Software zunächst auf diese Kopie eingespielt. Während dieses Vorgangs läuft die Datenbank ohne Downtime. Ist die neue Software jetzt installiert, wird die Datenbank herunter- und mit der neuen Software wieder hochgefahren. Je nach Patch werden dann noch SQL-Skripte gestartet. Diese Patch-Methode findet auch in den neuen Patchsets der Datenbank Anwendung.

Fazit

Das Cloud Control ist notwendig, um komplexe Umgebungen zu überwachen, und zeigt seinen Wert erst Recht im Einsatz des Provisioning. Mit den vorgefertigten Bausteinen von Venisolvere wird der Übergang zu DevOps im Oracle-Fusion-Middleware-Umfeld leichter.

Andreas Chatziantoniou
andreas@foxglove-it.nl

Tobias Weih
obias.weih@cabp.de

Dirk Ruchatz
itc-dr@gmx.de

Production Redeployment von ADF-Anwendungen

Ulrich Gerkmann-Bartels und Andreas Koop, enpit

Durch die Einführung von agilen Vorgehensweisen sind nachweislich Ergebnisse aus der Software-Entwicklung früher sichtbar und auch nutzbar. In vielen Fällen, in denen „Feature Driven Development“ und „Continuous Integration“ erfolgreich im Einsatz sind, zeigen sich jedoch manchmal neue Hindernisse aus Architektur- und Betriebssicht.

Zwei häufig auftauchende Herausforderungen sind der Anwendungsschnitt einer Applikation und die Bereitstellung von neuen Features oder Modulen mit minimaler Unterbrechung der im Einsatz befindlichen Anwendung. Der Artikel beschreibt, welche Voraussetzungen und Vorgehensweisen im Deployment von ADF-Anwendungen eingeführt werden können, um ein Production Redeployment dieser Anwendungen und Bibliotheken einzuführen.

Ausgangslage

Möchte man Flexibilität, Granularität und Änderungshäufigkeit in der Software-Entwicklung mit der Bereitstellung und Hochverfügbarkeit einer Anwendung im Betrieb kombinieren, treffen zwei unterschiedliche Zielvorstellungen aufeinander. In der Software-Entwicklung möchte man nach agilem Prinzip oftmals wie folgt verfahren: „Das eine Feature bringe ich noch rein“, frei übersetzt nach dem Ausspruch „Den Bug behebe ich noch“, um die Fachlichkeit laufend mit Funktionalität zu erfreuen. Demgegenüber stehen betriebliche Aspekte, die vielleicht nach dem Motto „Never Change a running System!“ ihre bestehenden Systeme mit Sicherheit zu schützen wissen. Neben der methodischen Vorgehensweise, „DevOps“ in Unternehmen einzuführen, gilt es zu erörtern, welche Möglichkeiten der Oracle-Fusion-Middleware-Stack bietet, insbesondere WebLogic Server und

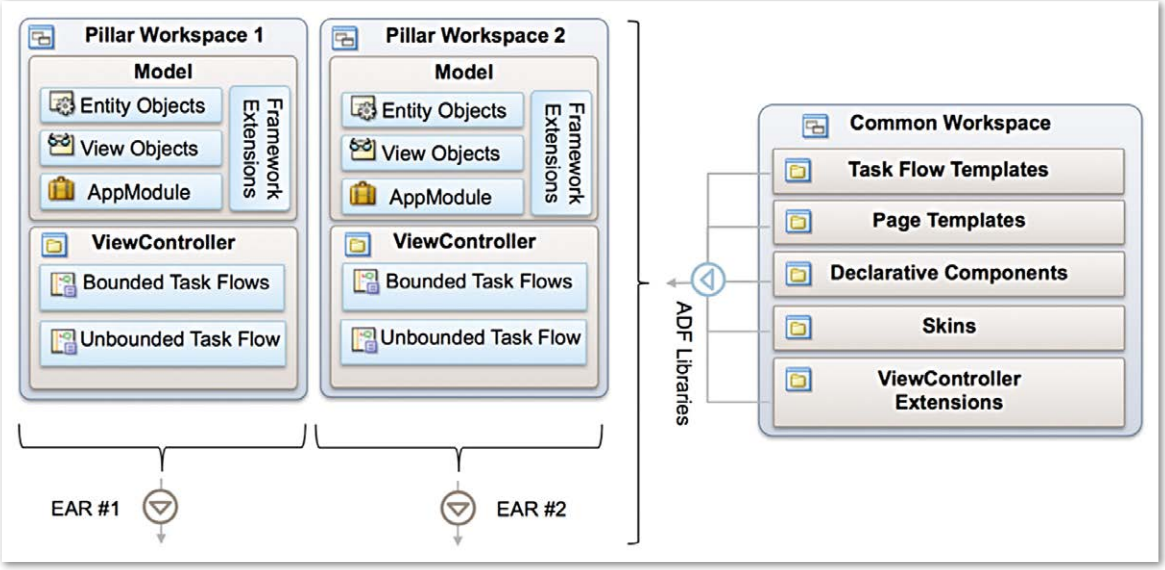

Abbildung 1: Pillar-Architektur „ADFArchitecture Fundamentals" (siehe „http://www.youtube.com/watch?v=toEuQvp73h8", Chris Muir)

Oracle ADF, um eine sichere, modulare und im Zweifel zurücksetzbare Anwendungsbereitstellung umzusetzen.

Architektur

Möchte man Flexibilität, Granularität und Änderungshäufigkeit in der Anwendungsentwicklung nutzen, ist es hilfreich, sich grundsätzlich einige Gedanken darüber zu machen, wie die Applikation und die beinhalteten Module aus dem Blickwinkel der Informations-Architektur und aus Sicht der Strukturierung der Workspaces/ Projekte geschnitten sind:

- Was lässt sich auf welcher Ebene modularisieren?
- Inwieweit beherrschen technische Aspekte den fachlichen Schnitt?
- Bewegt sich die Software-Architektur hinsichtlich der bereitzustellenden Artefakte zu einem Riesenmonolithen in der Ausprägung eines 300 MB Enterprise Archiv Files (EAR)?
- Wie können andere funktionale Bausteine in Projekten wiederverwendet werden und gegebenenfalls ihren eigenen Lebenszyklus haben?

Eine Architektur, die bezüglich dieser Aspekte grundsätzlich eine Antwort gibt, ist die sogenannte Pillar-Architektur *(siehe Abbildung 1)*.

Dieser Entwurf basiert auf dem Ansatz, dass fachliche Module zu einem Enterprise Archiv (EAR) gebündelt und durch eine entsprechende ADF-Library mit allgemeinen, nicht fachlichen Komponenten versorgt werden. Die jeweiligen Enterprise Archive sind unabhängig voneinander bereitstellbar und unterliegen ihrem eigenen Lebenszyklus. Ein gemeinsamer Kontext zwischen den Modulen kann gegebenenfalls durch einen „ContextStore" in der Datenbank oder durch einen Coherence-Cluster abgebildet werden *[1]*. Ausgehend von dieser Architektur stellen sich für das Deployment der Anwendung folgende Fragen:

- *Ist es möglich, ein neues fachliches Modul in Form eines EAR im laufenden Betrieb bereitzustellen?*
  Ja, durch die Option „Production Redeployment"
- *Ist es möglich, die nicht fachlichen Komponenten einer ADF Library für alle Applikationen zentralisiert zur Verfügung zu stellen, um bei neuen Features und Fehlerbehebungen nicht alle Anwendungen neu zu bauen und bereitzustellen?*
  Ja, durch den Einsatz von „WebLogic Shared Library"
- *Ist es möglich, verschiedene Versionen einer WebLogic Shared Library bereitzustellen?*
  Ja, durch den expliziten Verweis auf die genutzte Version in der entsprechenden Applikation

Production Redeployment

Production Redeployment ermöglicht es, im WebLogic Server zwei Versionen einer Anwendung gleichzeitig durch die Angabe unterschiedlicher und aufsteigender Versionsnummern auszuführen. Um die Versionierung von Applikationen im WebLogic Server zu aktivieren, muss dem Archiv

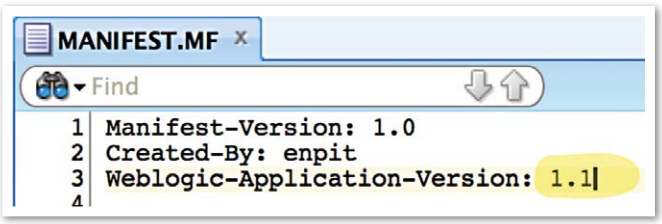

Abbildung 2: Manifest mit WebLogic-Application-Version

☐	enpit-common-war-lib(1.0,1.0.5)	Active		Library
☐	enpit-common-war-lib(1.0,1.0.6)	Active		Library
☐	enpittestcommons-reflib.war (1.0)	Retired		Web Application
☐	enpittestcommons-reflib.war (1.1)	Active	OK	Web Application

Abbildung 3: Retired Web Application nach Timeout der letzten Sessions

ein entsprechendes Manifest unter „/src/META-INF/MANIFEST.MF“ mit den notwendigen Properties hinzugefügt werden *(siehe Abbildung 2)*.

Vor dem Deployment muss sichergestellt sein, dass eine Applikation mit derselben Versionsnummer nicht existiert. Sollte dies der Fall sein, lässt sich die Anwendung nicht bereitstellen. Durch zusätzliche Parameter kann die Version auch beim Deployment beeinflusst werden. Aktive Anwendungssitzungen werden bei der Bereitstellung mit der vorherigen Version weitergeführt. Es ist kein Neustart der gesamten Anwendung notwendig *(siehe Abbildung 3)*.

Nach Ablauf aller Anwendungssitzungen der Anwendung in Version 1.0 wird der Status der Applikation in den Modus „Retired“ versetzt. In diesem Zustand nimmt die Anwendung keine neuen Sessions mehr an. Diese werden ausschließlich von der Anwendung in Version 1.1 bedient *(siehe Abbildung 4)*.

Um im nächsten Release-Zyklus eine neue Version (1.2) zur Verfügung zu stellen, muss die Anwendung mit dem Status „Retired“ entfernt werden. Dies kann leicht vor dem Deployment automatisiert durch ein WLST-Script erfolgen, das über alle Applikationen mit dem gewünschten Applikationsnamen iteriert und sich die Version der entsprechenden Retired-Applikation merkt. Dies ist unbedingt notwendig, da zur Entfernung einer versionierten Anwendung die entsprechende Versionsnummer beim „Undeployment“ notwendig ist.

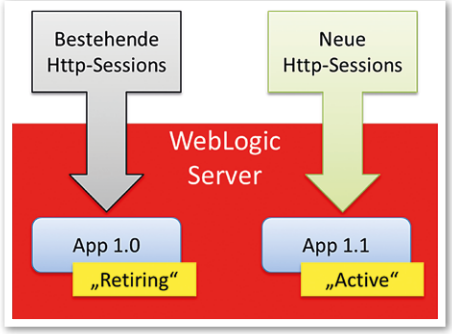

Abbildung 4: Production Redeployment – oder auch Side-by-Side-Deployment

WebLogic Shared Library

Neben versionierten Deployments von Applikationen erlaubt der WebLogic Server auch die Versionierung sogenannter „Shared Libraries“. Wie der Name schon verrät, lassen sich damit Bibliotheken in einer WebLogic-Domain zentral zur Verwendung durch mehrere Applikationen bereitstellen. Im Kontext von ADF-Anwendungen lassen sich damit einzelne fachliche Module oder auch Module mit Querschnittsfunktionalitäten als Shared-ADF-Library bereitstellen. Ähnlich den versionierten Deployments von EARs beziehungsweise WARs können auch die Bibliotheken versioniert bereitgestellt sein. Um eine Shared Library bereitzustellen, gehe man wie folgt vor:

- Erstellung einer „/META-INF/MANIFEST.MF“ mit folgenden Eigenschaften: „Extension-Name: enpit-common-war-lib“, „Specification-Version: 1.0“ und „Implementation-Version: 1.0.5“
- Paketierung des gewünschten Bibliotheksinhalts als WAR. Dies kann beispielsweise über ein JDeveloper-Deploymentprofil erfolgen oder auch mit jedem beliebigen anderen Tool. Entscheidend ist, dass die unter Punkt 1 genannten Metadaten enthalten sind. Die Best Practice bei ADF-Libraries ist, dass das resultierende JAR im Shared-Library-WAR unter „WEB-INF/lib“ gebündelt ist. Die Shared-WAR-Library kann in diesem Zusammenhang auch als „Proxy-Library“ für die ADF-Library bezeichnet werden.
- Deployment des WAR als Bibliothek im WebLogic Server. Je nach eingesetzter Technik muss beim Deployment angegeben werden, dass es sich um eine Library handelt. Am Beispiel des „weblogic.Deployer“-Kommandozeilen-Werkzeugs sieht es dann wie folgt aus: „java weblogic.Deployer -adminurl t3://eden.local:7001 -username weblogic -password welcome1 -upload -library -targets AdminServer -deploy -source enpit-shared-lib-war.war“.

Als Ergebnis erhält man eine Shared Library, deren Version und Verfügbarkeit in der Admin Console verifiziert werden kann *(siehe Abbildung 5)*.

Zur Verwendung einer Shared Library muss die entsprechende Applikation eine Library-Reference im WebLogic-spezifischen Deployment-Deskriptor spezifiziert sein („weblogic.xml“, siehe *Listing 1*).

Optional können Spezifikations- und Implementierungs-Versionsnummern mitgegeben werden. Fehlen diese, bedeutet dies, dass die Applikation stets die Bibliothek mit der höchsten Versionsnummer verwenden soll. Um neue Versionen bereitzustellen, muss die Version in der „MANIFEST.MF“ geändert und das anschließend paketierte WAR mit dem Standard-Library-Deployment-Befehl bereitgestellt sein. Man kann auf diese Weise beliebig viele Versionen einer Shared Library installieren (beim Application Production Redeployment können maximal zwei Versionen aktiv sein). Zu beachten sind für Shared Libraries folgende Bedingungen/Einschränkungen:

- Eine Bibliotheksversion kann nicht entfernt werden, solange mindestens eine Applikation darauf verweist
- Alle referenzierten Bibliotheken müssen auf denselben Servern bereitgestellt sein, auf denen die Applikation installiert ist
- Eine neue Bibliotheksversion wird für eine Anwendung, die die Bibliothek ohne Versionsnummern referenziert, erst maßgeblich, sobald die Anwendung erneut gestartet wird

Bibliotheken, die ein ADF-Library-JAR in „/WEB-INF/lib“ enthalten, müssen unbedingt per „library-ref“ in der „weblogic.

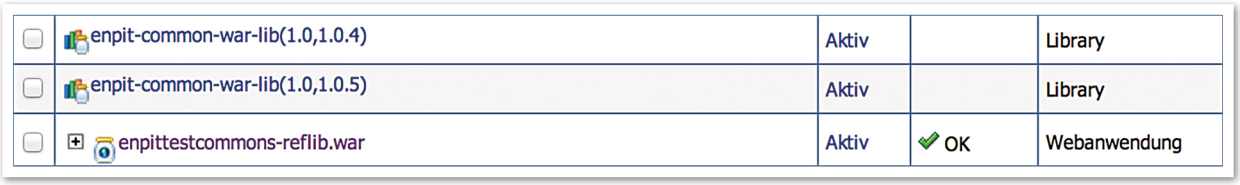

Abbildung 5: Shared-Library-Übersicht in der Admin-Console

```
<?xml version='1.0' encoding='UTF-8'?>
<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-web-app.
xsd" xmlns="http://www.bea.com/ns/weblogic/weblogic-web-app">
  <library-ref>
       <library-name>enpit-common-war-lib</library-name>
  </library-ref>
</weblogic-web-app>
```

Listing 1

xml" und nicht in der „weblogic-application.xml" referenziert sein. Andernfalls gibt es eine Exception zum Zeitpunkt des Deployments *[2]*. Weitere Einzelheiten und Beispiele zu Shared Libraries im WebLogic finden sich im Blog-Beitrag unter *[3]*.

Möglichkeiten

Kombiniert man jetzt alle Möglichkeiten, so erlaubt diese Methode auch die Realisierung fachlicher Module, die Anwendungen von außen integrieren. Die Pillar-Architektur kommt an dieser Stelle an ihre Grenzen. Realisiert mit ADF-Task-Flows und verpackt in einer ADF-Library, die durch eine WebLogic Shared Library der konsumierenden Anwendung zur Verfügung gestellt wird, ist dies möglich. Die entsprechende WebLogic Shared Library dient als Proxy für die nutzende Anwendung. Ein abgestimmter Build- und Deployment-Prozess führt eine modulare und flexible Bereitstellung durch. Dass das alles nicht nur blanke Theorie ist, zeigt die Entwicklung eigener ADF-Task-Flows für WebCenter Portal ab der Version 11.1.1.8, siehe *[4]*.

Fazit

Wie bei allen Dingen gibt es natürlich auch bei Nutzung der skizzierten Vorgehensweisen Grenzen, die man nicht unterschätzen sollte. Insbesondere die Herausforderung, zwei gleiche Versionen einer Software für einen bestimmten Zeitraum zur Verfügung zu stellen, oder gegebenenfalls das Zurückfahren auf eine vorherige Version kann erhebliche Komplexität beinhalten. Die Grenzen beginnen natürlich bei der Integration und hier sicherlich als Erstes im Backend. Insbesondere, wenn es sich dabei um Anwendungen handelt, die die Datenbank weitreichend für Logik und Verarbeitung nutzen.

Basieren die neuen Versionen der einzelnen Bausteine und Applikationen auf entsprechenden Datenbank-Schemata-Änderungen im Backend, so sind diese entsprechend in eine Rollback-Strategie mit einzuplanen. Unter Umständen kann eine gleichzeitige Nutzung verschiedener Versionen bei einer Änderung im Backend auch ausgeschlossen sein. In diesem Fall bleibt immer noch die Möglichkeit, eine entsprechende Bereitstellung in einem Wartungsfenster durchzuführen. Der Vorteil, die Artefakte durch die eingeführte Versionsnummer identifizieren zu können, bleibt bestehen.

Bestehen Abhängigkeiten zu Web-Services- oder Rest-Schnittstellen, sind diese ebenso zu berücksichtigen, können aber mit entsprechenden Konzepten bezüglich der Versionierung und des Lifecycle durch eine vorhandene Governance geregelt werden und ermöglichen die Verfügbarkeit verschiedener Versionen zur Nutzung in den jeweiligen Modulen.

Nicht zuletzt sollte man sich gut überlegen, wieweit man die Modularisierung der einzelnen Module in eigene WebLogic Shared Libraries treibt, sonst hat man sich eines Tages einen eigenen DLL-Hell *[5]* erschaffen.

Referenzen und weitere Hinweise

[1] Continuous Delivery in ADF-Projekten: http://de.slideshare.net/ugb/20130619-continuousdeliveryinadfprojekten (2013)

[2] Production Redeployment with ADF Shared WebLogic Libraries: http://multikoop.blogspot.de/2013/06/production-redeployment-with-adf-shared.html

[3] WebLogic Application redeployment using shared libraries - without downtime: http://multikoop.blogspot.de/2013/06/weblogic-application-redeployment-using.html

[4] S. 1340 ff., Developing Portals with Oracle WebCenter Portal and Oracle JDeveloper, 11gR1 (11.1.1.8), E27739-01, Juli 2013

[5] DLL-Hell: http://en.wikipedia.org/wiki/DLL_Hell

[6] Deploying Applications to Oracle WebLogic Server 11g Release 1 (10.3.6): http://docs.oracle.com/cd/E23943_01/web.1111/e13702/redeploy.htm

[7] WebLogic Tutorial for Production Redeployment: http://www.oracle.com/webfolder/technetwork/tutorials/obe/fmw/wls/12c/14-Redeployment--4465/redeploy.htm

Ulrich Gerkmann-Bartels
ulrich.gerkmann-bartels@enpit.de

Dipl.-Inf. Andreas Koop
andreas.koop@enpit.de