Eine Einführung in Maven Continuum

Kontinuierlich zum Ziel

VON HALIL-CEM GÜRSOY

„Continuous Integration“ wird in der Java-Welt häufig nur mit CruiseControl in Verbindung gebracht. Schon fast unbemerkt haben in der Zwischenzeit die Maven-Entwickler ein eigenes Continuous Integration-Werkzeug implementiert: Maven Continuum. Nun ist es an der Zeit, einen etwas genaueren Blick auf das schon nicht mehr ganz so neue Produkt aus der Maven-Schmiede zu werfen. Wird Continuum den Ansprüchen, die mit dem Namen Maven verbunden sind, gerecht und kann es sich gegen den Hauptkonkurrenten CruiseControl absetzen?

Was verbirgt sich eigentlich hinter der mysteriösen Wolke „Continuous Integration“ (CI)? Das Thema wurde vor einigen Jahren schon von Kent Beck im Rahmen seiner Arbeiten zu Extreme Programming als eines der Pfeiler der XP beschrieben (z.B. in [1]). Letztendlich wurde durch Martin Fowler in seinem Weg ebnenden Artikel „Continuous Integration“ [2], in dem er auch die praktische Machbarkeit demonstrierte, die Thematik in die breite Masse getragen.

„Je öfter, umso besser“

Der Grundgedanke, der CI durchsetzt, lautet „Je öfter, umso besser“. Gemeint sind der Build eines Projektes aus den aktuellen Quelltexten sowie das Deployen und Testen dieses Builds in einer Integrationsumgebung. Der Hintergrund ist folgender: Je später solche Tests durchgeführt werden, umso größer werden die Aufwände, um möglicherweise vorhandene Fehler zu korrigieren, da diese womöglich erst durch die Interaktion verschiedener Komponenten zutage treten. Ziel ist es also, so schnell wie möglich vorhandene Fehler aufzudecken und zu korrigieren. Optimal wäre dabei, dass jedes Commit im Versionsmanagementsystem zu einem Build mit anschließendem Test führt. Dies ist natürlich in der realen Projektwelt durch die Laufzeit eines Builds limitiert. Weiterhin stützt sich CI neben dem „Je öfter, umso besser“ darauf, dass nur eine Quelltextbasis existiert (z.B. ein Versionsmanagementsystem) und dass die Builds, das Deployment und die Tests automatisiert erfolgen.

Der Platzhirsch CruiseControl und weitere Konkurrenten

Continuum muss sich seit dem Erscheinen der ersten verfügbaren Version in einem Markt behaupten, in dem schon seit einigen Jahren diverse Produkte zu diesem Themenfeld existieren. Die bekanntesten frei verfügbaren Produkte möchte ich hier kurz aufführen, wobei es eine Vielzahl an weiteren Produkten gibt.

Martin Fowler publizierte mit seinem oben erwähnten Artikel eine Webapplikation, die automatisiert CI unterstützen sollte. Heute ist diese Webapplikation unter ihrem Namen „CruiseControl“ (CC) [3] sehr bekannt und steht schon fast synonym für Continuous Integration. Die Konfiguration von CC und das Einbinden der Projekte (es werden sowohl Ant als auch Maven/Maven 2-Projekte unterstützt) ist umfangreich und erfolgt in einer zentralen XML-Datei, wobei allerdings erwähnt werden sollte, dass die Dokumentation zur Konfiguration auch ihren Namen wert ist. Inzwischen gibt es aber auch eine Konfigurations-GUI, um die Konfiguration etwas zu erleichtern.

Nach dem Installations- und Konfigurationsmarathon entlohnt CC mit einem recht großen Funktionsumfang die Mühen. Erwähnt werden sollte die sehr breite Palette der unterstützen SCM-Systeme: neben den obligatorischen Open-Source-Systemen CVS und Subversion werden u.a. Harvest, Perforce, Visual SourceSafe und auch Exoten wie SnapshotCM [4] unterstützt. Weiterhin besticht CC durch die Vielzahl an Möglichkeiten, Entwickler über einen aktuellen Build zu informieren. Zur Verfügung steht neben Mail und dem XMPP-Protokoll z.B. auch Lotus Sametime. CruiseControl kann parallel mehrere Projekte gleichzeitig bauen und testen. Es kann für jeden Build im SCM-System automatisch ein Label gesetzt werden, wobei es möglich ist, auch später auf alle erzeugten Artefakte und Testberichte eines Builds nachträglich zuzugreifen. Übrigens gibt es CruiseControl inzwischen nicht nur für die Java-Welt sondern auch für .NET.

Anthill OS [5] ist ein weiteres Produkt aus dem CI-Spektrum und fällt in erster Linie durch die Namensgebung auf, die eine Nähe zum Apache Ant-Projekt vermuten lässt. Allerdings hat das Tool nichts mit dem Ant-Projekt oder der ASF an sich zu tun, sondern wird von einem Unternehmen namens Urbancode betreut. Neben der freien „OS“-Version von Anthill existiert eine kommerzielle Version „Anthill Pro3“. Hier fällt auf, dass die Featurematrix der freien Version gegenüber der kommerziellen Version deutlich abgespeckt ist. Für Open Source-Projekte bietet Urbancode jedoch eine freie Lizenz an.

Zuletzt sei noch Luntbuild [6] erwähnt. Luntbuild besticht nach der Installation zwar nicht unbedingt durch eine penibel durchdachte Struktur der Webseiten, es ist aber dennoch im Gegensatz zu den bereits erwähnten Systemen mög-

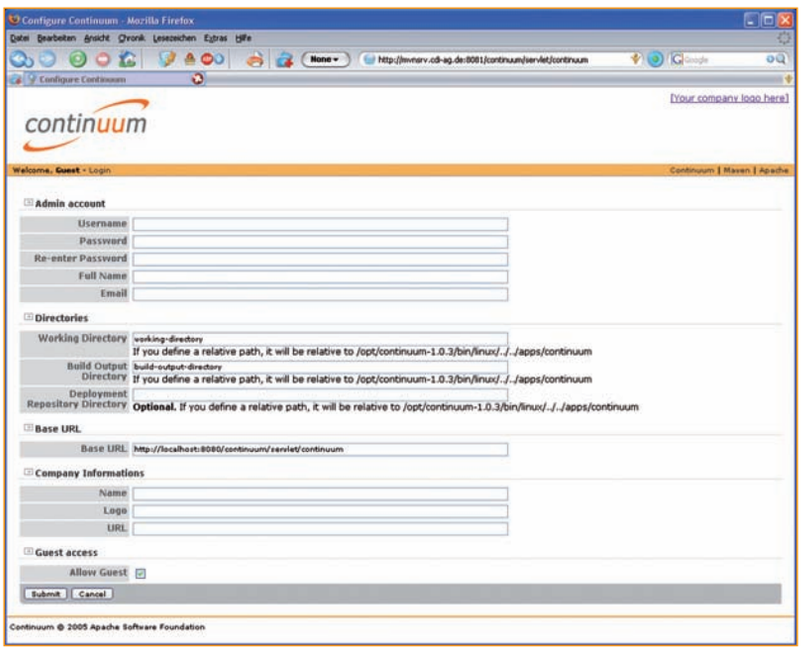

Abb. 1: Konfigurationsseite von Continuum

lich, das System über die Weboberfläche zu konfigurieren. Auch Luntbuild verfügt über ein beachtliches Spektrum an unterstützten SCM-Systemen und Build-Tools, wobei hier neben den üblichen Verdächtigen auch „rake", ein Build-Tool für Ruby-Projekte [7], direkt unterstützt wird.

Warum ein weiteres CI-Tool?

Maven Continuum wurde aufgrund des Gedankens geboren, ein CI-Tool zur Verfügung zu stellen, mit dem soweit wie möglich die beste Unterstützung für Maven-Projekte gegeben wird (s. Kurzinterview mit Brett Porter). Im April 2005 wurde die erste Alpha-Version von Continuum 1.0 veröffentlicht, die finale Version 1.0 folgte im Oktober 2005.

Continuum baut, wie auch Maven 2, auf Plexus [8] auf. Bei Plexus handelt es sich um ein Application Framework ähnlich Spring, das die Möglichkeit bietet, auf eine elegante Art und Weise sauber gekapselte Komponenten zu erstellen, die einen hohen Grad an Wiederverwendbarkeit ermöglichen. So konnten viele Komponenten von Maven 2 direkt durch Continuum verwendet werden. Zudem kommt Continuum gebündelt mit dem Servlet Container Jetty, der wiederum als ein Plexus-Service im Gesamtkontext eingebunden ist.

Installation und erste Schritte

Nach dem Download des Continuum-Pakets (die aktuelle Version 1.0.3 ist zu finden unter [9]) und Entpacken des Archivs in ein Verzeichnis kann Continuum sofort gestartet werden. Dies geschieht mithilfe des für das verwendete Betriebssystem vorgesehenen Startskriptes, das sich jeweils in einem der verschiedenen Unterverzeichnisse unter *continuum-1.0.3/bin* befindet.

```
[hgu@mvnsrv] linux]$ ./run.sh start
Starting continuum...
[hgu@mvnsrv] linux]$ ./run.sh status
Continuum is running (2685)
```

Voraussetzung ist nur ein aktuelles Java auf dem System (die Umgebungsvariable JAVA_HOME sollte gesetzt sein). Für Windows wird übrigens auch gleich ein Skript mitgeliefert, um Continuum als Windows Service einzurichten. Der erste Start dauert etwas länger, da die integrierte Derby-Datenbank erst einmal eingerichtet werden muss. Die nachfolgenden Starts gehen dann wesentlich schneller vonstatten.

Per Default ist die Weboberfläche von Continuum auf dem Port 8080 (*http://localhost:8080/continuum*) zu erreichen. Ist dieser Port belegt oder wünschen Sie aus anderen Gründen den Port zu wechseln, müssen Sie in der Konfigurationsdatei *apps/continuum/conf/application.xml* die Konfiguration des Jetty Services anpassen:

```
<listeners>
 <http-listener>
  <port>8080</port>
 </http-listener>
</listeners>
```

Beim ersten Zugriff auf die Webapplikation wird der Benutzer mit einer initialen Konfigurationsseite begrüßt. Hier können die Benutzerdaten des Administrators, die verschiedenen benötigten Verzeichnisse sowie Informationen zu dem Unternehmen eingepflegt werden. Mit dem Absenden der eingetragenen Daten, der anschließenden Neuanmeldung als Administrator und darauf folgend gegebenenfalls der Einrichtung von weiteren Benutzern über die Weboberfläche ist die Grundkonfiguration schon erledigt. Das System ist somit bereit für die ersten Projekte.

Soll ein Maven2-Projekt aufgenommen werden, gestaltet sich dies sehr einfach. Hierzu wird in der linken Navigati-

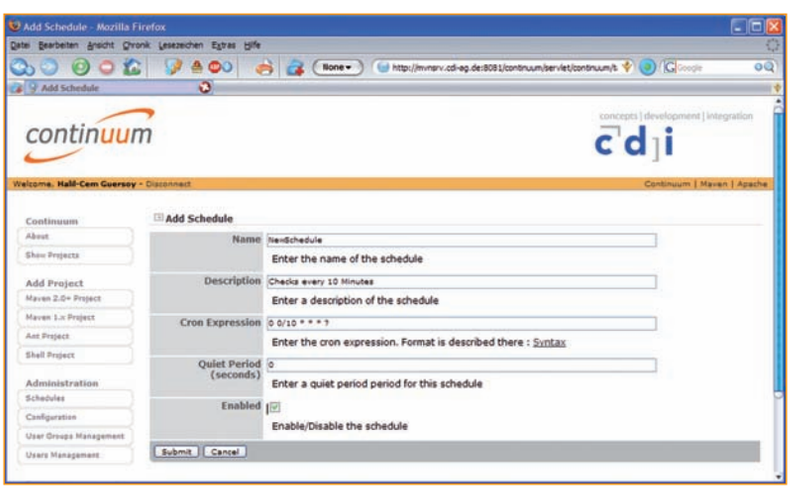

Abb.2: Mit dem Cron-Ausdruck 0 0/10 * * * ? startet der Scheduler alle 10 Minuten.

on einfach Add Project | Maven 2.0.x Project ausgewählt. In der nun erscheinenden Maske existiert die Wahlmöglichkeit, ob nun die von Continuum benötigte POM-Datei hochgeladen oder der URL der Datei eingegeben werden soll. Wichtig ist nur, dass in der POM selbst die Informationen über das Versionierungssystem im Abschnitt SCM enthalten sind, da ansonsten das Projekt abgewiesen wird.

```
<scm>
 <connection>scm:svn:svn://guersoy/maven2/
                        proficio-client/trunk</connection>
 <developerConnection>scm:svn:svn://guersoy/maven2/
                                    proficio-client/trunk
 </developerConnection>
</scm>
```

Da nun ein CI-System verwendet wird, sollte dies auch in der POM im Abschnitt *ciManagement* kundgetan werden. Dieser Eintrag wird im Rahmen der Site-Generierung mit in die Projektdokumentation eingebunden, hat aber ansonsten keinen weiteren Effekt.

```
<ciManagement>
        <system>Continuum</system>
        <url>http://localhost:8080/continuum</url>
</ciManagement>
```

Ist das Projekt in Continuum aufgenommen, können weitere Konfigurationen vorgenommen werden. Continuum überprüft regelmäßig, ob Änderungen an den Sourcen im System vorliegen. Die Zeitsteuerung wird über die so genannten Scheduler geregelt, die mithilfe von Cron-Ausdrücken konfiguriert werden. Per Default ist der DEFAULT_SCHEDULER eingerichtet, der stündlich startet. Diese Scheduler werden den Build Definitions zugewiesen. Ein neu importiertes Projekt bekommt eine Build-Definition zugewiesen, die die Goals *clean install* mithilfe des DEFAULT_SCHEDULER stündlich ausführt, falls Änderungen auftreten. Sollen die Builds in kürzeren Intervallen erfolgen, kann ein neuer Scheduler einfach über Administration | Schedules eingerichtet oder der vorhandene DEFAULT_SCHEDULER angepasst werden.

Es können auch mehrere Build-Definitionen einem Projekt zugewiesen werden, um z.B. neben den eigentlichen Test-Builds täglich einmal einen Site-Build durchzuführen.

Benachrichtigungen einfach gemacht

Nun möchten vor allem Verantwortliche (und manchmal auch Entwickler) gerne darüber informiert werden, ob die Builds und die Tests erfolgreich durchgeführt worden sind. Hierzu können „Notifier“ eingerichtet werden. Continuum unterstützt zurzeit Benachrichtigungen über Mail, IRC, MSN Messanger und dem Jabber Protokoll.

Die Einrichtung eines Notifier erfolgt recht einfach. Für einen Jabber Notifier z.B. werden nur die Daten des Jabber-Servers und die entsprechenden Daten des Senders und Empfängers benötigt. Es kann ausgewählt werden, wann eine Benachrichtigung erfolgen soll, z.B. nur dann, wenn ein Fehler auftritt. In der Grundeinstellung versendet Continuum keine weiteren Nachrichten mehr, wenn der Build-Status eines Projektes

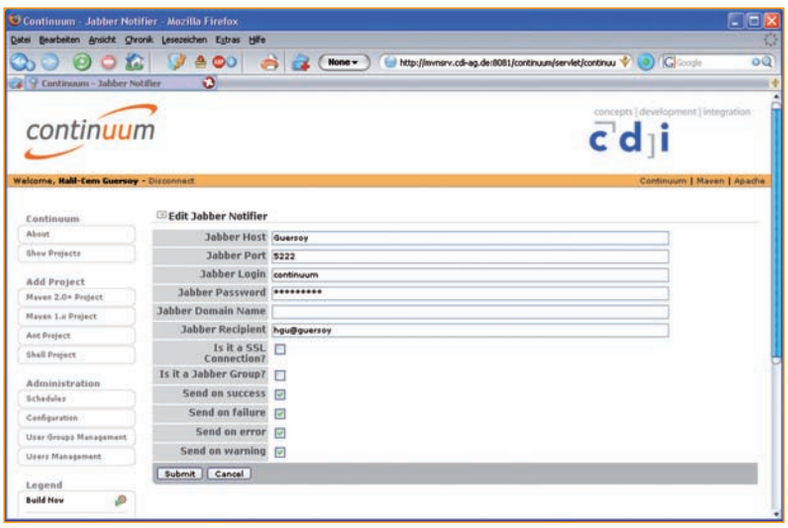

Abb. 3: Konfiguration eines Notifier

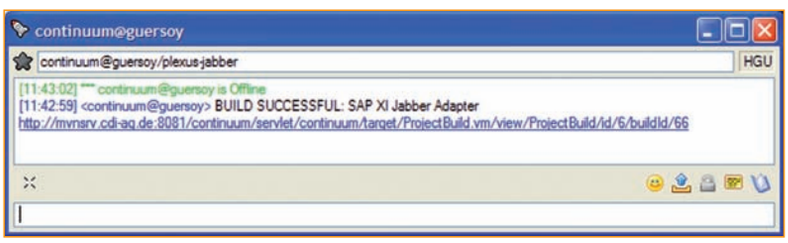

Abb. 4: Ist ein Notifier richtig konfiguriert, wird über die aktuellen Builds informiert.

sich nicht ändert (z.B. alle nachfolgenden Builds sind ebenfalls erfolgreich). Soll die Benachrichtigung tatsächlich immer unabhängig von einer Statusänderung bei allen Builds erfolgen, so muss dies über die Konfigurationsdatei *apps/continuum/conf/application.xml* eingestellt werden:

```
<configuration>
 <alwaysSend>true</alwaysSend>
</configuration>
```

Möchten Sie E-Mails zur Benachrichtigung versenden und der SMTP-Server befindet sich nicht auf dem gleichen System wie Continuum, dann muss ebenfalls die Datei *apps/continuum/conf/application.xml* wie folgt bearbeitet werden:

```
<configuration>
 <smtp-host>mailsrv</smtp-host>
 <smtp-port>25</smtp-port>
 <sslProvider>com.sun.net.ssl.internal.ssl.Provider
                                    </sslProvider>
</configuration>
```

Das Einbinden von Ant-Projekten gestaltet sich ebenfalls sehr einfach. Hier müssen allerdings die Informationen zu dem Versionierungssystem per Hand in der Maven SCM-Syntax eingegeben werden, da in einem Ant-Skript keine standardisierten Informationen hierzu vorgesehen sind. Bei einem Ant-Projekt geht Continuum davon aus, dass das Ant-Skript *build.xml* lautet und das Default-Target aufgerufen werden muss. Ist dies nicht der Fall, können diese Einstellungen in den Build Definitions analog der Konfiguration eines Maven 2-Projektes nachträglich angepasst werden.

Auf der Überholspur?

Maven Continuum besticht durch seine Einfachheit und sehr schnelle Konfiguration. Theoretisch ist ein System in fünf Minuten aufgesetzt und in der Lage, Projekte aufzunehmen. Ein Projekt selbst ist ebenfalls in maximal weiteren fünf Minuten eingebunden. Das System ist stabil und schnell, wobei aber im Gegensatz zu CruiseControl keine parallelen Builds stattfinden können. Die Builds kommen vielmehr in eine Warteschlange. Dies kann, wenn mehrere größere Projekte mit längeren Build- und Testzeiten über eine Continuum-Instanz verwaltet werden, sehr schnell zu Verzögerungen führen.

Ein Kurzinterview mit Brett Porter

Brett Porter ist seit dem Jahr 2003 im Maven-Projekt involviert. Als Mitglied des Maven-Projekt-Management-Komitees ist er verantwortlich für das Project Release Management und zudem aktives Mitglied der Apache Software Foundation seit 2004. Er ist einer der vier offiziellen Continuum Commiter.

JM: Seit wann arbeitest Du an Continuum mit und was waren Deine Gründe für die Mitarbeit?
BP: Ich bin seit 2005 in dem Projekt aktiv. Der Hauptgrund war, dass ich einfach Interesse hatte, an Continuum mitzuarbeiten.

JM: Leider vermisst man auf den Maven Continuum-Seiten so etwas wie eine Projekthistorie. Was war der Grund für den Start des Projektes?
BP: Die Entwicklung an Continuum hatte bereits begonnen, als ich zu dem Projekt stieß. Aber es ist schon immer mit dem Ziel geplant und entwickelt worden, die bestmögliche Continuous Integration-Lösung für Maven-Benutzer zu sein.

JM: Als das Continuum-Projekt gestartet wurde, existierten bereits einige interessante freie Continuous Integrations-Werkzeuge, wie z.B. CruiseControl. Was waren die Pläne zur Positionierung von Continuum in diesem Umfeld?
BP: Vor dem Projektstart hatte es den Anschein, dass die Entwicklung von CruiseControl sich verlangsamen würde. Die meisten anderen Produkte waren kommerziell. Continuum zielte daher darauf, besser mit Maven zu interagieren und eine einfachere Bedienung zu ermöglichen.

JM: Es hat den Anschein, dass für Continuum von Eurer Seite aus keine „Werbung" gemacht wird. Gibt es Pläne, Continuum wie Maven stärker zu fördern?
BP: Wir machen für keines der Produkte „richtige" Werbung. Vielmehr wird dies aus der Community getrieben. Je größer die Community wird, umso größer wird auch die Sichtbarkeit von Continuum werden.

JM: Was sind für dich die Vorteile von Continuum gegenüber CruiseControl – und die Nachteile?
BP: Ich habe schon seit einiger Zeit CruiseControl nicht mehr verwendet, aber ich denke, Continuum ist wesentlich einfacher zu konfigurieren. Dafür ist CruiseControl das reifere Produkt mit mehr Funktionalität, z.B. bei der Unterstützung von weiteren Versionsmanagementsystemen.

Positiv fällt die Benutzerverwaltung von Continuum auf. Hier kann ein Administrator gezielt Benutzern bestimmte Rechte einräumen bzw. entziehen, wie z.B. das Anlegen von neuen Projekten oder Starten von Builds.

Leider werden Unit-Testergebnisse oder Metriken (wie z.B. von PMD erzeugt) nicht wie in CruiseControl direkt in die Detailansichten eines Builds integriert. Diese müssen umständlich dem Konsolenauszug der Builds entnommen werden. Eine Alternative bei Maven 2-Projekten besteht darin, diese Tests im Rahmen des Site-Builds vorzunehmen und die erzeugten Berichte in die Projektseiten zu integrieren. Leider liegen dann die Daten zu älteren Builds aber nicht mehr vor.

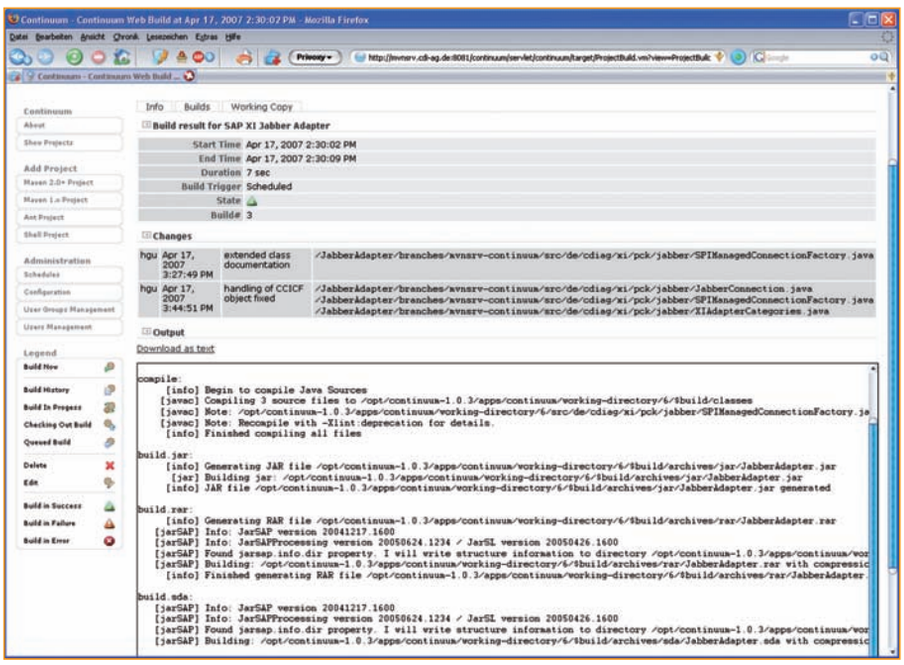

Abb. 5: Ist ein Build gelaufen, gibt die Detailseite Auskunft über den Vorgang.

Kontinuität mit Grenzen

Mit der Zeit werden einem Benutzer schnell auch weitere Grenzen von Continuum aufgezeigt. Ich vermisse z.B. die Möglichkeit von CruiseControl, schnell auf die Artefakte aller vergangenen Builds nachträglich zugreifen zu können. Auch auf die Quelltexte eines alten Builds kann nicht ohne Weiteres zugegriffen werden, da Label im Versionierungssystem nicht automatisch gesetzt werden. Selbstverständlich könnte dies alles auch in Continuum im Rahmen der Build Definitions nachgebildet werden, z.B. mit dem Maven-Plugin *release* bei einem Maven 2-Projekt. Aber dennoch ist hier leider deutlich ein Bruch zu erkennen und in diesem Bereich hat CruiseControl noch einen gewissen Vorsprung. Zwar ist in der kommenden Release 1.1 die Verwendung des Maven-Plugins *release* in Continuum integriert, allerdings fungiert dabei Continuum eher als eine GUI: Hier fehlt noch die Automatisierung.

Und die Zukunft?

Neben vielen Bugfixes weist die kommende Version 1.1, die noch in diesem Jahr erscheinen soll, ein wesentlich erweitertes Autorisierungskonzept auf, in dem rollenbasiert Rechte an Benutzer vergeben werden können. Geplant ist auch, die Liste der unterstützten Notifier und Versionsierungssysteme zu erweitern, u.a. für PVCS, Visual SourceSafe und andere. Intern soll in den kommenden Versionen die Webapplikation auf WebWorks bzw. Struts 2 aufbauen. Hierfür wurde diese vollständig umgeschrieben. Eine komplette Liste der anstehenden Änderungen ist über den Issue Tracker von Continuum zu erhalten [10].

Einen Vorgeschmack auf die kommende Version kann der geneigte Leser recht einfach bekommen. Die Sourcen stehen im Subversion-Repisotory der ASF unter http://svn.apache.org/viewcvs.cgi/maven/continuum/trunk/ zur Verfügung und können schnell gebaut werden. Voraussetzung ist, dass Maven 2 installiert ist und einige Bibliotheken in das lokale Maven Repository geladen wurden (s. hierzu die beiliegende README.txt). Diese Bibliotheken sind aus lizenzrechtlichen Gründen nicht im Subversion Repository des Projektes sowie den öffentlichen Maven Repositorys enthalten.

Lohnt sich der Einsatz?

Der Einsatz von Continuum lohnt sich in erster Linie für Maven 2-Projekte, die noch kein anderes CI-System einsetzen. Ist allerdings mehr Funktionalität gewünscht und ist man gewillt, diese Funktionalität mit einem wesentlich höheren Aufwand für Installation und Konfiguration zu erkaufen, ist ein Projektteam zur Zeit noch mit CruiseControl besser bedient. Ein Umstieg von einem bereits laufenden CruiseControl lohnt sich aufgrund vieler fehlender Funktionen bisher nur bedingt. Allerdings schließt die bald erscheinende neue Version einige der Lücken, die Continuum von CruiseControl und den anderen Produkten in diesem Umfeld trennt. Wir können also gespannt in die Zukunft blicken.

Dr. Halil-Cem Gürsoy ist Senior Consultant bei CDI Concepts Development Integration und kann auf eine mehrjährige Erfahrung in Java EE-Entwicklung und -Architektur auf unterschiedlichen Plattformen zurückblicken. Sein aktueller Fokus liegt auf der Planung und Implementierung von Event-getriebenen Architekturen auf der Basis von SOA und der Erarbeitung von Konzepten zur Migration von Java EE-Applikationen.

Links & Literatur

[1] Kent Beck, Cynthia Andres: Extreme Programming Explained, Addison-Wesley Longman, Amsterdam (2004)

[2] www.martinfowler.com/articles/originalContinuousIntegration.html

[3] cruisecontrol.sourceforge.net/

[4] www.truebluesoftware.com/

[5] www.anthillpro.com/html/products/anthillos/default.html

[6] luntbuild.javaforge.com/

[7] rake.rubyforge.org/

[8] plexus.codehaus.org/

[9] maven.apache.org/continuum/download.html

[10] jira.codehaus.org/browse/CONTINUUM