BPM as a Service?

Prozessanwendungen mit Java EE 6 und Activiti

Daniel Meyer, Bernd Rücker

Sind Sie genervt von nicht funktionierenden Zero-Code-Tools oder komplizierten Stacks, die Ihnen Hersteller von Systemen für das Business Process Management (BPM) anbieten? Sehen Sie Cloud sowieso nur als Buzzword und die Verbindung mit BPM als Marketing-Gag? Das muss nicht sein, es gibt auch BPM-Ansätze, die sehr entwicklerfreundlich und leichtgewichtig sind und tatsächlich auch im Sinne von „Plattform as a Service" in der Cloud betrieben werden können. Dieser Artikel möchte einen konkreten Überblick über „Prozessanwendungen" mit Java EE 6 und der Open-Source-Engine Activiti geben und diese beispielhaft mit JBoss AS 7 und JBoss OpenShift in der Amazon-Cloud betreiben.

Doch zurück auf Anfang: Business Process Management (BPM) ist ein weites Feld, das schwierig zu überblicken ist und sowohl Aspekte der Organisation als auch der Prozessautomatisierung subsumiert. Die Tragweite reicht dann von der Fragestellung „Welche Prozesse sind im Unternehmen vorhanden?" bis zu „Wie arbeiten die einzelnen Mitarbeiter zusammen?", „Wie werden dabei IT-Systeme benutzt?", „Was müssen diese leisten?" und schließlich „Wie ist die benötigte Funktionalität implementiert?". Diese Fragen können in der Regel nicht von einer einzelnen Person beantwortet werden, sondern es braucht eher verschiedene Rollen und Experten mit unterschiedlichsten Bedürfnissen und Hintergründen, die zusammenarbeiten müssen, was spätestens für die Toolkette sehr anspruchsvoll ist.

Die Erkenntnis, dass in BPM-Projekten dieser weite Bogen gespannt werden muss, ist in die Entwicklung des neuen Standards zur Prozessmodellierung [BPMN2.0] eingeflossen. BPMN 2.0 erlaubt die Modellierung von Prozessen aus fachlicher Sicht, aber auch von ausführbaren Workflows und Serviceorchestrierungen.

Abbildung 1 zeigt einen beispielhaften Prozess im Zentrum einer Prozesslösung, worauf wir später im Artikel noch genauer eingehen. Rund um den Standard BPMN 2.0 werden zurzeit eifrig Werkzeuge entwickelt, von Modellierungsumgebungen und Prozessportalen für die fachliche Modellierung im Unternehmenskontext bis zu Eclipse-Plug-ins und Java-Prozessmaschinen für die Prozessautomatisierung.

Ein Beispiel für eine derartige Prozessmaschine (engl. Process Engine) ist das quelloffene Activiti [Activiti]. Dabei handelt es sich um eine leichtgewichtige, in Java geschriebene Prozessmaschine, die BPMN 2.0-Prozesse ausführen kann. Prozesse sind allerdings nur die „halbe Miete". Will man einen Prozess wirklich implementieren oder „automatisieren", muss der Ausführungskontext betrachtet werden, also sind Fragen zu beantworten wie „Welche Services sollen aufgerufen werden?", „Wie geht das?", „Wo sind die Transaktionsgrenzen?" oder „Wie werden Nachrichten empfangen?". Kurz: Es muss immer noch eifrig programmiert werden.

Ziel sollte aber sein, den Programmieraufwand wirklich in die Umsetzung von fachlichen Anforderungen und weniger in die Schaffung von Infrastruktur zu investieren und über den Prozess eine „Landkarte" für die Anordnung anderer Anforderungen zu haben, was sich in der Praxis als sehr hilfreich erwiesen hat.

Java EE 6

So wie BPMN 2.0 der Standard für Prozessmodellierung ist, ist Java Enterprise der Standard für die Entwicklung von Geschäftsanwendungen. Java EE liegt mittlerweile in der Version 6 vor, und sowohl der Standard als auch die Ausführungsumgebungen (Applikationsserver) wurden einer radikalen Verschlankungskur unterzogen. Damit lässt sich heute wesentlich produktiver arbeiten als mit Vorgängerversionen, die gerne als zu kompliziert und schwergewichtig kritisiert wurden.

Java EE bietet standardisiert und integriert Antworten auf die häufigsten Probleme bei der Entwicklung. So kann man „klein" anfangen, mit einfachen Services (Beans) und vielleicht JSF-Masken, sobald man aber weitere Features wie Persistenz, Web- oder REST-Services brauche, sind diese Features durch wenige Annotationen verfügbar. Neu ist dabei auch, dass der Preis für diese Features dramatisch gesunken ist. Waren Applikationsserver traditionell schwergewichtige Monster, sind Server der neusten Generation äußerst flexibel und teilweise sogar leichtgewichtiger als konkurrierende Umgebungen. Konkret bedeutet dies zum Beispiel, dass ein Java EE 6-zertifizierter JBoss 7 schneller hochfährt als ein Tomcat 6.

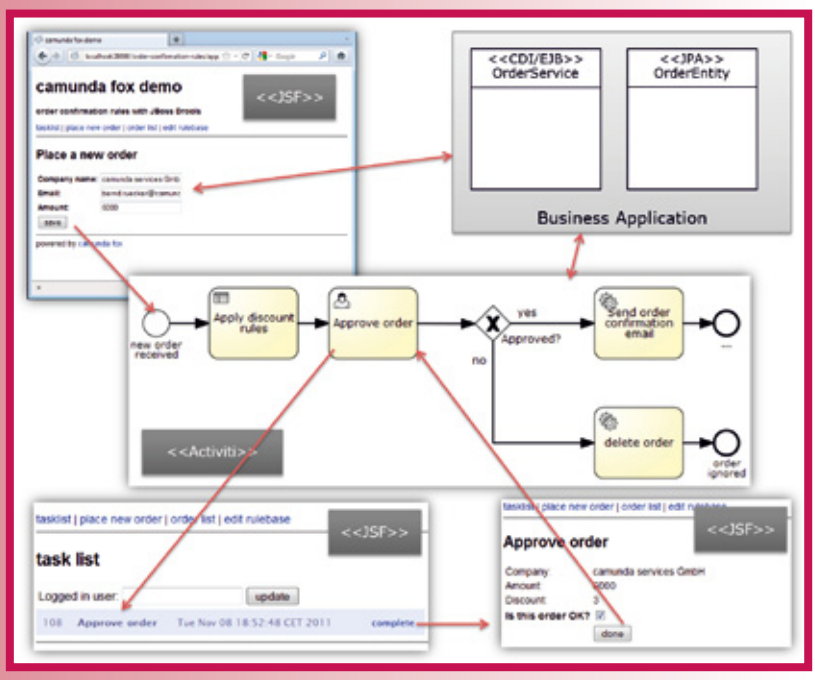

Abb. 1: Beispielhafter Prozess

Standards im Doppelpack: BPMN 2.0 + Java EE 6

Um nun also Prozessanwendungen maximal standardkonform zu entwickeln, liegt es nahe, Java EE und BPMN 2.0 zusammenzubringen, konkret wollen wir dies in diesem Artikel anhand von Activiti und JBoss 7 zeigen, die sich ideal integrieren lassen. Im Rahmen des Activiti-Projekts gibt

es mit dem Modul „activiti-cdi“ die wichtigsten Grundlagen: Der CDI-Standard, neu in Java EE, definiert ein sogenanntes SPI (Service Provider Interface), über welches sich in der Java EE-Plattform nicht vorhandene Technologien integrieren lassen.

Das Interessante ist hier, dass sich diese Technologien so integrieren lassen, dass sie ins Java EE 6-Programmiermodell passen. So lassen sich dann auch Prozessaspekte deklarativ mit Annotationen steuern. Methoden können zum Beispiel mit `@StartProcess` oder `@CompleteTask` annotiert werden, CDI-Beans oder EJBs lassen sich direkt aus dem Prozess aufrufen, und es wurde mit `@BusinessProcessScoped` ein neuer Scope für Prozessvariablen implementiert. Dies ermöglicht es nun, Anwendungen sehr produktiv zu entwicklen und sich dabei auf die fachlichen Anforderungen zu konzentrieren, ohne zuerst Lösungen für Infrastrukturprobleme finden und ständig den gleichen „Glue Code“ schreiben zu müssen.

Mit camunda fox gibt es ein Projekt, das noch einen Schritt weiter geht: Es integriert Activiti in den JBoss 7, also einen kompletten Java EE 6-Applikationsserver. Damit ist die Prozessmaschine als Dienst des Applikationsservers für alle Anwendungen verfügbar und kann wie Persistenz einfach verwendet werden. Auch ist der Activiti Explorer, eine vorgefertigte Aufgabenliste, direkt verfügbar und kann so zentral für Aufgaben aus verschiedenen Anwendungen verwendet werden.

Abbildung 1 zeigt alle Aspekte einer möglichen Prozessanwendung, vor allem Entitäten, Services, JSF-Oberflächen und BPMN 2.0-Prozesse. Dies kann dann in ein Deployment-Artefakt (WAR oder EAR) gepackt werden und wird sofort vom JBoss 7 verstanden. Auch für die Prozesse braucht es nur einen Deployment-Deskriptor, im einfachsten Fall nur die leere Datei META-INF/processes.xml. Das Archiv wird dann nach Prozessdefinitionen gescannt, diese werden bei Bedarf in die Datenbank der Prozessmaschine deployed und Vorkehrungen für das korrekte Classloading getroffen. Im Anwendungscode kann einfach über ein `@Inject Process Engine` die Programmierschnittstelle von Activiti aufgerufen werden. Da dies alles transparent passiert, wollen wir hier nicht weiter in die Tiefe einsteigen. Bei Bedarf ist auch ein kompletter Showcase online verfügbar [FoxDemo].

Rapid Development und Deployment

Es lassen sich also mit Java EE und Activiti relativ rasch Prozesslösungen in Java erstellen. Sind ein erster Prototyp erstellt und die ersten Anforderungen umgesetzt, so möchte man die Anwendung natürlich auch auf einem Integrationssystem deployen, wo sich dann ausgewählte Anwender oder auch der Business Analyst einmal „durchklicken“ können. Vor allem bei Geschäftsprozessen ist dies oft ein probates Mittel, um vorher modellierte Sachverhalte nochmals mit der Fachabteilung durchzugehen, aber eben nicht „trocken“ am BPMN-Modell, sondern anfassbar an einem Prototyp.

Aber davor sind noch einige Probleme zu lösen: Auf welcher Maschine setzt man das auf? Wie lange braucht ein Ticket zur Einrichtung eines neuen Servers? Wie installieren wir dort einen JBoss 7 und wie erklärt man seinem Administrator, was ein JBoss 7 ist?

Hier kann uns Plattform as a Service (PaaS) helfen, da sich dort bereits eine vollständig eingerichtete Maschine auf Anforderung und Bedarf hochfahren lässt. Im Firmenalltag könnte dies durchaus als Private Cloud abgebildet und damit abgesichert sein, in diesem Artikel wollen wir es beispielhaft anhand einer Public Cloud zeigen. Hierbei kommt JBoss OpenShift zum Einsatz, ein relativ junges Projekt, um den JBoss AS 7 in die Cloud zu hieven. Unter der Haube tickt eine Amazon EC2, also quasi die Mutter aller Public Clouds. OpenShift gibt es aktuell in zwei Ausprägungen: Express und Flex, wobei Express eine kostenfreie Einstiegsvariante und Flex die flexible Variante für den realen Einsatz sein soll.

Soweit die Rahmenparameter, aber was machen wir damit? Hätten wir einen Wunsch frei, so wollten wir doch eigentlich im Projekt auf Knopfdruck ein Testsystem erstellen und unsere aktuelle Prozessanwendung deployen. Die gute Nachricht: Das geht :-)

Cloud à la JBoss: OpenShift

Zuerst einmal muss man sich dazu ein Konto bei JBoss OpenShift [OpenShift] erstellen, was über die Weboberfläche aber schnell vonstattengeht. Danach kann man zur Einrichtung der konkreten Instanzen entweder die REST-Schnittstelle, ein Kommandozeilentool, ein Webinterface oder bald ein Eclipse-Plugin benutzen. Alles in allem ist dies wirklich schnell erledigt. Dabei kann man übrigens nicht nur JBoss 7-Instanzen erstellen, sondern auch andere Technologien wie PHP verwenden.

Ist die Instanz, die verwirrenderweise „Application“ heißt, eingerichtet, so bekommt man eine GIT-URL. Damit kann man nun über GIT Anwendungen deployen, die Serverkonfiguration anpassen oder auch dem JBoss 7 sogenannte Module hinzufügen. Letzteres können wir nutzen, um den nackten JBoss 7 in den „Activiti ready“ fox-Server zu verwandeln [Fox]. Ein GIT-Push führt dazu, dass die Cloud-Instanz diese Änderungen aufgreift und aktuell den Server automatisch neu startet, sofern man das SSH-Zertifikatsmanagement mit GIT gemeistert hat. Danach steht die Anwendung sofort im Internet mit eigener URL (anwendungsname.rhcloudc.com) zur Verfügung. Wem dies etwas zu knapp war, den möchten wir auf unseren Blog [BPMGuide] verweisen, in dem wir noch mehr technische Details zu OpenShift veröffentlichen.

Einrichtung per Knopfdruck mit cycle

Moment, sprachen wir nicht von „per Knopfdruck“ und haben dann eine Schritt-für-Schritt-Anleitung gegeben? Richtig, da fehlt noch etwas ;-) Für BPM-Projekte gibt es in camunda fox noch eine Komponente zur Kollaboration und Projektabwicklung: camunda fox cycle. In cycle hat man alle Artefakte zu einer Prozessanwendung im Überblick, so zum Beispiel das BPMN 2.0-Prozessmodell im fachlichen Modellierungstool, Issues im JIRA, aber auch das Entwicklungsprojekt im SVN. Mehr Informationen zu diesem ebenfalls quelloffenen Tool gibt es online [FoxCycle]. Cycle selbst ist dabei recht generisch und kennt eine Plug-in-Struktur, die es ermöglicht, eigene Funktionalität einzubinden.

Damit haben wir einen Ort für unseren Knopfdruck. Wir haben ein Plug-in geschrieben, welches alle oben genannten Schritte wirklich per Klick ausführt. Der Quellcode ist online [FoxCycleQuellen] verfügbar. Abbildung 2 zeigt das Ergebnis: Für unser Entwicklungsprojekt können wir per Kontextmenü einfach eine neue Cloud-Instanz erzeugen und die Anwendung deployen. Im gleichen Tool können wir übrigens auch einen Anwendungsrumpf für eine Prozessanwendung mit Java EE 6 von einem Template oder Maven Archetype erzeugen, oder einen Hudson/Jenkins-Job. Sprich: Den Weg von einem BPMN 2.0-Prozessmodell über ein neues Entwicklungsprojekt hin zu einem Deployment in der Cloud kann man wirklich innerhalb von Minuten zurücklegen. Auch wenn dies im ersten Moment nach Spielerei klingen mag, so haben wir doch sehr

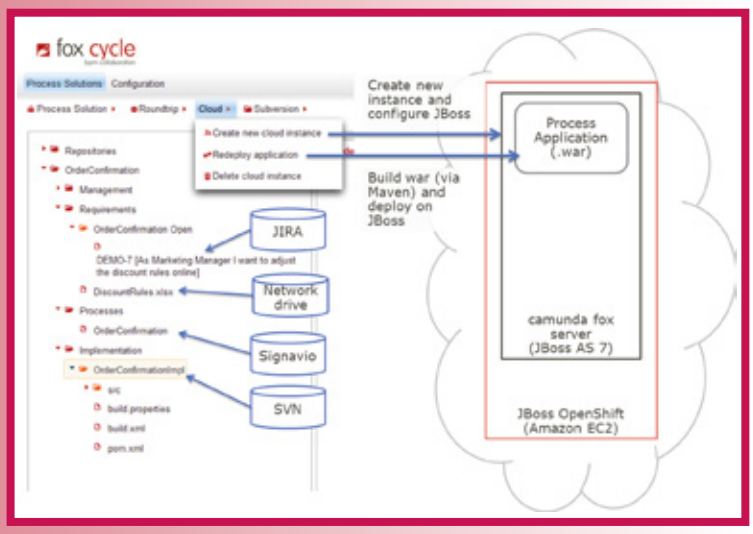

Abb. 2: Ergebnis des Knopfdrucks

positive Erfahrungen damit gemacht, ein Entwicklungsvorgehen derart Tool unterstützt in die Breite zu tragen.

Den gesamten Cloud-Machbarkeitsnachweis einschließlich Cycle-Plug-in haben wir übrigens an einem Samstag entwickelt, es ist also keine große Magie und somit wirklich eine interessante Bereicherung für das Entwicklungsprojekt. Aus unserer Sicht bleiben also kaum noch Ausreden, keine Prozessmaschine im Projekt einzusetzen, wenn es passende Anforderungen gibt.

Activiti und camunda fox

Im Artikel wurde sowohl Activiti als auch camunda fox genannt. Auf den ersten Blick ist die Beziehung zwischen den Projekten etwas verwirrend, was jedoch schnell aufgeklärt ist. Activiti ist ein von Alfresco initiiertes Open-Source-Projekt, an dem camunda von Anfang an mit entwickelt hat. Alfresco produktisiert Activiti als Teil ihres Enterprise-Content-Management-Systems und camunda produktisiert Activiti als BPM-Plattform unter dem Namen camunda fox. Da Alfresco lediglich Spring als Zielplattform im Sinn hat, ist auch bereits die Java EE-Integration der Prozessmaschine Teil von camunda fox. Durch den Scope der BPM-Plattform gibt es auch weitere Komponenten, wie zum Beispiel cycle, ein Werkzeug zur Kollaboration in BPM-Projekten. In camunda fox ist Activiti als Prozessmaschine enthalten.

Zero Code is dead

Dies bringt uns zu einem interessanten Punkt, der Zukunft von BPM-Projekten und -Tools. An Zero Code glauben wir nämlich nicht – oder nur für begrenzte Anwendungsfälle. Viel spannender ist es, Prozessmaschinen in die vorhandenen Stacks einzuklinken und es Entwicklern so einfach wie möglich zu machen. Denn die Entwickler werden für viele Aspekte in Prozessanwendungen sowieso gebraucht. Und sie können Formulare für die Aufgabenabarbeitung einfacher und flexibler in JSF oder gar JSP erstellen als im proprietären Framework eines Herstellers. Wir sprechen gerne von „less code“ anstatt „zero code“, oder wie Jacob Ukelson es gut auf den Punkt gebracht hat: „Are BPM suites another 4GL?“ [Ukel11].

Mit dem Stack um camunda fox, also Java EE 6, Activiti und Tools wie cycle, gehen wir genau diesen Weg, der dank vieler Open-Source-Komponenten zum Ausprobieren anregt. Das in der Einstiegsvariante sogar kostenfreie Cloud-Angebot mit JBoss OpenShift ist dabei eine willkommene Ergänzung, die die Einrichtung einer Test- oder sogar ersten Produktivumgebung massiv beschleunigt. Spätestens in einer privaten Cloud in Unternehmen könnte dies ein spannendes Zukunftsmodell sein, um die Zeiten zwischen dem Entwicklungsende und der Marktfreigabe zu verkürzen. Aber auch für neue Projekte oder gar Start-ups ermöglicht dies ein schnelles Online-Setzen von Anwendungen.

Wer enttäuscht ist, in Zukunft weniger tief in der Technik rumfummeln zu müssen, sollte sich eher bei einem Applikationsserver-Hersteller bewerben, für den Rest von uns die gute Nachricht: Es geht Schritt für Schritt in die richtige Richtung, lasst uns lieber Fachlichkeit entwickeln als Infrastruktur :-)

Links

[Activiti] Activiti Business Process Management (BPM) Platform, `www.activiti.org`
[BPMGuide] Blog von camunda services GmbH, `www.bpm-guide.de`
[BPMN2.0] Business Process Model and Notation (BPMN), Version 2.0, `http://www.omg.org/spec/BPMN/2.0/`
[Fox] camunda fox, The Enterprise BPM Platform based on Activiti, s. a. `www.camunda.com/fox/community/`
[FoxCyle] Cycle – The BPM collaboration component, camunda, `www.camunda.com/fox/components/cycle/`
[FoxCycleQuellen] Quellcode zu Cycle, `https://svn.camunda.com/fox/trunk/cycle-plugins/fox-cycle-plugin-openshift/`
[FoxDemo] fox – Revision 2038: /demo/fox/order-confirmation-rules, `https://svn.camunda.com/fox/demo/fox/order-confirmation-rules/`
[JavaEE6] Java EE 6 Technologies, `http://www.oracle.com/technetwork/java/javaee/tech/index.html`
[OpenShift] JBoss AS7 + OpenShift, `www.jboss.org/openshift`
[Ukel11] Are BPM suites another 4GL (fourth generation programming language), Jacob Ukelson's Blog, 22.10.2011, `http://ukelson.wordpress.com/2011/10/22/are-bpm-suites-another-4gl-fourth-generation-programming-language/`

Daniel Meyer ist Berater bei der camunda services GmbH. Er hat seinen Schwerpunkt im Bereich der Java-Enterprise-Technologie sowie der Prozessautomatisierung in BPM-Projekten. Daniel Meyer entwickelt aktiv in Activiti mit und ist Lead der CDI-Integration.
E-Mail: daniel.meyer@camunda.com

Bernd Rücker ist Geschäftsführer der camunda services GmbH und Autor. Er verfügt über mehrjährige Erfahrung als Softwarearchitekt, Coach, Berater, Trainer und Entwickler im Umfeld von Java EE, BPM und SOA.
E-Mail: bernd.ruecker@camunda.com