Implementierungen

Es existieren viele Smalltalk-Implementierungen:

Squeak: frei verfügbar und plattformunabhängig. Entwickelt von einer aktiven internationalen Gemeinschaft.

VisualWorks: kommerziell und plattformunabhängig, frei verfügbar für nicht-kommerzielle Nutzung.

Gemstone: hochperformante Objektdatenbank. GLASS (Gemstone, Linux, Apache, Seaside, Smalltalk).

Und andere: GNU Smalltalk, Smalltalk/X, SyX, VA Smalltalk, Dolphin...

Anwendungen

Lehre: EToys (Squeak), SqueakBot, BotsInc, Scratch ...

Multimedia: Sophie, OpenCroquet, Plopp ...

Web Entwicklung: Seaside, Aida, Komanche, Swazoo ...

Persistenz: Objektorientierte Datenbanken (Magma, GemStone), Relationale Datenbanken (MySQL, PostgreSQL), Objektrelationales Mapping (Glorp).

Eine Mal-Session mit Plopp

Glossar

Image: Die Smalltalk-Umgebung beinhaltet einen persistenten Objektspeicher, das Image. Im Image werden alle Klassen- und Methoden-Objekte des Systems und somit sein gesamter Source-Code gespeichtert, aber auch der aktuelle Zustand aller Werkzeug-Objekte wie Klassenbrowser, Inspektoren oder Debugger.

Virtuelle Maschine: Eine virtuelle Maschine ist ein Programm, das fähig ist, andere Programme auszuführen und damit die Portabilität von Anwendungen erleichtert.

Reflektion: Eine Sprache nennt man reflektiv, wenn sie Mechanismen beinhaltet, mit deren Hilfe man Code zur Laufzeit betrachten und ändern kann.

Dynamische Typisierung: Einige Sprachen zwingen die Entwickler den Typ jeder Variable (Integer, String ...) anzugeben. Dies nennt man statische Typisierung. Mit dynamischer Typisierung limitiert man Variablen nicht auf einen gegebenen Typ.

Smalltalk

eine rein
objektorientierte
Programmiersprache
und dynamische
Entwicklungsumgebung

EUROPEAN SMALLTALK USER GROUP
esug
www.esug.org

Wichtige Konzepte von Smalltalk

Squeak ist ein Smalltalk-Dialekt. Smalltalk ist eine objektorientierte Sprache, dynamisch typisiert, mit einer minimalen Syntax, die in fünfzehn Minuten gelernt werden kann. Ein wesentlicher Vorteil der Smalltalk-Sprache ist ihre Kohärenz:

- Alles ist ein Objekt: Klassen, Methoden, Zahlen, etc.
- Es existieren nur eine kleine Anzahl von Syntaxregeln und keine Ausnahmen.

Smalltalk läuft auf einer virtuellen Maschine. Die Entwicklung erfolgt in einem Image, in dem alle Objekte "leben" und modifiziert werden können.

Smalltalk-Syntax

Reservierte Wörter	
`nil`	Undefiniertes Objekt
`true` und `false`	Boolesche Objekte
`self`	Aktuelles Objekt
`super`	Aktuelles Objekt im Kontext der Superklasse
`thisContext`	Laufzeit-Stack der aktuellen Methode

Reservierte Zeichen	
`:=` (oder ←)	Zuweisung
^ (oder ↑)	Ergebnisrückgabe
`| var1 var2 var3 |`	Deklaration dreier temporärer Variablen
`$a`	Zeichen `a`
`#(abc 123)`	Array mit zwei Literalen `#abc` und `123`
. (punkt)	Stoppzeichen eines Ausdrucks
;	Nachrichten-Kaskadierung
`[ ]`	Code-Block (auch ein Objekt!)
`"Kommentar"`	
`'Zeichenkette'`	

Nachrichtenversand

Der Aufruf einer Methode erfolgt durch das Versenden einer Nachricht an ein Empfängerobjekt, welches anschließend wiederum ein Objekt zurückliefert. Es existieren nur unäre, binäre und Schlüsselwort-Nachrichten.

Unäre Nachrichten. Eine unäre Nachricht hat kein Argument.

```
array := Array new.
array size.
```

Das erste Beispiel erzeugt eine neue Instanz von `Array`, indem es die unäre Nachricht `new` an die Klasse `Array` sendet. Das zweite Beispiel liefert die Größe dieses leeren Arrays, also `0`.

Binäre Nachrichten. Eine binäre Nachricht hat nur ein Argument, wird durch ein Symbol bezeichnet und oft für arithmetische Ausdrücke verwendet.

```
3 + 4.
'Hello', ' World'.
```

Die Nachricht `+` wird an das Objekt `3` mit `4` als Parameter geschickt. Im zweiten Fall wird die Nachricht `,` an den String `'Hello'` mit `' World'` als Parameter geschickt.

Schlüsselwort-Nachrichten. Im Unterschied zu Sprachen wie C, C++, Java oder Python werden Parameter nicht mittels Klammern übergeben, sondern in Anlehnung an die natürliche Sprache mit in den Methodenaufruf eingebaut. Die Parameter werden dazu zwischen sogenannte Schlüsselwörter positioniert, jeweils durch einen Doppelpunkt getrennt.

```
'Smalltalk' allButFirst: 5.
3 to: 10 by: 2.
```

Das erste Beispiel ruft die Methode `allButFirst:` mit dem Parameter `5` auf einem String auf. Das zweite Beispiel ruft die Methode `to:by:` mit den Parametern `10` und `2` zwischen den Schüsselwörtern `to:` und `by:` auf der Integerzahl `3` auf.

Blöcke

Blöcke sind Code in eckigen Klammern, der nicht sofort ausgeführt wird, sondern bedingt, wiederholt oder später.

```
#('Hallo ' 'Welt')
       do: [:wort | Transcript show: wort].
```

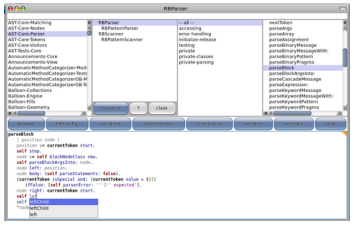
Dieses Beispiel schickt die Nachricht `do:` an ein Feld von Zeichenketten und übergibt dabei einen Block. Die Methode `do:` führt den Block mit jedem Feldelement aus, das als Parameter `wort` übergeben wird und nacheinander `'Hallo '` und `'Welt'` ausgibt.

Entwicklungsumgebung

Die meisten Smalltalk-Implementierungen beinhalten eine Entwicklungsumgebung, die es erlaubt, im Source-Code zu navigieren und mit Objekten zu interagieren. Diese Umgebungen beinhalten viele Werkzeuge, die alle in Smalltalk selbst implementiert sind:

- ein Klassen- und Methodenbrowser;
- Refactoring-Werkzeuge;
- Objekt-Inspektoren;
- Debugger;
- etc.

Die Umgebung erlaubt die direkte Ausführung von Code mittels der Tastatur und zeigt sofort das Ergebnis.



Der Pharo-Code-Browser