Apex und Workflows

Sven Böttcher, Apps Associates

Als Apex-Berater ist man in Kundenprojekten oft mit der Frage konfrontiert, welche Möglichkeiten Apex für die Entwicklung komplexer Workflows bietet. Die Antwort lautet: Leider keine, es muss alles in PL/SQL implementiert werden! Dies führt häufig zu einer sehr komplexen und nur schwer wartbaren Applikationslogik. Grund genug, sich mit Alternativen der Workflow-Implementierung in Apex auseinanderzusetzen.

Apex erfreut sich als Rapid-Application-Development-Tool für die Entwicklung datenbankzentrischer Web-Anwendungen großer Beliebtheit. Gründe sind unter anderem, dass Apex lizenzkostenfrei verwendet werden kann und die Entwicklung von Eingabemasken durch einen GUI-Editor schnell und einfach erfolgt. Die GUI-Editor-gestützte Anwendungsentwicklung macht vor allem ein prototypisches Vorgehen möglich, wodurch sich die späteren Benutzer sehr gut in den Software-Entwicklungsprozess einbeziehen lassen. Auf diese Weise kann auf Missverständnisse oder gegebenenfalls auftretende Änderungswünsche bereits in einem frühen Stadium des Entwicklungsprozesses reagiert werden.

Es kommt jedoch häufig vor, dass für eine komplexe Applikationslogik entsprechend aufwändige Workflows umgesetzt werden müssen. An dieser Stelle bietet Apex keinerlei Unterstützung. Die einzige Möglichkeit, Workflows umzusetzen, besteht in der manuellen Programmierung, im Allgemeinen in der Programmiersprache PL/SQL. Dies führt nicht selten zu einem

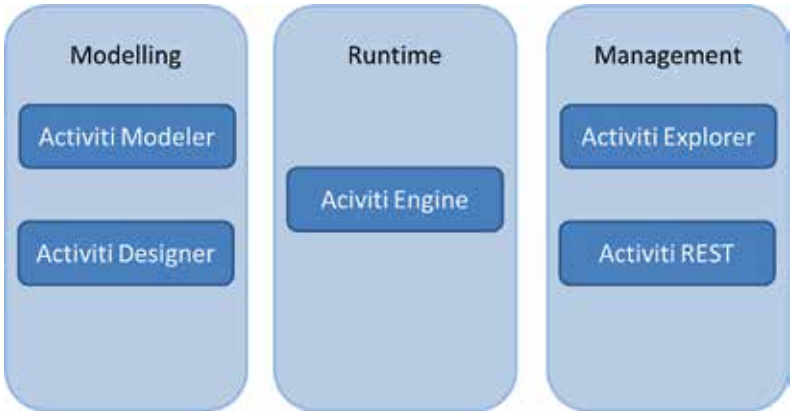

Abbildung 1: Die Activiti-Komponenten (Quelle: Activiti.org)

sehr langen und komplizierten Programmcode, der fehleranfällig und nur schwer wartbar ist. Zudem kann der Programmcode in der Regel nicht mehr von den späteren Benutzern der Anwendung verstanden werden. Dies und der zumeist hohe Aufwand der Workflowprogrammierung erschweren wiederum die prototypische Anwendungsentwicklung und das Einbeziehen der Benutzer in den Entwicklungsprozess. Eine Lösung dieses Dilemmas besteht in der Integration von Apex und einer Workflow-Engine, die idealerweise in einer grafischen Beschreibungssprache (wie Business Process Model and Notation – BPMN) beschriebene Workflows ausführen kann.

Integration von Apex und Workflow Engines

Die Vorteile der Integration von Workflow Engines beziehungsweise BPM-Suiten und Apex liegen auf der Hand. Prozess-Analysten können die zugrunde liegenden Workflows in ihnen bekannten Modellierungssprachen modellieren und die so entstehenden Workflows lassen sich ohne eine Re-Implementierung (in PL/SQL) direkt ausführen. Da die verwendeten Modellierungssprachen zumeist grafischer Natur sind, werden die beschriebenen Workflows auch von den Benutzern der Anwendung verstanden. Zudem wird die Applikation wartungsärmer und die Applikationslogik verständlicher, da sich die Menge des Quellcodes reduziert.

Die grundlegende Idee der Integration besteht darin, dass die Workflow Engine den Programmablauf beziehungsweise die Programmlogik steuert. Apex hingegen sollte Eingabemasken für die Nutzer-Interaktion bereitstellen. Die einzelnen Tasks eines Workflows müssen daher mit entsprechenden Apex-Seiten für die Bearbeitung der Tasks verknüpft sein. Diese Verknüpfung kann dabei über das Anreichern der Tasks eines Workflows mit Meta-Informationen erfolgen. Für die Abfolge von Eingabemasken beziehungsweise Apex-Seiten würde das zum Beispiel bedeuten, dass zu jedem Task des Workflows festgelegt wird, mit welcher Apex-Seite dieser Task zu bearbeiten ist.

Die eigentliche Integration beziehungsweise Kommunikation zwischen Apex und der Workflow Engine kann über den Aufruf von Webservices erfolgen. Diese werden durch Apex initialisiert und dienen beispielsweise dem Starten von Prozessen, der Abfrage von als Nächstes auszuführenden Tasks, der Zuweisung von Tasks zu Benutzern und dem Beenden von Tasks.

Da jeder Task über Meta-Informationen mit einer Apex-Seite verknüpft ist, kann einem Benutzer nach der Übernahme eines Tasks die entsprechende Apex-Seite für die Bearbeitung dieses Tasks angezeigt werden. Den Entwicklern einer Apex-Anwendung kommt nun die Aufgabe zu, die Tasks des Workflows mit entsprechenden Meta-Informationen anzureichern sowie die Implementierung der eigentlichen Integration von Apex und der Workflow Engine durchzuführen. Auch wenn Letzteres aufwändig erscheint, entsteht der Vorteil, dass die Webservice-Aufrufe im Gegensatz zu in PL/SQL implementierten Workflows nicht projektspezifisch sind und sich als eine Art Framework in den verschiedensten Projekten wiederverwenden lassen.

Activiti

Activiti ist eine quelloffene Workflow- und Business-Process-Management-Plattform (BPM), die im Rahmen der Apache-Lizenz kostenlos verwendet werden kann. *Abbil-*

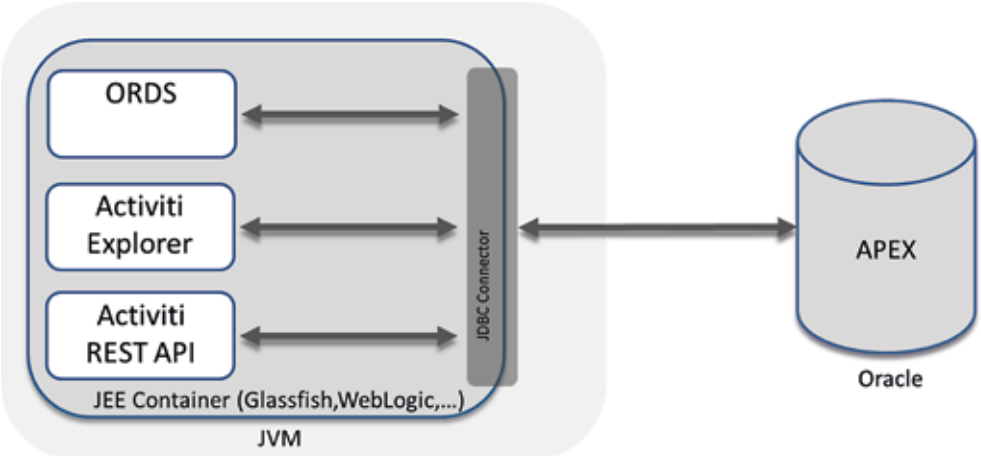

Abbildung 2: Apex- und Activiti-Infrastruktur

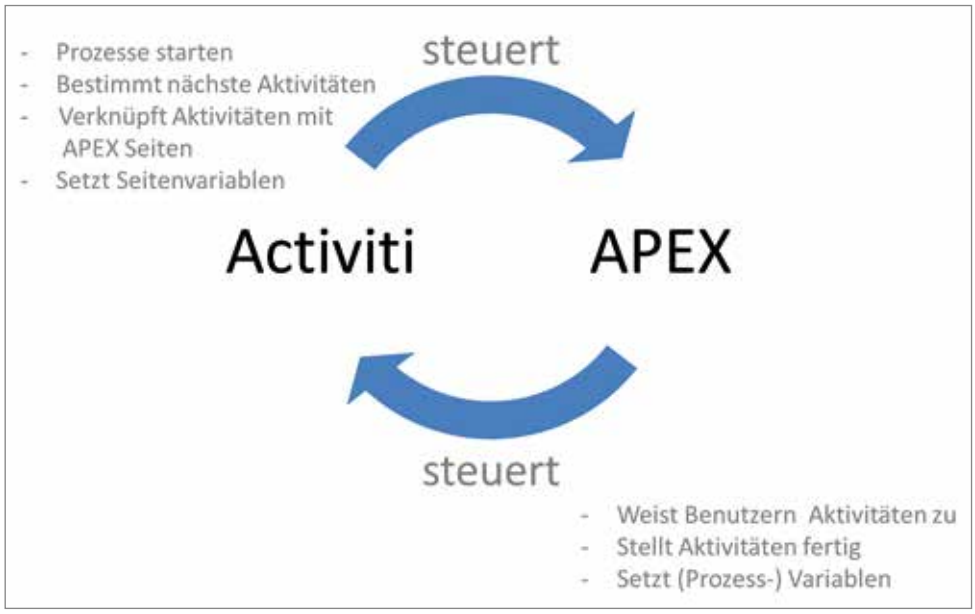

Abbildung 3: Wechselseitige Steuerung zwischen Apex und Activiti

dung 1 zeigt die zentralen Komponenten von Activiti. Sie lassen sich grob in die Bereiche „Modellierung", „Laufzeit" und „Management" gliedern.

Während der Activiti Modeler eine sehr einfache und webbasierte Möglichkeit zur Modellierung von BPMN-2.0-Prozessen bietet, besteht der eigentliche Kern der Activiti-BPM-Suite aus der Activiti Engine. Diese führt BPMN-2.0-Prozesse aus und kann entweder standalone quasi als Java-Programm oder auf einem JEE-konformen Application Server (wie GlassFish, WebLogic etc.) betrieben werden. Die komplette Steuerung der Activiti Engine kann entweder Java-basiert oder über ein REST-API (Activiti REST) erfolgen. Für Letzteres ist ein JEE-konformer Application Server erforderlich. Dieses Feature ist Grundlage der hier beschriebenen Apex-Integration.

Apex und Activiti

Activiti verfügt über ein sehr umfangreiches REST-API, mit dem Activiti komplett gesteuert werden kann. Darüber hinaus lassen sich Tasks über den Activiti Modeler (mit dem man auch die Workflows modelliert) mit den benötigten Meta-Informationen anreichern. Die beschriebenen Anforderungen für die Apex-Integration sind demzufolge von Activiti erfüllt.

Apex und Activiti passen aber auch hinsichtlich der benötigten Infrastruktur bestens zusammen. Da Apex in der Oracle-Datenbank läuft und sich diese problemlos als Activiti-Repository verwenden lässt, ist keine zusätzliche Datenbank notwendig. Wird auf Apex (wie von Oracle empfohlen) mithilfe der Oracle REST Data Services (ORDS) zugegriffen, kann auch für das Activiti-REST-API und die Activiti Modeler die Apex-Infrastruktur mitverwendet werden. Je nach Deployment ist keine zusätzliche Infrastruktur nötig, was insbesondere die Betriebskosten hinsichtlich der Administration und Wartung gering hält. *Abbildung 2* zeigt schemenhaft die gemeinsame Nutzung der Infrastruktur von Apex und Activiti.

Für die Integration beider Tools existiert eine PL/SQL-Bibliothek, die im Wesentlichen eine Menge von Webservice-Aufrufen darstellt. Diese steuern die Activiti Engine wie beschrieben und liefern als Ergebnis verschiedene Informationen zurück, etwa die Apex-Seite, die zur Bearbeitung eines Tasks angezeigt werden soll. Dafür müssen aber zunächst die Tasks eines Workflows mit entsprechenden Meta-Informationen angereichert sein. Dazu dient ein Activiti-spezifisches Dokumentations-Tag; die Meta-Informationen werden innerhalb dieses Tags durch Kommas getrennt in die Zeichen „/{...}/" gefasst und so als Meta-Informationen kenntlich gemacht.

Zum jetzigen Entwicklungsstand können mithilfe dieser Meta-Informationen die zur Bearbeitung des Tasks aufzurufende Apex-Seite, allgemeine Informationen zu dem Task und zu setzende Apex-Page-Items angegeben werden. Durch „apex_page:page_number" wird zum Beispiel die Seite „page_number" aufgerufen und durch „set_page_item(page_item:activiti_process_variable)" das Page-Item „page_item" auf den Wert der Activiti-Prozess-Variable „activiti_process_variable" gesetzt. Über „get_info(activiti_process_variable)}"

```
--Request erstellen
 l_http_request := utl_http.begin_request(
   'http://192.168.56.101:8080/activiti-
   rest/service/runtime/tasks?candidateGroup=it_helpdesk
&unassigned=true',
   'GET',
   'HTTP/1.1');


--Activiti Benutzerdaten setzen
 utl_http.set_authentication(
   l_http_request,
   'admin',
'admin');

--Request absetzen
l_http_response := utl_http.get_response(l_http_request);
utl_http.read_text(l_http_response, l_response_text);
utl_http.end_response(l_http_response);

--Response parsen
l_jsonObj := json(l_response_text);
l_jsonList := json_ext.get_json_list(l_jsonObj, 'data');
```

Listing 1

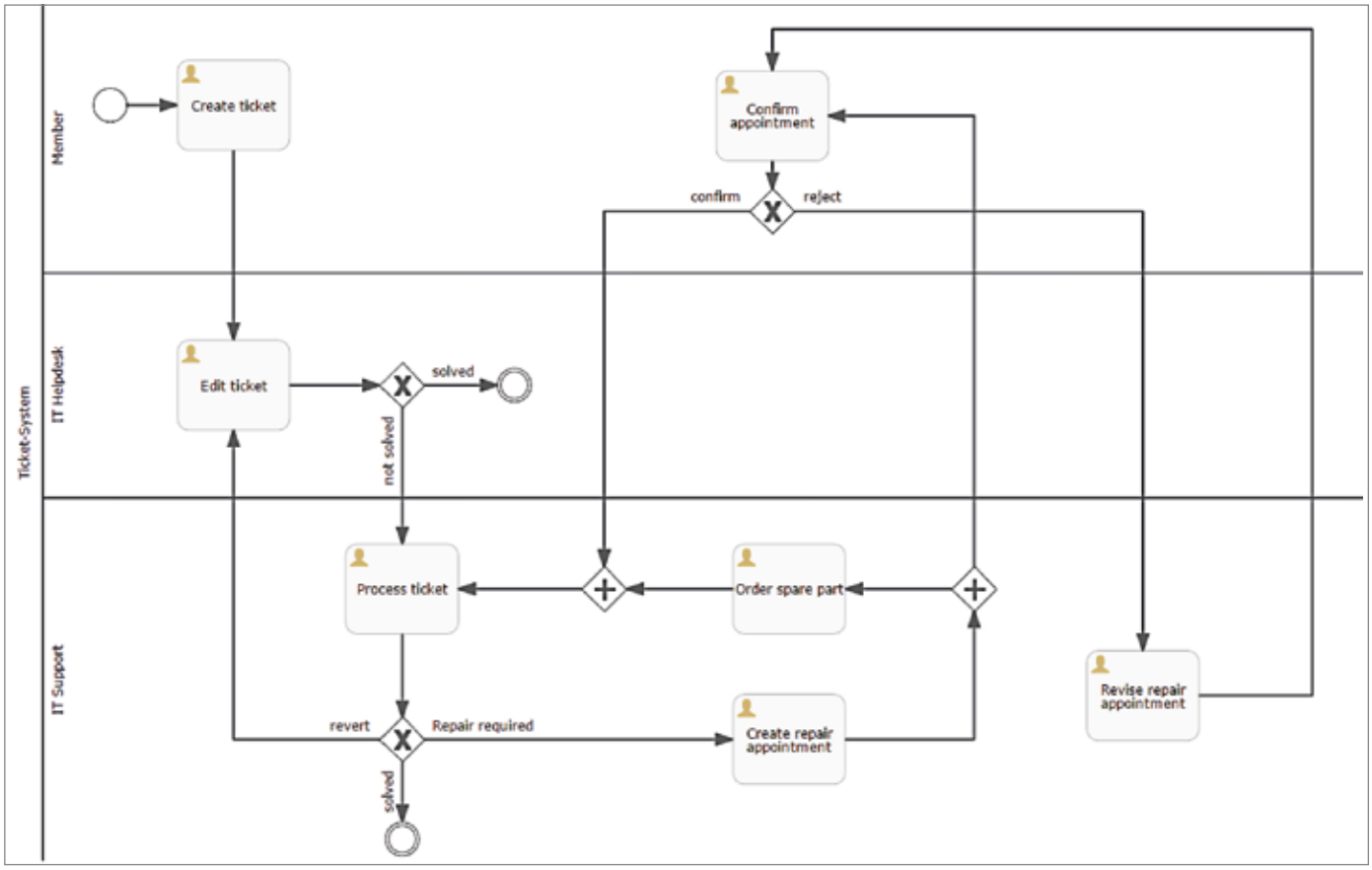

Abbildung 4: Workflow der Beispiel-Anwendung

lassen sich allgemeine Informationen zu dem Task abfragen, die zuvor durch eine Activiti-Prozess-Variable über einen Webservice-Aufruf gesetzt wurden.

Eine vollständige Meta-Information besitzt das Format „/{apex_page:page_number,set_page_item(item:value) ,get_info(info)}/". Auf diese Weise erfolgt die wechselseitige Steuerung zwischen Apex und Activiti, wie in *Abbildung 3* dargestellt. Durch Apex beziehungsweise entsprechende Webservice-Aufrufe werden Activiti-Prozesse gestartet, Benutzern Aktivitäten zugewiesen, Aktivitäten abgeschlossen sowie benötigte Activiti-Prozess-Variablen gesetzt. Activiti steuert dagegen über die beschriebenen Meta-Informationen den Ablauf der anzuzeigenden Apex-Seiten, setzt Apex-Page-Items und bestimmt, welche Aktivitäten als Nächstes ausgeführt werden können.

Über einen Webservice-Aufruf lassen sich beispielsweise alle Tasks abfragen, die noch keinem Benutzer zugeordnet sind und als Nächstes ausgeführt werden können *(siehe Listing 1)*. Die Tasks werden in Form einer JSON-Liste zurückgeliefert und können einfach geparst werden, zum Beispiel mithilfe der PL/JSON-Bibliothek. Alternativ kann dies natürlich auch über das seit Apex 5.0 verfügbare Paket „Apex_JSON" erfolgen.

Beispiel-Anwendung

Als Beispiel wurde ein einfaches Ticketsystem implementiert. Der Workflow wurde mit dem Activiti Modeler modelliert und ist in *Abbildung 4* dargestellt. Alle Tasks, die ein Benutzer ausführen kann, werden durch den oben beschriebenen Webservice-Aufruf abgefragt und dem Benutzer auf einer Übersichtseite zur Auswahl dargestellt. Die ausführbaren Tasks hängen dabei von der Rolle des Benutzers ab. Ein Benutzer mit der Rolle „IT-Helpdesk" könnte beispielsweise die in *Abbildung 5* dargestellten Tasks sehen.

Abbildung 5: Auswahlseite der als Nächstes ausführbaren Tasks

Wird die Bearbeitung des Tasks abgeschlossen, geht der entsprechende Prozess in den nächsten Zustand. Der abgeschlossene Task verschwindet aus der Liste der als Nächstes ausführbaren Tasks des Benutzers mit der Rolle „IT-Helpdesk“ und die Benutzer mit der Rolle „IT-Support“ sehen einen neuen Task „Process Ticket“ in ihrer Liste der nachfolgend ausführbaren Tasks.

Die Apex-Seite, auf die ein Benutzer nach der Auswahl eines Tasks geleitet wird, hängt ausschließlich von den mit dem Task verknüpften Meta-Informationen ab. Der Task „Edit ticket“ wurde beispielsweise mit der Meta-Information „/{apex_page:301,set_page_item(P301_TICKET_ID:ticket_id)}/“ versehen. Dadurch wird der Benutzer bei der Auswahl eines Tickets auf die Apex-Seite mit der Nummer „301“ geleitet und das Page-Item „P301_TICKET_ID“ auf den Wert der Activiti-Variablen „ticket_id“ gesetzt. Durch das Setzen der Variable „ticket_id“ beim Starten eines Activiti-Prozesses, der hier die Bearbeitung eines Tickets beschreibt, wird der Prozess mit einem in der Datenbank gespeicherten Ticket verknüpft.

Durch die Activiti-Integration müssen keinerlei Workflow-Informationen wie zum Beispiel darüber, welcher Task durch welchen Benutzer ausgeführt wird, durch die Applikation gespeichert beziehungsweise verwaltet werden. Apex wird ausschließlich zum Bearbeiten von in der Datenbank gespeicherten Informationen (hier Tickets) verwendet, während der Workflow der Anwendung komplett durch Activiti gesteuert wird und ausschließlich in der in *Abbildung 4* dargestellten grafischen Beschreibung vorliegt.

Fazit und Ausblick

Durch die vorgestellte Methode lässt sich Apex grundsätzlich mit diversen Workflow-Engines beziehungsweise BPM-Plattformen integrieren. Die Voraussetzungen dafür sind, dass sich zum einen die zu integrierende Workflow-Engine über Webservices steuern lässt und sich zum anderen Tasks um Meta-Informationen anreichern lassen. Während beispielsweise eine Integration mit der Activiti-BPM-Plattform prototypisch durchgeführt wurde, besteht das Ziel in der Entwicklung eines Frameworks, über das sich verschiedene BPM-Plattformen integrieren lassen. Auch wenn die Integration zunächst einen recht hohen Entwicklungsaufwand fordert, ergibt sich der Vorteil, dass ein solches Framework in verschiedenen Projekten wiederverwendet werden kann.

Konkrete Workflows, die in PL/SQL programmiert werden, sind dagegen sehr spezifisch und werden von Prozess-Analysten und den Anwendern der Applikation nicht verstanden. Durch die toolgestützte, grafische Entwicklung und die anschließende Einbindung dieser Workflows in Apex lässt sich der Rapid-Prototyping-Ansatz zumindest auf Teile der Programmlogik ausweiten. Zudem lassen sich Fehler in der Programmlogik minimieren, da die zeitaufwändige Re-Implementierung der Workflows in PL/SQL entfällt.

Sven Böttcher
sven.boettcher@appsassociates.com