Alexander Ziegler

Programmierung mit Servlets und Applets in Java

Am Beispiel der grafischen Darstellung von Prozessdaten

Diplomarbeit

Diplom.de

Bibliografische Information der Deutschen Nationalbibliothek:

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

ISBN: 9783832452414

Alexander Ziegler

Programmierung mit Servlets und Applets in Java

Am Beispiel der grafischen Darstellung von Prozessdaten

Diplom.de

Alexander Ziegler

Programmierung mit Servlets und Applets in Java

Am Beispiel der grafischen Darstellung von Prozessdaten

Diplomarbeit
an der Fachhochschule Reutlingen
Fachbereich Wirtschaftsinformatik
Januar 2001 Abgabe

Diplom.de

Diplomica GmbH
Hermannstal 119k
22119 Hamburg

Fon: 040 / 655 99 20
Fax: 040 / 655 99 222

agentur@diplom.de
www.diplom.de

ID 5241

ID 5241
Ziegler, Alexander: Programmierung mit Servlets und Applets in Java: Am Beispiel der grafischen Darstellung von Prozessdaten / Alexander Ziegler - Hamburg: Diplomica GmbH, 2002
Zugl.: Reutlingen, Fachhochschule, Diplom, 2001

Diplomica GmbH
http://www.diplom.de, Hamburg 2002
Printed in Germany

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Ziel der Studienarbeit

Ziel dieser Studienarbeit war es, Erfahrungen im Umgang mit der Programmiersprache im Umfeld des Internets zu sammeln. Im Besonderen wollten wir dabei auf die Bereiche Applets, Servlets und Datenbankanbindungen mit der JDBC-API eingehen. Dabei wurden folgende Themenbereiche intensiv bearbeitet:

- Darstellung von grafischen Elementen in Applets
- Kommunikation zwischen Applets und Servlets
- Aufruf von Servlets über einen Webserver
- Funktionen einen Webservers
- Aufbau von Datenbankverbindungen
- Auslesen von Daten aus einer Datenbank
- Schreiben von Daten in eine Datenbank

Außerdem konnten wir Erfahrung in der Projektplanung sammeln, was für zukünftige Arbeit an der Hochschule und auch im Praxissemester sehr wichtig ist. Es war das erste Mal während des Studiums, dass wir ein Projekt dieser Größenordnung bearbeiten konnten. Deswegen gestaltete sich die zeitliche Planung und die Abschätzung des Aufwands etwas schwierig. Gäbe es die Möglichkeit, bereits in früheren Semestern Softwareprojekte durchzuführen, wären uns einige Schwierigkeiten erspart geblieben. Allerdings muss sich unsere Studienarbeit am Ergebnis messen lassen und das ist für ein Einsteigerprojekt meiner Meinung nach mehr als ausreichend.

2 Java – Die Programmiersprache

2.1 Geschichte von Java

Die Geschichte der Programmiersprache Java begann bereits im Jahre 1990, als ein Entwicklungsteam um James Gosling (er entwickelte den Texteditor Emacs) bei Sun Microsystems mit der Planung für eine neue Programmiersprache mit Namen **Oak** beauftragt wurde. Diese Programmiersprache wurde eigentlich nicht für den Einsatz im Internet, sondern zur Steuerung von Consumer-Electronic-Geräten, wie Fernseher und Stereo-Anlagen, aber auch Toaster oder Waschmaschinen, konzipiert. Leider setzte sich diese Sprache nie durch. Die Eigenschaften der neuen Programmiersprache (Plattformunabhängigkeit, geringer Umfang des Quellcodes) waren aber für einen anderen Einsatzzweck geradezu prädestiniert: Software für das World Wide Web.

Nachdem man die Sprache weiter verbessert und in **Java** umbenannt hatte, stellte Sun die Technologie im Jahr 1995 der Öffentlichkeit vor und entwickelte gleichzeitig einen Web-Browser (**HotJava**), der die Java-Virtual-Machine, die den Java-Bytecode interpretiert, implementiert hatte. Jetzt war es möglich, kleine Java-Programme, die sogenannten **Applets** im Browser auszuführen. Diese Mini-Applikationen ermöglichten es, die einstmals starren Web-Seiten auf der Basis von HTML lebendiger zu machen. Als dann Netscape mit dem Navigator 2.0 Ende 1995 nachzog und die JVM ebenfalls integrierte, hatte Java endgültig seinen Durchbruch.

Heute wird Java nicht nur für client-seitige Applets eingesetzt, sondern ebenso zur Datenbankanbindung und zur Programmierung in verteilten Systemen. Java ist eine vollwertige Programmiersprache, mit der sich alle Arten von Anwendungen realisieren lassen. Weil es aber sein Vorteile in Netzwerken voll ausspielt, wird es hauptsächlich dort eingesetzt. Welche Vorteile das sind und welche Merkmale Java des weiteren besitzt, darauf soll in den nächsten Abschnitten eingegangen werden.

2.2 Merkmale der Programmiersprache

2.2.1 Portabilität

Java ist plattformunabhängig. D. h. es ist möglich, Java-Code der nach den Konventionen der Programmiersprache implementiert wurde, auf beliebigen Betriebssystemen auszuführen. Dies ist eine der wichtigsten Forderungen für den Einsatz im Internet.

Dies erreicht Java auf zweierlei Arten. Zum einen wird der Java-Code auf jeder Plattform mit Hilfe der Java-Virtual-Machine (JVM) interpretiert. Ein Java-Programm wird dafür zuerst in sogenannten Bytecode compiliert und bei Ausführung von der JVM übersetzt. Zum anderen sind Datentypen in Java ebenso unabhängig von der jeweiligen Implementierung.

2.2.2 Objektorientierung

Java ist vollständig objektorientiert aufgebaut (im Gegensatz zu C++). So deckt es die Forderungen an objektorientierte Programmiersprachen, wie die Kapselung, die Vererbung, die Polymorphie und die Operatorüberladung weitestgehend ab.

2.2.3 Multithreading

Java unterstützt die gleichzeitige Verarbeitung von Prozessen. Oft ist es notwendig, verschiedene Aufgaben in einem Programm gleichzeitig auszuführen. Ein Beispiel hierfür sind Applets, die Eingaben des Benutzers am Client entgegenzunehmen, während sie auf eintreffende Nachrichten vom Server warten. Früher wurden solche Probleme mit Hilfe von Schleifen gelöst, was aber Schwierigkeiten macht, weil die Tasks nicht mehr sauber voneinander getrennt werden können. Mit sogenannten **Threads of Control** (Threads = Fäden) können diese Aufgaben effizienter und einfacher gehandhabt werden. Praktisch läuft folgendes ab: in einem Prozess können beliebige Threads gestartet werden, die keine eigenen Prozesse mehr sind, sondern Teilprozesse, die im selben Speicher arbeiten.

2.2.4 Verteilte Programmierung

Java ist für die Programmierung in verteilten Systemen entwickelt worden. Es nutzt TCP/IP als Protokoll, wobei die Kommunikation entweder über Sockets, CGI-Programme (Servlets) oder Remote Method Invocation (RMI) erfolgt.

2.2.5 Robustheit

Java ist eine sehr robuste Programmiersprache, weil bewusst auf unsichere Elemente verzichtet wurde. Im Gegensatz zu C++ verzichtet Java ganz auf den Einsatz von Pointern. Die strenge Einhaltung der Objektorientierung und die Ausklammerung von Operatorpüberladung, multipler Vererbung und automatischer Typumwandlung, tragen einen Großteil dazu bei.

2.2.6 Sicherheit

Ein mindestens ebenso wichtiger Punkt, wie die Portierbarkeit, ist die Forderung nach der Sicherheit von Java-Code. Dies gilt vor allem für Applets, die auf dem Client-Rechner ausgeführt werden. In Java wurde deshalb ein weitestgehender Schutz vor Viren und Zugriffen auf die Festplatte des Clients implementiert. Dies wird hauptsächlich durch den Einsatz der JVM erreicht. Sie unterbindet den Zugriff auf sicherheitskritische Ressourcen und führt vor dem Ablauf des Programms eine **Bytecode-Verifizierung** durch.

2.2.7 Garbage Collector

In Java gibt es nur sehr vereinzelt eine explizite Speicherfreigabe, weil sie sehr fehleranfällig ist. Java verwendet den **Garbage Collector**, der in die JVM integriert ist und sorgt während der Laufzeit dafür, dass nicht mehr benötigte Objekte aus dem Speicher gelöscht werden.

2.2.8 Unterschiede zu C++

Java ist mit Syntax und einigen Konventionen an C/C++ angelehnt. Allerdings gibt es einige wichtige Unterschiede. Java wurde auf Sicherheit und für den Einsatz im Internet konzipiert. In Java enthalten sind deshalb nicht:

- Typedefs, Defines und der Präprozessor
- Strukturen und Unions
- Funktionen außerhalb von Klassen
- Multiple Vererbung
- Goto-Befehle
- Automatische Typumwandlung
- Pointer
- Manuelle Speicherverwaltung

2.3 Das objektorientierte Konzept

2.3.1 Einführung

Das objektorientierte Programmierparadigma wurde in den späten Sechziger Jahren bzw. Anfang der Siebziger Jahre mit den Sprachen **Simula** und **Smalltalk** entwickelt. Man versprach sich damals eine einfachere Modellierung von Problemen der realen Welt in Programmen. Mittlerweile hat sich das objektorientierte Konzept durchgesetzt. Welche Gründe gab es für den enormen Aufschwung der objektorientierten Programmierung in den Neunziger Jahren?

Zu Beginn des Computerzeitalters waren die Sprachen maschinenorientiert, d.h. ganz nah an der Hardware ausgerichtet. Damit konnten die Eigenschaften der Bauteile sehr effizient genutzt werden, aber aufgrund dieser Nähe konnten keine komplexeren Programme entwickelt werden.

Anwendungsorientierte Sprachen wie Algol, Cobol oder Fortran sollten die Programmierung vereinfachen und schnellere Software-Entwicklung ermöglichen. Doch diese frühen Programme waren unstrukturiert und wenig effizient.

Bessere Ergebnisse versprach man sich von der Einführung von strukturierten Programmiersprachen wie Pascal oder C. Diese Art der Programmierung brachte echte Vorteile denn sie brachte übersichtlichere Programme und schuf die Möglichkeit, komplexere Sachverhalte zu implementieren. Leider kam man durch den prozeduralen Programmierstil dem eigentlichen Problem (dem Was) nicht näher, sondern entfernte sich immer weiter von der realen Problemlösung. Es stand das Konzept (das Wie) im Vordergrund, nicht aber die Modellierung eines Weltausschnitts.

Die objektorientierte Programmierung geht ein Stück weiter in die Richtung, die semantische Lücke zwischen Wirklichkeit und Code zu schließen. Es ist möglich ein reales Problem mit Objekten zu implementieren, die Eigenschaften und Fähigkeiten besitzen, untereinander kommunizieren und ihre Eigenheiten weitervererben können. Damit können sehr komplexe Sachverhalte abgebildet werden und die Flexibilität ist hoch genug, um mit