Streamline Development im Oracle-Projekten

Markus Lohn
esentri AG
Ettlingen

Schlüsselworte
Softwareentwicklung, Automatisierung, Oracle

Abstrakt
Aufbau und Bereitstellung von Entwicklungsumgebung muß schnell, automatisiert und für alle Entwickler gleich funktionieren. Insbesondere darf die Anlaufzeit für Entwickler im Projekt bis zur tatsächlichen Unterstützung im Projekt nicht mehr Tage oder Woche dauern. Anhang von Beispielen und Erfahrungen werden Lösungsmöglichkeiten im Oracle-Umfeld aufgezeigt. Gezeigt werden Technologien wie Virtual Box, Vagrant, Ansible, Maven, WLST...

Einleitung
Dynamische Märkte und das stetige Reagieren auf veränderte Bedingungen ist heute ein entscheidender Erfolgsfaktor für Unternehmen. Alle ergriffenen Maßnahmen in den vergangenen Jahren im Bereich der Softwareentwicklung hatten zum Ziel diese Dynamik IT-technisch zu unterstützen bzw. zu ermöglichen. Entwicklungszeiten mussten drastisch reduziert und Ergebnisse früher als bisher zum Review zur Verfügung gestellt werden. IT hat heute einen anderen Stellenwert im Unternehmen erreicht. Durch IT können neue Wachstumsfelder erschlossen und bestehende Kundenbindungen gestärkt werden. Aus diesem Grund ist es wichtig sich mit der Fragestellung zu befassen, wie schnell ein Entwickler in einem neuen Projekt produktiv mitarbeiten kann. In diesem Vortrag werden Werkzeuge und Vorgehensweisen vorgestellt, um einen Entwickler in die Lage zu versetzen, in weniger als einem Tag produktiv im Projekt mitzuwirken. Im Detail bedeutet das beispielsweise eine User Story aus dem Backlog zu bearbeiten.

Die Erfahrung zeigt, dass in vielen Projekten das Aufsetzen der Entwicklungsumgebung mit vielen manuellen Schritten verbunden ist. Fehlende oder unzureichende Dokumentation verlangsamt diesen Prozess erheblich. Teilweise benötigt das Aufsetzen mehrere Tage. Durch die hier vorgestellte Herangehensweise ist der Entwickler in der Lage, innerhalb weniger Stunden mit einer Entwicklungsumgebung versorgt zu werden. Aufgrund des hohen Automatisierungsgrades entfällt auch die Notwendigkeit einer umfassenden Dokumentation. Fehler werden auf diese Weise weitestgehend vermieden.

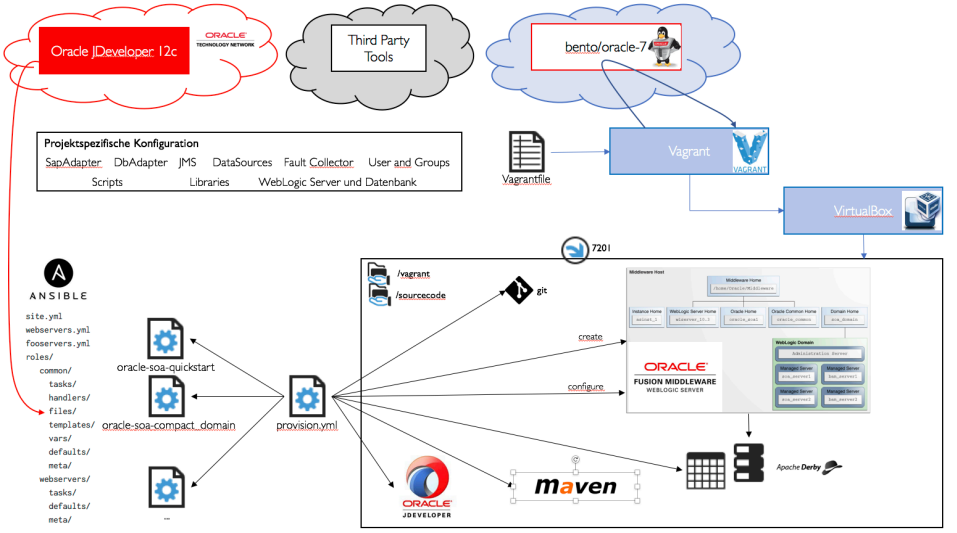

Abbildung 1 Aufbau und Konfiguration einer Entwicklungsumgebung für Oracle-Projekte

In der Abbildung sind typische Komponenten aufgeführt, die zu einer Entwicklungsumgebung gehören. In diesem Vortrag fokussiere ich mich auf den Einsatz von Werkzeugen in Projekten mit Oracle-Technologien (SOA/Integration, ADF, Java EE).

git

Es ist sinnvoll die Beschreibung und Konfiguration einer Entwicklungsumgebung als Sourcecode zu betrachten und somit in einer Sourcecodeverwaltung abzulegen. Die Versionierung der Artefakte und zentrale Bereitstellung über die Sourcecodeverwaltung sind die entscheidenden Vorteile dieses Ansatzes. Obwohl jedes beliebige System für diesen Zweck genutzt werden kann, bietet git jedoch durch einen dezentralen Ansatz viele Vorteile in der Benutzung.

Um die Entwicklungsumgebung aufzubauen muss zunächst das Repository aus git mit dem folgenden Befehl kopiert werden:

git clone https://git-servername/oracle/vagrant-oracle-soa12231-dev-desktop.git

Im Root-Ordner befindet sich die Dokumentation und Provisionierungs-Skripte. Im Unterordner /files werden alle Skripte und Konfigurationsdateien für den Aufbau der Entwicklungsumgebung gespeichert. Die Verwaltung von Installationsdateien, z. B. JDeveloper, muss jedoch auf andere Art und Weise durchgeführt werden. Es ist nicht zu empfehlen diese ebenfalls in git zu speichern, wenn die Größe mehrere 100 MB übersteigt. Eine Möglichkeit besteht darin, die Installationsdateien über Skripte (wget) aus dem Internet bei Bedarf zu laden. Die andere Variante besteht darin, die Dateien in den Ordner /files manuell zu kopieren. Der automatische Download der JDeveloper-Dateien über OTN oder eDelivery funktioniert aufgrund der Sicherheitseinstellungen (Token) jedoch nicht zuverlässig.

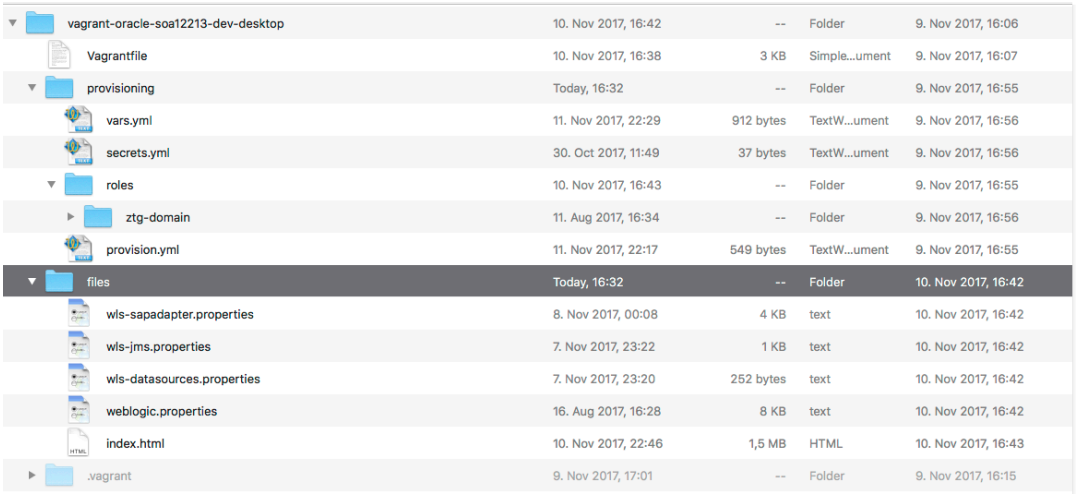

Abbildung 2 git-Repository mit Entwicklungsumgebung

Nach dem Checkout aus git müssen die Installationsdateien für den JDeveloper beschafft und im Ordner /files abgelegt werden.

Vagrant

Die Entwicklungsumgebung basiert auf einer virtuellen Maschine (VM). Die VM wird mit Oracle Virtual Box ausgeführt und durch Vagrant verwaltet. Aus diesem Grund muss Oracle Virtual Box und Vagrant vorinstalliert werden. Vagrant ist ein Werkzeug, um virtuelle Maschinen zu konfigurieren und

zu verwalten. In einer Konfigurationsdatei (Vagrantfile) werden die Eigenschaften der VM definiert, z. B. Name, Netzwerkeinstellungen, Memory, etc. Ferner enthält die Datei Informationen über die Art der Provisionierung. Es gibt mehrere Möglichkeiten eine Provisionierung durchzuführen, z. B. mit Ansible oder mit Shell-Skripten. Eine VM kann mit Vagrant jederzeit gelöscht und wieder neu erstellt werden. Im Internet existiert ein Repository für Boxes. Boxes sind vorgefertigte VM Konfigurationen, die für das Aufsetzen von neuen Images herangezogen werden können.

Mit dem folgenden Befehl wird die VM gestartet:

vagrant up

Beim ersten Ausführen wir die VM neu erstellt und konfiguriert. Anschließend wird die Provisionierung ausgeführt. Mit dem Befehl

vagrant halt

wird die virtuelle Maschine heruntergefahren.

Die Verwaltung von VM's wird durch Vagrant erheblich vereinfacht. Beispielsweise werden Shared Folders, Portforwarding etc. automatisch eingerichtet. Die manuelle Konfiguration dieser Eigenschaften für eine VM entfällt. Ferner wird die gesamte Konfiguration in einer Textdatei gespeichert und in der Sourcecodeverwaltung abgelegt. Jeder Entwickler hat somit die gleiche Ausgangsbasis. Entwicklerspezifische Konfigurationen können in eigenen Branches konfiguriert und getestet werden.

Entwicklungsumgebung: Oracle JDeveloper 12c
In diesem Beispiel wird als Entwicklungsumgebung Oracle JDeveloper 12c eingesetzt. Es wird empfohlen die JDeveloper-Variante mit SOA-Plugin zu verwenden. Mit dieser Option stehen alle wichtigen Oracle Module (SOA Suite, OSB, ADF, etc.) für die Entwicklung und Runtime zur Verfügung. Die Installationsdateien müssen zunächst beschafft werden und in den Ordner /files kopiert werden. Dieser Schritt kann nicht automatisiert werden, aufgrund der Sicherheitseinstellung im OTN bzw. eDelivery (Token wird generiert und ist nur für einen bestimmten Zeitraum gültig). Die benötigten Ordner und Rechte müssen vor dem Ausführen des Installationsprogramms im Image erstellt und konfiguriert werden. Um die Option „silent" bei der Installation vom JDeveloper benutzen zu können, ist ferner ein sog. Responsefile erforderlich. Mit dem JDeveloper und SOA-Plugin kann eine komplette Runtime-Umgebung mit Datenbank und WebLogic Server aufgebaut werden. Es werden unterschiedliche Konfigurationsarten angeboten (siehe Oracle Dokumentation: Introduction to Quick Start Distributions). In diesem Beispiel wird eine sog. compact_domain bestehend aus Admin-Server + Servicebus + SOA Suite installiert. Als Datenbank kommt eine JavaDB (Apache Derby) zum Einsatz. Diese Variante eignet sich sehr gut für die Entwicklung und erlaubt auch das Debugging direkt im JDeveloper.
Die compact_domain kann mit einem WLST-Skript automatisch erstellt werden. Dieser Vorgang dauert üblicherweise nur wenige Minuten. Je nach Einsatzzweck kann es notwendig sein, eine Oracle-Datenbank in der Entwicklung einzusetzen. In einem solchen Fall kann eine Oracle XE-Datenbank genutzt werden. Auch die Installation und Konfiguration einer XE-Datenbank kann mit den vorgestellten Werkzeugen automatisiert werden. Auch die Integration über ein Docker-Image wäre möglich.

Mit dem JDeveloper 12c und SOA-Plugin kann eine vollständige Entwicklungsumgebung für Oracle in wenigen Minuten erstellt werden. Somit bietet Oracle jetzt auch einen leichtgewichtigen Ansatz für die Entwicklung an.

Middleware Konfiguration

Neben der Basisinstallation mit Entwicklungs- und Runtimeumgebung gibt es in jedem Projekt noch spezifische Konfigurationen und Applikationen die für die Entwicklung benötigt werden:

- Datenbankschemata und Tabellen
- Data Sources
- Messaging (Queues)
- JCA Adapter
- Security
- Externe Libraries
- ...

Über die Provisionierung können alle diese Konfigurationen automatisiert durchgeführt und jederzeit wiederholt werden. Im Bereich der Datenbank können zusätzliche Werkzeuge, wie beispielsweise Flyway, genutzt werden. Konfigurationseinstellungen am WebLogic Server können über WLST-Skripte automatisiert werden.

```
domainRuntime()
edit()
startEdit()

cd('/')
cmo.createJDBCSystemResource(dsName)

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName)
cmo.setName(dsName)

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDataSourceParams/' + dsName)
set('JNDINames',jarray.array([String(dsJNDIName)], String))

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName)
cmo.setDatasourceType('GENERIC')

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName)
cmo.setUrl(dsURL)
cmo.setDriverName(dsDriver)
set('Password', dsPassword)

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCConnectionPoolParams/' + dsName)
if dsTestQuery != None:
   cmo.setTestTableName(dsTestQuery)
else:
   cmo.setTestTableName('SQL SELECT 1 FROM DUAL\r\n\r\n')

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName + '/Properties/' + dsName)
cmo.createProperty('user')

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDriverParams/' + dsName + '/Properties/' + dsName + '/Properties/user')
cmo.setValue(dsUsername)

cd('/JDBCSystemResources/' + dsName + '/JDBCResource/' + dsName + '/JDBCDataSourceParams/' + dsName)
if 'XA' in dsDriver:
   cmo.setGlobalTransactionsProtocol('TwoPhaseCommit')
else:
   cmo.setGlobalTransactionsProtocol('EmulateTwoPhaseCommit')

cd('/SystemResources/' + dsName)
set('Targets',jarray.array([ObjectName('com.bea:Name=' + dsTargetName + ',Type=' + dsTargetType)], ObjectName))

print 'Creating Data Source ', dsName, ' done.'

save()
activate()
```

Abbildung 3 Codebeispiel zum Anlegen von Data Sources als WLST-Skript

Somit ist es sehr einfach möglich die lokale Entwicklungsumgebung jederzeit in einen initialen Zustand zu versetzen. Insbesondere für die Testautomatisierung ein wichtiges Kriterium.

Für die Provisionierung können unterschiedliche Technologien und Frameworks zum Einsatz kommen. Unterstützt werden auf jeden Fall Shell-Skripte, Ansible, Puppet und Docker. Andere Alternativen können auf der Vagrant Homepage nachgeschlagen werden. In diesem Vortrag wurden nur die Möglichkeiten mit Shell-Skripten und Ansible aufgezeigt.

Fazit
Zu Beginn ist natürlich mit einem erhöhten Aufwand bei der Erstellung der Skripte für die Automatisierung zu rechnen. Im Internet gibt es für die jeweilige Provisionierungstechnik aber viele gute Beispiele, die herangezogen werden können und somit den Aufwand verringern. Zukünftig wird sich dieser Invest aber mehr als auszahlen, da Entwickler viel schneller in einem Projekt starten können und die Ausgangslage für alle Entwickler immer gleich ist. Somit kann die Entwicklung schneller und koordinierter erfolgen. Auch der Testprozess wird erheblich vereinfacht und fördert somit auch die Qualität insgesamt. Insbesondere beim Einsatz von Ansible kann die Provisionierung in Modulen organisiert werden, wodurch sich neue Image-Varianten sehr einfach und schnell zusammenstellen lassen. Oracle hat in den vergangenen Jahren seine Werkzeuge stetig verbessert, so dass eine Integration mit aktuellen Automatisierungswerkzeugen viel einfacher möglich ist.

Kontaktadresse:
Markus Lohn
esentri AG
Pforzheimer Str. 132
D-76275 Ettlingen

Telefon: +49 (0) 7243 354900
Fax: +49 (0) 7243 3549099
E-Mail markus.lohn@esentri.com
Internet: www.esentri.de