

Effizientere Anwendungsentwicklung mit AspectJ

Java erfreut sich einer anhaltend großen Beliebtheit, und das aus gutem Grund – kann es doch mit Plattformunabhängigkeit sowie fortschrittlichen, aber einfach anzuwendenden Konzepten überzeugen.

Als objektorientierte Programmiersprache fördert Java Wiederverwendung, die durch aspektorientierte Programmierung (AOP) noch wirksam unterstützt werden kann. Gemeinsam mit der Eclipse-Plattform bildet das AOP-Paradigma eine geradezu ideale Symbiose in Bezug auf Qualität, Effizienz und Standardisierung in der Softwareentwicklung.

Aspektorientierte Programmierung (AOP) ist ein Programmierparadigma, das die Modularisierung sogenannter Cross Cutting Concerns ermöglicht. Dabei handelt es sich um Belange eines Softwaresystems, die im gewählten Objektmodell über viele Klassen verteilt sind – so zum Beispiel Berechtigungen, Transaktionen, Logging und so fort. Genau hier greift AOP an, indem sie diese Cross-Cutting-Concerns in sogenannten Aspekten kapselt. Als AOP-Sprache für Java findet meist **AspectJ** Verwendung, das somit den **De-facto-Standard für Java** darstellt.

Modulare Architektur mit AspectJ und Eclipse

Die Eclipse-Plattform eignet sich hervorragend als Basis für modulare und flexible Systeme und bietet darüber hinaus bereits eine große Fülle an wertvollen Basisfunktionalitäten für **Rich Clients oder Web Applications**. Aus Architektur-Sicht erweist sich insbesondere die Plug-in-Architektur als vorteilhaft, die Java um ein dynamisches Komponentenmodell erweitert. Diese Modularisierung in Form von Plug-ins kann durch den Einsatz von AspectJ noch verfeinert werden. Da AspectJ ebenfalls ein Projekt der Eclipse-Community ist, existiert mit AspectJ Development Tools (AJDT) eine hervorragende **Integration in die Eclipse-Entwicklungsumgebung**.

Anwendungsbeispiel „Logging“

Zu den gängigen Anforderungen unternehmenskritischer Systeme zählt es, **Performancekennzahlen** aus dem produktiven Betrieb abzurufen. Das erreicht man, indem man an geeigneten Codestellen Logging-Anweisungen platziert. Hierbei handelt es sich um einen typischen Cross-Cutting-Concern, der hervorragend in Aspekten ausgelagert werden kann. Als positiver Effekt ist zunächst zu verbuchen, dass dabei der eigentliche Code nicht mit artfremden Anweisungen durchsetzt wird. Darüber hinaus ist es mit AOP ein Leichtes, die Logging-Anweisungen an andere Codestellen zu verschieben oder auch zu deaktivieren, **ohne dabei den eigentlichen Code ändern zu müssen**. Mit der Eclipse-Architektur

werden die Logging-Aspekte in einem eigenen Plug-in zusammengefasst und können auf diese Weise leicht vom Rest separiert werden, sowohl beim Build als auch zur Laufzeit.

Ausblick

Zusätzliche Vorteile bietet das sogenannte **Load-Time-Weaving**, bei dem die Klassen und Aspekte erst zur Laufzeit, nicht schon beim Kompilieren zusammengeführt werden. Um diese Technologie auch im Rahmen der Eclipse-Plattform bereitzustellen, sind besondere Erweiterungen erforderlich. In diesem hochaktuellen Bereich engagiert sich die **Market Unit Enterprise Architecture** mit ihrem Leistungssegment Core Technologies aktiv mit Contributions zu verschiedenen Open-Source-Projekten.

Java zählt zu den technischen Kernkompetenzen der metafinanz. Als hausinterner Innovationsmotor kann unsere Market Unit Enterprise Architecture auf diesem Gebiet auf eine über zehnjährige Erfahrung zurückblicken.

Aktuell spielen die zukunftsweisenden Themen Eclipse und AOP eine wichtige Rolle am Markt, weil sie gesteigerte Produktivität und Qualität versprechen. Davon profitieren unsere Bestandskunden, bei denen sie als strategische Plattform zum Einsatz kommen.