Orientation in Objects

Mylyn (ehem. Mylar)

Aufgaben orientiertes Arbeiten mit Eclipse

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOR

Soenke Sothmann
Orientation in Objects GmbH

Veröffentlicht am: 20.4.2007

MYLYN (EHEM. MYLAR) - AUFGABEN-ORIENTIERTES ARBEITEN MIT ECLIPSE

Moderne Entwicklungsumgebungen wie Eclipse erleichtern dem Entwickler die Arbeit, indem sie Projekte strukturiert darstellen und bei der Navigation Unterstützung bieten. Dennoch sehen sich Entwickler gerade bei großen Projekten häufig mit zu vielen Informationen konfrontiert und verbringen viel Zeit damit, die wichtigen Stellen in einem Projekt im Rahmen einer Aufgabe wiederzufinden.

Das Eclipse-Plugin Mylar hilft dabei, den Arbeitsalltag eines Entwicklers zu vereinfachen, indem es das Aufgaben-orientierte Arbeiten mit Systemen wie Bugzilla in die Entwicklungsumgebung einbindet und die für die Erledigung von Aufgaben unwichtigen Teile eines Projektes ausblendet.

Dieser Artikel beschreibt, wie Mylar den Entwickler produktiver arbeiten lässt, indem es ein Aufgaben-basiertes Arbeiten ermöglicht und verhindert, dass der Entwickler mit zu vielen Informationen konfrontiert wird.

Orientation in Objects GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.de info@oio.de

WAS IST MYLAR?

Mylar ist nicht neu, allerdings findet es erst jetzt größere Aufmerksamkeit. Auf der Eclipse Konferenz EclipseCon2007 fanden gleich eine ganze Reihe von Vorträgen zum Thema Mylar statt. Mylar entwickelt sich in der Eclipse Welt zu einem regelrechten Hype und verspricht, diese ein Stück weit zu revolutionieren. Ins Leben gerufen wurde Mylar von Mik Kersten, der neben der Arbeit an Mylar auch an AspectJ und dem AspectJ Eclipse Plugin AJDT mitarbeitet. Mylar ging aus seiner Doktorarbeit hervor.

Das Plugin Mylar steht in der Version 1.0 für Eclipse 3.2 und für die kommende Version 3.3 zur Verfügung. Mit der für Ende Juni 2007 angekündigten Version 2.0 soll sich Mylar hauptsächlich auf Eclipse 3.3 spezialisieren. Eclipse 3.2 hingegen wird nur noch mit Mylar Bugfixes versorgt werden. Generell setzt Mylar Java 5 oder höher voraus.

Wie in Abb. 1 dargestellt, verwendet Mylar sog. Bridges (Brücken), um sich an verschiedenen Punkten in die Eclipse IDE einzuklinken. Die Entwicklung eigener Brücken, um Mylar in Plugins einzubinden, ist über das Bridge API möglich. Für die Anbindung an Aufgaben-Verwaltungswerkzeuge werden Connectoren verwendet. Auch hier können eigene Connectoren über das Connector API entwickelt werden.

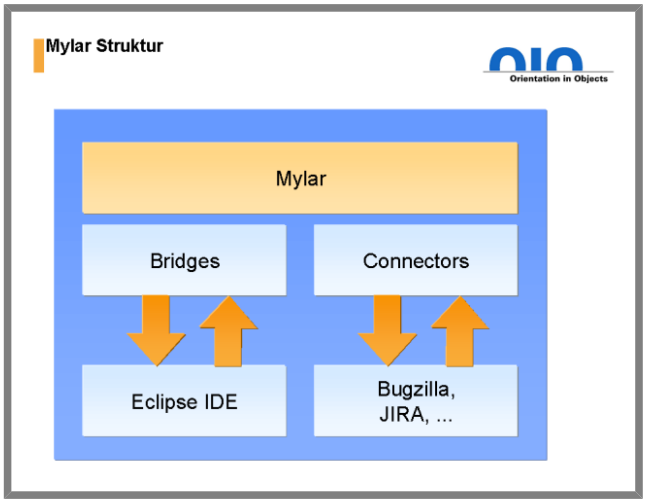

Abbildung 1: Mylar Struktur

EINBETTUNG VON AUFGABEN IN ECLIPSE

Entwicklungsprozesse schließen meist Aufgaben-orientiertes Arbeiten ein. Aufgaben umfassen dabei das Implementieren neuer Funktionen oder das Beheben von Fehlern, aber auch Dokumentation und Refactoring gehören dazu. Unterstützt werden Projektteams dabei meist durch Bug-Tracking und Issue-Management Werkzeuge wie Bugzilla oder JIRA.

Diese Vorgehensweise wird durch Entwicklungsumgebungen in der Regel nicht widergespiegelt. Entwickler müssen oft ihre IDE verlassen, um mit Werkzeugen zu arbeiten, mit denen die Aufgaben verwaltet werden können.

Mit dem Eclipse Plugin Mylar ist es möglich, Aufgaben in die Entwicklungsumgebung zu integrieren. Dabei werden die Tools Bugzilla, JIRA und Trac durch Connectoren unterstützt. Weitere Connectoren, wie z.B. für die Anbindung des Projekt-Management-Werkzeugs XPlanner, sind in Arbeit. Existiert kein passender Connector für ein Tool, so kann auf einen generischen Web-Connector zurückgegriffen werden, der jedes Task Management Werkzeug unterstützt, solange es über eine Weboberfläche gesteuert werden kann. Wird einem Entwickler eine Aufgabe zugewiesen, so wird er in seiner Entwicklungsumgebung durch ein Popup benachrichtigt. Auch können neue Aufgaben direkt in der IDE erstellt und bearbeitet werden. Dabei ist der Entwickler nicht auf externe Tools wie Bugzilla angewiesen, er kann auch lokale Aufgaben definieren. Das bietet sich vor allem an, wenn der Entwickler sich selbst eine Aufgabe setzt, wie z.B. einen Code-Abschnitt noch einmal zu überarbeiten.

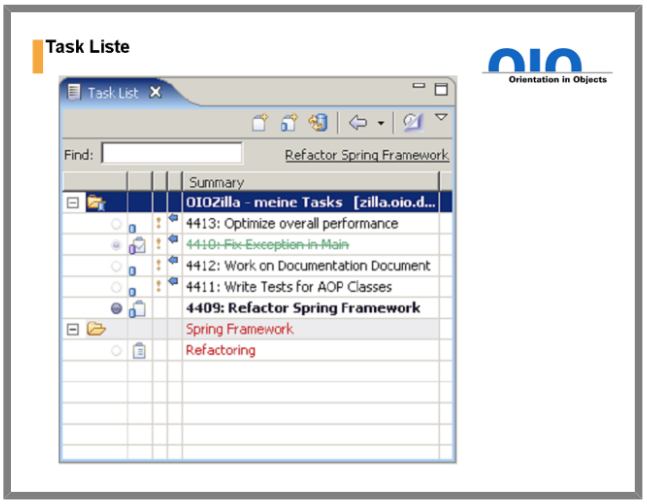

Abbildung 2: Task Liste

Eine Erleichterung stellt dies für den Entwickler dar, da er stets eine Liste von anstehenden Aufgaben, wie in Abb. 2 dargestellt, auf einen Blick hat. Die ihm zugewiesenen Aufgaben kann der Entwickler zeitlich planen. Um nicht mit zu vielen Aufgaben auf einmal überflutet zu werden, die bereits im Voraus von der Projektleitung definiert wurden, kann die Aufgabenliste so gefiltert werden, dass nur Aufgaben angezeigt werden, die der Entwickler für die aktuelle Arbeitswoche geplant hat.

Arbeitet man an einer Aufgabe, so lässt Mylar eine Stoppuhr mitlaufen. Der Entwickler erhält so die Möglichkeit, einfache Reports, wie in Abb. 3 dargestellt, zu Erzeugen und eine Übersicht zu erhalten, wie nahe er an einer vorher veranschlagten Entwicklungszeit ist, oder ob er dem Zeitplan hinterher hinkt.

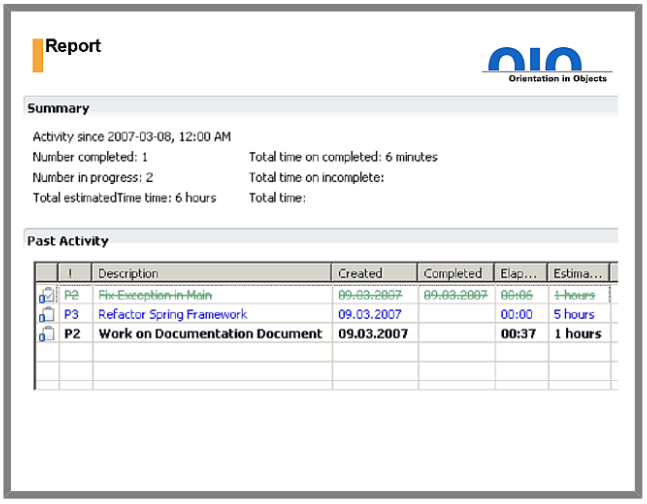

Abbildung 3: Report

Häufig werden in den Beschreibungen von gemeldeten Bugs Stack-Traces hinterlegt, mit denen der aufgetretene Fehler dokumentiert wird. Hier offenbart sich ein weiterer Vorteil der in die IDE eingebundenen Aufgaben: Wie in Abb. 4 zu sehen, sind Stack-Traces in Aufgaben-Beschreibungen Eclipse-typisch mit dem Quellcode verlinkt und die Fehler verursachende Codestelle kann sofort angesprungen werden.

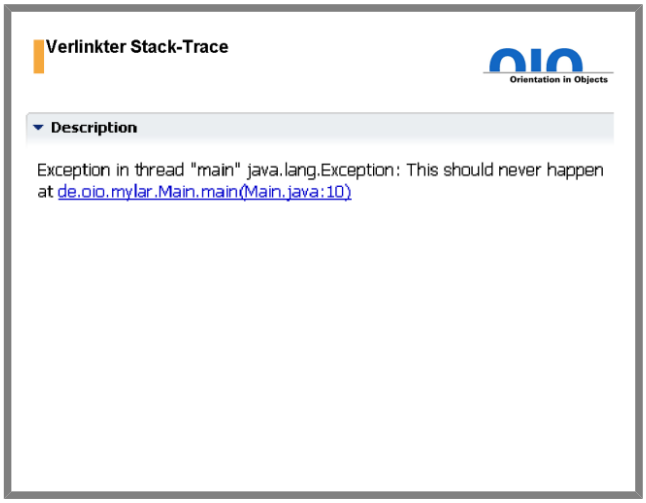

Abbildung 4: Verlinkter Stack-Trace

KONTEXT BEZOGENES ARBEITEN

In umfangreicheren Projekten häufen sich schnell große Mengen von Ressourcen wie Quell- und Konfigurationsdateien an. Um dieser Ressourcenmenge Herr werden zu können, bietet Eclipse mit Package-Explorer und Outline Strukturierungswerkzeuge an. Dennoch ist der Bildschirm des Entwicklers oft mit Informationen, wie in Abb. 5 dargestellt, überladen und er verbringt viel Zeit mit der Navigation im Projekt und dem Finden der richtigen Codeabschnitte zum Erledigen seiner Aufgaben.

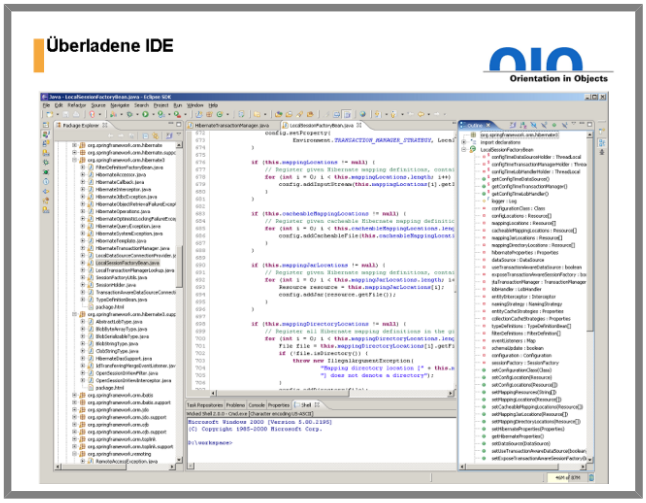

Abbildung 5: Überladene IDE

Ein weiteres Problem entsteht, wenn der Entwickler zwischen der Bearbeitung von Aufgaben wechselt. Wird zum Beispiel eine Funktionalität implementiert und zwischenzeitlich ein Fehler behoben, müssen nach dem Bugfix erneut die interessanten Codestellen der Implementierung gesucht werden.

Um diesem Problem entgegenzuwirken, führt Mylar Aufgaben-Kontexte ein: Projektteile, mit denen häufig gearbeitet wird, werden optisch hervorgehoben und uninteressante Teile können ausgeblendet werden. Dabei beeinflusst Mylar sowohl die Ansicht des in Abb. 6 dargestellten Package Explorers als auch die der Outline und des Navigators. Darüber hinaus kann auch der Quellcode selbst entsprechend aufbereitet werden: unwichtige Codeblöcke werden eingeklappt dargestellt, wichtige ausgeklappt. Mylar baut, während der Entwickler an einer Aufgabe arbeitet, ein Modell auf, das den Grad des Interesses an Ressourcen des Projekts widerspiegelt. Dabei werden nicht nur geänderte Ressourcen als interessant erachtet, sondern auch solche, die oft geöffnet werden. Auch das Öffnen einer Dokumentation führt somit dazu, dass diese im Kontext einer Aufgabe als interessant eingestuft wird. Das Interessensmodell kann bei Bedarf auch manuell durch den Benutzer angepasst werden. So lassen sich fälschlicherweise als wichtig eingestufte Ressourcen aus dem Modell entfernen und noch nicht aufgenommene Ressourcen als wichtig markieren.

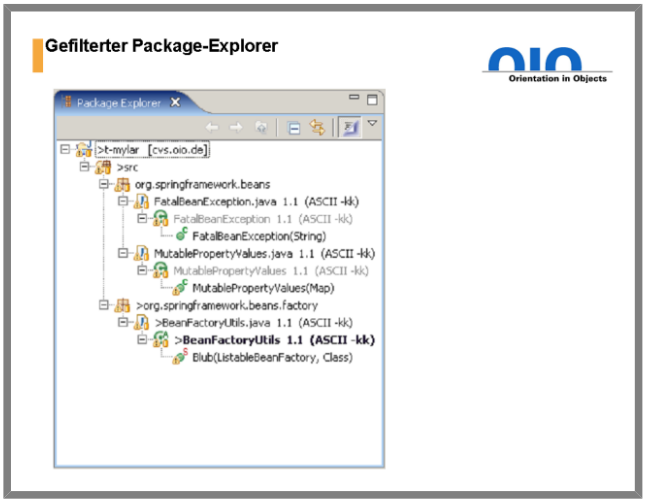

Abbildung 6: Gefilterter Package-Explorer

Neben dem Interessensmodell wird auch der Zustand der IDE im Kontext gespeichert. Dies schließt geöffnete Editor-Fenster und die aktive Perspektive ein. Beim Wechsel zwischen Aufgaben wird der zugehörige Kontext wiederhergestellt. Multitasking wird so deutlich erleichtert, da mit einem Klick der Kontext einer Aufgabe gewechselt werden kann.

Informationen, welche Projektteile beim Beheben eines Fehlers oder beim Implementieren einer Funktionalität wichtig waren, bleiben nachhaltig bestehen. Diese Kontexte können auch dem übrigen Team zur Verfügung gestellt werden, indem der Kontext als Attachment einer Aufgabe angehängt wird. Wurde bei der Implementierung ein solcher Kontext einer Aufgabe angehängt, kann im Falle eines Fehlers der Entwickler, dem der Fehler zugewiesen wird, den Kontext der Implementierung wiederherstellen und so die relevanten Codestellen, die den Fehler beinhalten könnten, direkt auf den Bildschirm bekommen. Um Kontexte mit dem Team zu teilen, muss das eingesetzte Werkzeug, wie z.B. Bugzilla, Dateianhänge unterstützen.

Der Umgang mit den Versionsverwaltungswerkzeugen CVS und Subversion (über das Subclipse oder Subversive Plugin) kann durch Mylar ebenfalls vereinfacht werden. Hat der Entwickler an verschiedenen Aufgaben gearbeitet und will nun die Änderungen einchecken, so kann er beim Synchronisieren die geänderten Dateien nach Aufgaben, wie in Abb. 7 zu sehen, strukturieren lassen und diese dann getrennt committen. Darüber hinaus können auch die Commit Messages generiert werden. Dabei werden die Informationen und der Status der Aufgabe verwendet und so Commit Messages im Stile von "RESOLVED - bug 4410: Fix Exception in Main" generiert.

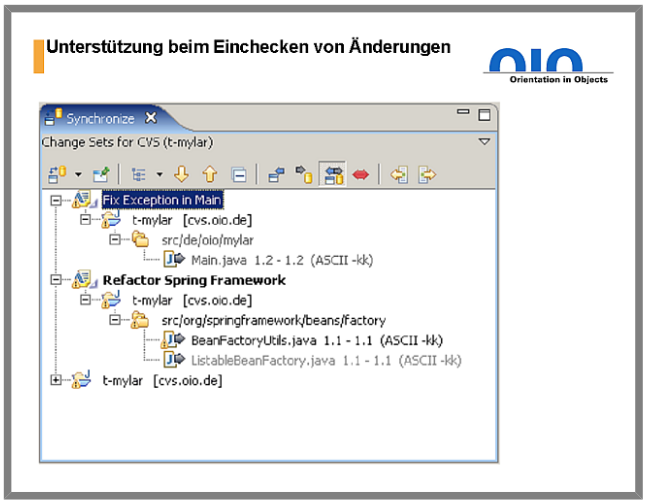

Abbildung 7: Unterstützung beim Einchecken von Änderungen

Ein weiteres nützliches Feature im Rahmen einer Aufgabe sind Kontext-bezogene Tests. Dabei handelt es sich um eine von Mylar automatisch bei der Arbeit angepasste Test-Suite, die alle JUnit-Tests beinhaltet, die im Kontext einer Aufgabe vorhanden sind. Der Entwickler kann durch Starten der Test Suite relevante Tests durchführen, ohne alle Tests des Projekts anstoßen zu müssen und kann so Zeit einsparen.

SCHATTENSEITEN?

Mylar 1.0 wirkt durchdacht und stabil. Hier und da gibt es noch kleinere Anzeigeprobleme im Zusammenspiel mit der Eclipse IDE. Dies äußert sich im Einzelfall durch nicht automatisch aktualisierte Views. Ein manueller Refresh schafft hier Abhilfe. Einige Dialoge sind noch nicht ganz intuitiv gestaltet. So muss beispielsweise beim Synchronisieren mit CVS erst auf ausgehende Änderungen umgeschaltet werden, um nach Aufgaben gliedern zu können. Im Ganzen wirkt die Bedienung von Mylar aber durchdacht und es ist nur eine kurze Eingewöhnungszeit notwendig.

FAZIT

Mylar beschleunigt die Arbeit des Entwicklers und hilft, den Fokus auf kontextbezogene Projektteile zu behalten. Der Entwickler wird nicht länger mit Informationen überflutet und behält stets den Fokus auf die für eine Aufgabe relevanten Ressourcen. Multitasking im Sinne des Wechsels zwischen Aufgaben wird erheblich erleichtert, da beim Wechseln zwischen den Aufgaben der Zustand der IDE wiederhergestellt wird. Durch das Teilen von Kontexten wird die Teamarbeit erleichtert.

Ob Mylar tatschlich auf breiter Basis Einzug in die Entwicklungsprozesse der Unternehmen halten wird, bleibt abzuwarten. Man darf gespannt sein, in welche Richtung sich das Plugin mit der kommenden Version 2.0 entwickeln wird. Nach einer kurzen Eingewöhnungsphase werden viele Entwickler auf dieses Plugin nicht mehr verzichten wollen.

REFERENZEN

- Mylar Website
  (http://www.eclipse.org/mylar/)