höhe (in Minicells)

\* - Pflichtparameter

Die Steuerung der Anzeige erfolgt über die Aktionsliste und das Event INITIALIZE:

### Syntax

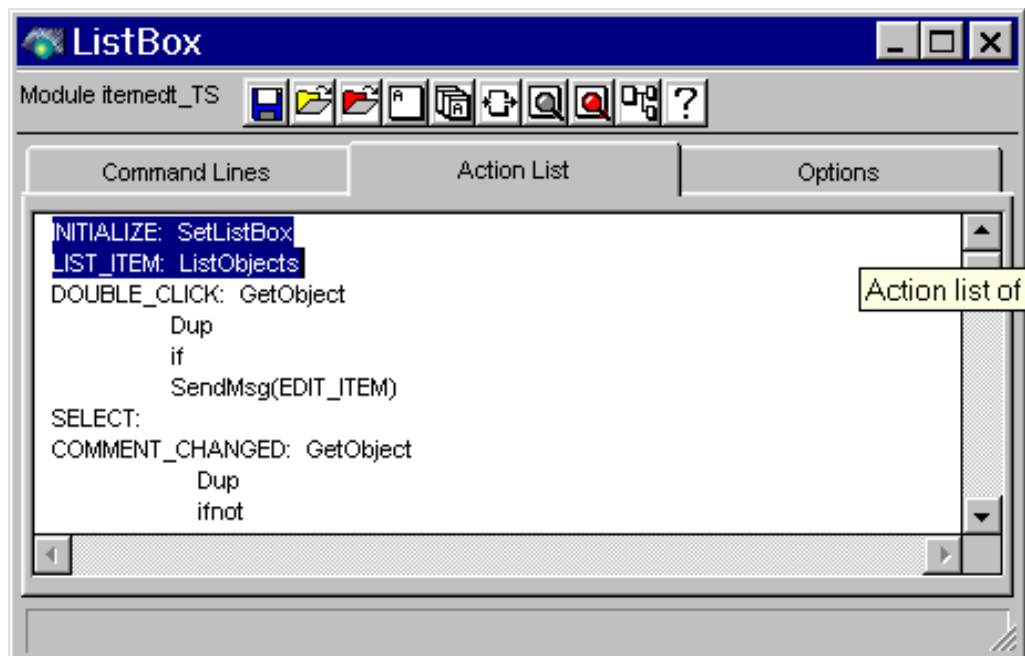
```
[ INITIALIZE : SetListBox ]
```

```
Define(SetListBox);
```

definiert die Inhalte der Listbox . Die Funktion SetListBox ist weiter oben im Modul oder einem Basismodul des Moduls definiert.

### Praktische Anwendung

ClassiX<sup>®</sup> wird geöffnet und das Arbeitsgebiet „Teilestamm“ angeklickt. Das Monitorfenster für die Listbox wird geöffnet und der Reiter „Action List“ wird aufgerufen.



INITIALIZE : SetListBox finden wir hier, aber im Code von itemedt\_TS.mod finden wir weder das erwartete ObjectListView "ListBox" mit der Aktionsliste mit INITIALIZE noch die Funktion SetListBox.

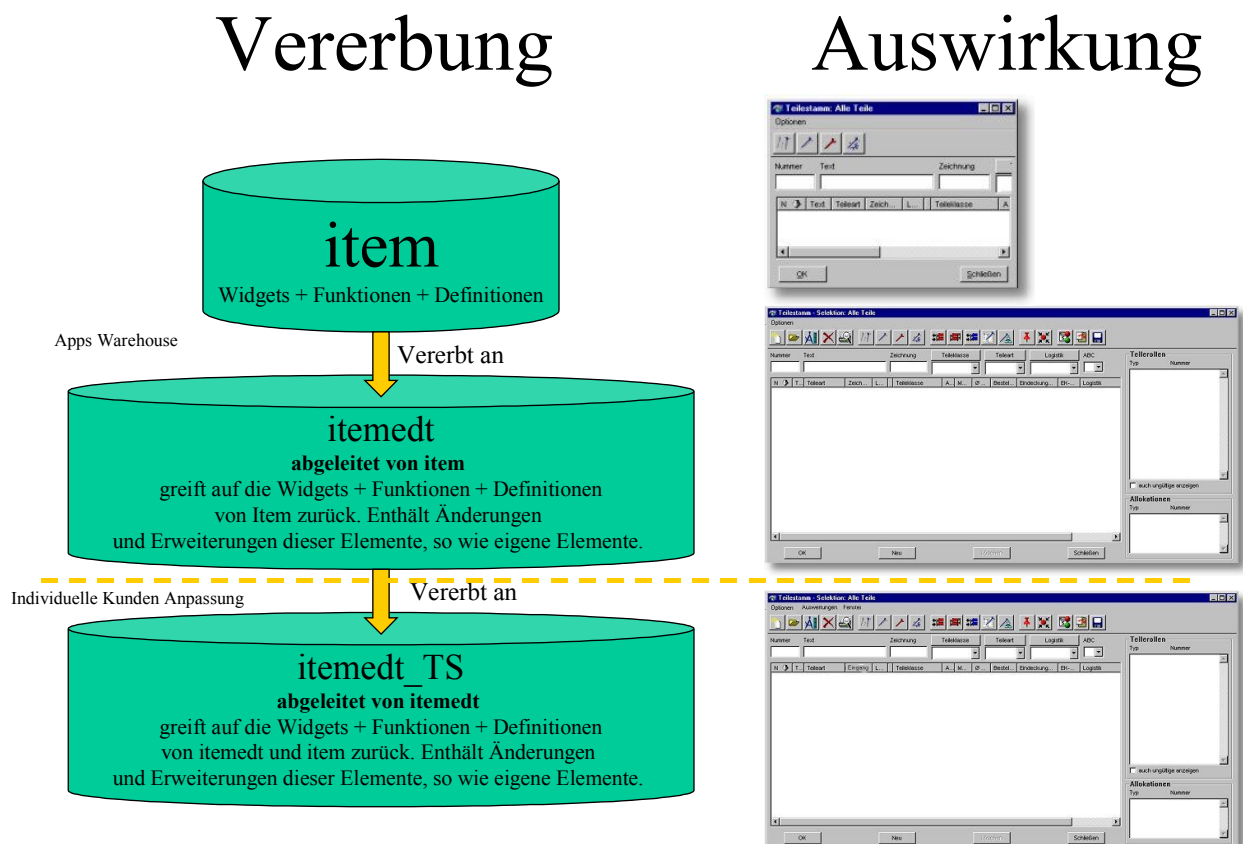
## Was ist passiert?

In der Symbolleiste wird der Modulname mit „itemedt\_TS“ angegeben. Aus dem How-to „Konventionen“ ist bekannt, dass es sich bei Modulen, die mit einem Unterstrich und einem großgeschriebenen Kürzel enden, um abgeleitete Kundenmodule handelt..

Das aufgerufene Modul ist also ein individuelles Kundenmodul, welches von einem bereits definierten Modul abgeleitet wurde: Der Code der Listbox ist in einem Basismodul zu finden.

## Wie also funktioniert die Vererbung?

Leitet man ein neues Modul von einem anderen Modul ab, übernimmt letzteres die Rolle des Basismoduls des neuen Moduls. Dabei wird der gesamte Code des Basismoduls „vererbt“. Das neue Modul ist also wie ein Klon des Basismoduls, ohne dass der gesamte Code erneut kopiert werden muss. Änderungen in dem abgeleiteten Modul wirken sich nicht auf das Basismodul aus, und vor allem wirken sich Änderungen am Basismodul in der Regel automatisch auch auf alle abgeleiteten Module aus. Dementsprechend können von demselben Basismodul verschiedene Module abgeleitet werden, die alle ein unterschiedliches Verhalten zeigen. Dagegen werden alle Elemente des Basismoduls übernommen, sofern sie nicht durch ein namensgleiches Element im abgeleiteten Modul überschrieben oder verändert werden.



Im einzelnen bedeutet dies für die Elemente:

### Variable

Existiert im Basismodul die Variable x, so besitzt der abgeleitete Modul ebenfalls eine Variable mit diesem Namen. Eine Wertänderung der Variablen x eines weiteren abgeleiteten Moduls hat keine Auswirkung auf x.

### Statement

Wurde im Basismodul ein Statement s vereinbart, so ist dieses Statement auch im abgeleiteten Modul bekannt, es sei denn, dort wird eine neue Variante von s definiert. Innerhalb der Neudefinition, die oft eine Erweiterung der alten Anweisung ist, kann man sich auf das Statement des Basismoduls mit der Schreibweise

**moduleName::s** oder **super::s**

beziehen, wobei immer super::s verwendet werden sollte!

### Aktionsliste

Auch die Aktionsliste eines Basismoduls wird übernommen. Im abgeleiteten Modul kann die Reaktion auf ein bestimmtes Ereignis dort einfach neu definiert werden. Statt der im Basismodul definierten wird jetzt die im abgeleiteten Modul überschriebene Aktion ausgelöst. Die überschriebenen Anweisungen können mit [SendMsg\(, MESSAGENAME\)](#) oder [SendMsg\(,SUPER\)](#) aktiviert werden.

### Windows

Zum Basismodul gehörende Windows werden auch Bestandteil abgeleiteter Module. Enthält das abgeleitete Modul ein Window mit gleichem Namen, so erbt dieses Window alle Childobjekte des Basis-Window für die im abgeleiteten Window kein gleichnamiges Gegenstück existiert.

### Wo darf Code geändert werden?

Der Superuser darf nur Code in kundenspezifischen Modulen ändern. Wurde das zu ändernde Modul noch nie geändert und somit auch noch nicht abgeleitet, dann muss ein neues, kundenspezifisches Modul abgeleitet werden. **Die Module des AppsWarehouse® dürfen NICHT vom Superuser geändert werden.**

### Zusammenfassung

Wir lernten wie eine Listbox definiert wird. Um den Code verändern zu können, erhielten wir grundlegende Kenntnisse von der Vererbung in ClassiX® und erhielten eine Regel welchen Code ein Superuser ändern darf und welchen nicht..

## **Weiterführendes**

**Teil 2 – Wie finde und ändere ich den Code?**

**Teil 3 – Wie finde ich ein Datenfeld?**

**Teil 4 – Komplexe Datentypen!**

**Teil 5 – Schönheitspflege für die Listbox**

**Teil 6 – Wie finde/verändere ich die Spaltenüberschrift?**