Ist Gradle auch für die APEX-Projekte?

Oleg Kiriltsev
MT AG
Balcke-Dürr-Allee 9, 40882 Ratingen

Schlüsselworte

DB, Oracle, Datenbank, CI, Build, Groovy, Continuous Integration, Gradle

Einleitung

Gradle ist ein Tool für die Automatisierung der Prozesse, wie z.B. Building, Deployment und Testing. Gradle lässt in einer deklarativen Form auf Basis von Groovy DSL die Schritte für die Erstellung / Deployment eines Projektes beschreiben. Im Vortrag werden die unterschiedlichen Möglichkeiten für den Einsatz von Gradle für die Erstellung/Deployment der datenbankbasierten Projekte, wie z.B. APEX dargestellt. Es wird mit Hilfe einer Demo gezeigt, was ist Gradle, welche Möglichkeiten bietet Gradle, wie kann man Gradle mit anderen Tools, wie Ant, Liquibase, SQLPlus kombinieren, um dieses Tool in einen bestehenden Deployment-Prozess zu integrieren oder komplett einen neuen Prozess auf zu bauen. Im Weiteren werden die Optionen für das Testen der Web-basierten Projekte mit Hilfe von Gradle und anderen Tools betrachtet.

Voraussetzungen

Nehmen wir an, wir entwickeln ein APEX-Projekt in einem Team, das aus 4 Entwicklern besteht. Wir haben mehrere Umgebungen (als Beispiel eine Integrationsumgebung und eine Testumgebung), wo alle Voraussetzungen erfüllt sind, damit unseres Projekt laufen könnte. Dazu haben wir eine oder mehrere Entwicklungsumgebungen, wo wir unsere neue Version der Anwendung entwickeln. Das sind unsere Rahmenbedingungen.

Hat ein Entwickler einen Teil der Entwicklung abgeschlossen, muss er seine Änderungen in der Integrationsumgebung bringen und nach den erfolgreichen Integrationstests, können die Änderungen in der Testumgebung installiert werden, wo unsere automatisierten Tests laufen. Damit wir alle diese Schritte ausführen könnten, müssen wir wissen:

- Was muss installiert werden?
- Wie muss installiert werden?
- Wo muss installiert werden?

Woraus besteht ein APEX-Projekt?

Damit wir die Frage – Was muss installiert werden – beantworten können, müssen wir wissen, woraus ein APEX-Projekt besteht. APEX ist ein Framework für die datenzentrierten Web-Anwendungen. In der Regel wird eine Web-Anwendung in 2 Bereiche unterteilt: Frontend und Backend. In einem APEX-Projekt wird unter dem Frontend-Teil der CSS-Code und JavaScript-Code und unter dem Backend-Teil eine Oracle Datenbank und die DB-Objekte verstanden. Darauf basierend besteht ein Standard APEX-Projekt aus einer Menge der DB-Objekte und dem JavaScript-/CSS-Code, der auf bestimmter Art und Weise in der Anwendung abgelegt und referenziert werden könnte.

Unter Umständen können die weiteren Dateien, wie z.B. CSS- und JavaScript-Dateien oder Java-Klassen einen Teil des Projektes sein. In diesem Artikel beschäftigen wir uns hauptsächlich mit den Oracle DB-Objekten.

Ausgehend davon, dass ein Standard-APEX-Projekt hauptsächlich aus der Menge der DB-Objekte besteht, müssen diese Objekte logischerweise auf der anderen Umgebung erstellt werden und einen validen Zustand haben. Abhängig davon, wie diese Objekte im Projekt verwaltet werden, werden sie in der Regel in den bestimmten Kategorien unterteilt und als Dateien abgespeichert. Die Objektkategorien werden unter anderem abhängig von den Objekteigenschaften gebildet. Die Eigenschaft, die auf jeden Fall bei der Erstellung in der anderen Umgebung beachtet werden muss, ist:

- Objekt, das immer wieder neu erstellt werden kann, unabhängig davon, ob es schon in der Datenbank vorhanden ist oder nicht
- Objekt, das vor der Neuerstellung gelöscht oder nicht erstellt werden muss, wenn es in der Datenbank schon vorhanden ist

D.h. die Objekte, die immer wieder erstellt werden können, dürfen ohne weiter nach zu denken in anderer Umgebung erstellt werden. Die Objekte, die nur erstellt werden können, wenn sie in der Umgebung noch nicht existieren, müssen entweder vor der Neuerstellung entfernt werden oder nur erstellt werden, wenn sie noch nicht existieren.

Wie sieht einen Build-Prozess von einem APEX-Projekt aus?

Nächste Frage, die beantwortet werden muss, ist: wie wir diese DB-Objekte installieren?

Im vorherigen Abschnitt haben wir festgestellt:

- Ein APEX-Projekt besteht aus der Menge der DB-Objekte, die auf bestimmter Art und Weise als Dateien abgespeichert werden
- Die DB-Objekte (und die entsprechenden Dateien) können mind. in zwei Kategorien unterteilt werden:
    - Die Objekte, die immer erstellt werden können
    - Die Objekte, die nur erstellt werden können, wenn sie in der Datenbank noch nicht vorhanden sind

Nächsten Punkt, der bei der Erstellung der DB-Objekte in der anderen Umgebung beachtet werden muss, ist der Projektstand.

Ist ein Projekt noch nicht im produktiven Einsatz, können auf der Umgebung alle DB-Objekte gelöscht werden und neu erstellt werden, spielt die Reihenfolge keine Rolle.

Ist ein Projekt schon im produktiven Einsatz und findet die Weiterentwicklung statt, müssen oder dürfen die bestimmten Objekte aus fachlichen Gründen während der Installation nicht gelöscht werden, spielt die Reihenfolge, in welcher die Objekte/Dateien auf der anderen Umgebung erstellt werden, eine wichtige Rolle. Besonders entscheidend kann es bei der Datenmigration sein. Daraus folgt, dass die Reihenfolge unter Umständen von einem Entwickler definiert werden sollte, damit die Anwendung in der anderen Umgebung erfolgreich installiert werden könnte. Die Reihenfolge kann auf unterschiedliche Art und Weise bestimmt werden:

- Entsprechende Benennung der Dateien und Ordner
- Bestimmung innerhalb Gradle als ein Groovy-Skript
- Ein Tool/ kleine Anwendung, wo die Reihenfolge der Dateien angegeben werden kann

Die Betrachtung dieser Möglichkeiten ist nicht Bestandteil dieses Artikels. Wir nehmen an, die Reihenfolge wurde definiert, in der die Dateien in der anderen Umgebung installiert / ausgeführt werden müssen.

Wenn der Build- und Installationsprozess zwei Prozessen sein sollten, und nicht unbedingt immer nacheinander durchgeführt werden, dann muss das Ergebnis des Build-Prozesses eine Datei sein, in der in richtiger Reihenfolge die zu installierende Dateien stehen. D.h. der Build-Prozess nimmt die zu installierende Dateien und packt sie in eine neue Datei in richtiger Reihenfolge.

Es bleibt nur eine Frage zu beantworten – Wo wird installiert? Diese Frage lässt sich mit Hilfe einer Konfigurationsdatei beantworten, wo die alle für die Installation nötigen Parameter/ Informationen stehen.

Welche Lösungen existieren?

Wir können die Lösungen, die schon existieren, im Großen und Ganzen in zwei Kategorien aufteilen:

- Einspielen der einzelnen Dateien „per Hand“: der Mitarbeiter hat die Entwicklung abgeschlossen und erstellt die angepassten Objekte in der anderen Umgebung / Datenbank neu
- Selbst programmierte Batch-Skripte, die für Bedürfnisse des bestimmten Projektes angepasst wurden

Wie schnell kann bei diesen Lösungen ein Fehler unterlaufen? Sehr schnell. Der Mitarbeiter vergisst irgendwelche Datei ein zu spielen, die Batch-Skripte enthalten einen Fehler – keine Möglichkeit zu debuggen, keine Kompilierungsfehler und so weiter.

Alle diese Lösungen bieten keinen etablierten und zuverlässigen Build- und Deployment-Prozess, der immer wieder in anderen DB-Projekten verwendet werden kann.

Warum Gradle?

Gradle ist ein Framework, das Groovy DSL als die Sprache für die Beschreibung eines Installationsprozesses beliebiger Komplexität anbietet. Die Gradle gibt eine Möglichkeit die Installationsschritte wieder zu verwenden als sogenannte Plugins. Die Gradle-Entwickler und Community stellen viele Plugins mit entsprechenden Installationsschritten bereit, die nur mit einem Befehl in einen Build / Deployment-Prozess integriert werden können. Dadurch, dass die Installationsschritte mit Hilfe der richtigen Programmiersprache definiert werden, bietet Gradle die Möglichkeit den Installationsprozess völlig den eigenen Bedürfnissen anzupassen.

Alternativen

Als Alternativen können die Build-Systemen oder Framework aus der JavaScript-Welt betrachtet werden. Als bekanntesten können hier die folgenden genannt werden:

- Gulp
- Grunt
- Node.js & Co.

Die Build-Systeme aus der JS-Welt sind mehr auf die JavaScript-Projekte gerichtet und für die typischen Aufgaben im JS-Bereich gedacht. Node.js & Co. bieten die große Menge der Erweiterungen und Plugins, die auch von Node.js-Entwicklern und Community bereitgestellt werden.

Fazit und Ausblick

Im Artikel wurde einen Deployment-Prozess von einem APEX-Projekt betrachtet. Es wurden die wichtigen Punkte von einem Prozess erläutert:

- Objekttyp
- Reihenfolge

Als nächsten Punkt wurde ein Build-Tool „Gradle“ vorgestellt und beschrieben, ob mit Hilfe dieses Tools die APEX-Projekte erfolgreich erstellt werden können. Es wurden kurz die Alternativen aus der JavaScript-Welt dargestellt, die sicherlich für die Bedürfnisse eines Projektes angepasst werden können.

Als sinnvolle Erweiterung werden die folgenden Punkte angesehen:

- Kapselung und Erstellung eines Plugins
- Einbindung des Installationsprozesses im Continuous Integration Zyklus
- Integration der Testing-Framework (wie Selenium, Unit Tests etc.) in Gradle

Kontaktadresse:

Oleg Kiriltsev

MT AG

Balcke-Dürr-Allee 9

D-40882 Ratingen

Telefon:	+49 (0) 2102-30961 0
Fax:	+49 (0) 2102-30961 101
E-Mail:	oleg.kiriltsev@mt-ag.com
Twitter:	@OKiriltsev
Internet:	www.mt-ag.com