Metadata Service Respository – sehen, lernen, verstehen

Carsten Wiesbaum, esentri AG

Früher oder später wird jeder, der sich im Umfeld der Oracle Fusion Middleware bewegt, einen Kontaktpunkt mit dem Metadata Service Repository haben – bewusst oder unbewusst. Oft lernt man dabei nur eine Facette dieses Bausteins der Oracle Fusion Middleware kennen, etwa beim ADF UI Customizing oder beim Deployment von SOA Composites. Dabei ist das Metadata Service Repository die Grundlage für eine Vielzahl von Funktionen im gesamten Oracle-Fusion-Middleware-Stack.

Der Artikel zeigt, was das Metadata Service Repository (MDS) genau ist, wo es innerhalb der Fusion Middleware eingesetzt wird und wie man selbst von der Funktionalität profitieren und sie im eigenen Projektalltag und Systemdesign einbinden kann.

In der heutigen Software-Entwicklung spielen Metadaten eine große Rolle. Nahezu jede Anwendung verwendet Metadaten in der einen oder anderen Form. Als Metadaten bezeichnet man Daten über Daten, im Bereich der Enterprise-Anwendungen auch Daten über Anwendungen. Große Teile und ganze Enterprise-Anwendungen werden nicht mehr in einer Programmiersprache entwickelt, sondern über Metadaten konfiguriert. Diese Metadaten ermöglichen erst das Verhalten bestehender Komponenten beziehungsweise deren Funktion.

Prominente Beispiele für die Verwendung von Metadaten in der heutigen IT sind zum Beispiel JSF-Seiten, WSDL-Dokumente oder auch XML-Schemata (XSDs). Wenn man die Bedeutsamkeit von Metadaten für heutige Enterprise-Anwendungen betrachtet, wird schnell klar, dass ein zentraler Ansatz zur Verwaltung und Verwendung von Metadaten notwendig ist. Innerhalb des Oracle-Fusion-Middleware-Stacks übernimmt das MDS genau diese Funktion.

Metadata Service Repository als Bindeglied der Oracle Fusion Middleware

Das MDS stellt eine zentrale Komponente der Oracle Fusion Middleware dar. Seine Aufgabe ist die zentrale Verwaltung der Konfigurationsdateien von Komponenten sowie Metadaten zu Applikationen. Die im MDS abgelegten Daten werden durch den kompletten Oracle-Fusion-Middleware-Stack verwendet. So verwalten SOA Suite und Oracle Identity Manager ihre Konfigurationen über das MDS. Auch die SCA-Anwendungen der SOA Suite sind im MDS-Repository abgelegt. So finden sich hier BPEL-Prozess, Business Rules und DVMs.

Darüber hinaus basieren viele Funktionen des Application Development Framework (ADF) auf der Nutzung des MDS-Repository. Gerade die Funktionen rund um Customizing und Personalization verwenden das Repository exzessiv. Während der Design-Time werden die ursprünglichen Metadaten, die die Funktion der ADF-Anwendung beschreiben, in Oracle JDeveloper erzeugt. Die generierten Metadaten reichen durch alle Schichten des ADF-MVC-Modells von Business Components bis hin zu den JSF-Seiten.

Beim Customizing einer Anwendung generiert der entsprechende Entwickler weitere Metadaten. Diese werden je nach Bedarf in der laufenden Anwendung verwendet, um das Aussehen und Verhalten der Applikation an bestimmte Benutzer oder Benutzergruppe anzupassen. Zudem kann dem Benutzer die Möglichkeit eingeräumt werden, über „Personalization“ weitere Metadaten zur Laufzeit zu generieren, die wiederum im MDS abgelegt sind.

Auch viele Funktionen der Fusion Applications und Oracle WebCenter bauen auf den Funktionen des MDS-Repository auf. Sie verwenden unter anderem ADF und SOA Suite als Basis-Technologie und bieten fortgeschrittene Customization, Personalization sowie „Design Time at Run Time (DT@RT)“-Funktionen an *(siehe Abbildung 1)*. Diese Beispiele zeigen die starke Integration des MDS-Repository in der Fusion Middleware.

Technische Grundlagen

Im Grunde besteht das MDS-Repository aus drei Komponenten – einem Datenspeicher, einer Runtime Engine und einigen MBeans zur Interaktion mit dem MDS-Repository über das WebLogic Sever Scripting Tool (WLST) oder Enterprise Manager Fusion Middleware Control (EM). Der Datenspeicher kann sowohl Datei- als auch Datenbank-basiert genutzt werden *(siehe Abbildung 2)*. Für eine Entwicklungsumgebung reicht die Speicherung der Metadaten auf Dateisystem-Ebene völlig aus. Oracle JDeveloper besitzt standardmäßig eine entsprechende Verzeichnis-Struktur, die während der Entwicklung verwendet werden kann.

Für den produktiven Einsatz sollte die Datenbank-basierte Variante zum Einsatz kommen. Sie bietet eine bessere Performance sowie die Versionierung der abgelegten Artefakte und verwendet Datenbank-Transaktionen, um die Integrität der abgelegten Metadaten zu gewährleisten. Auch für den Cluster-Betrieb bietet diese Variante Vorteile, so können mehrere Ins-

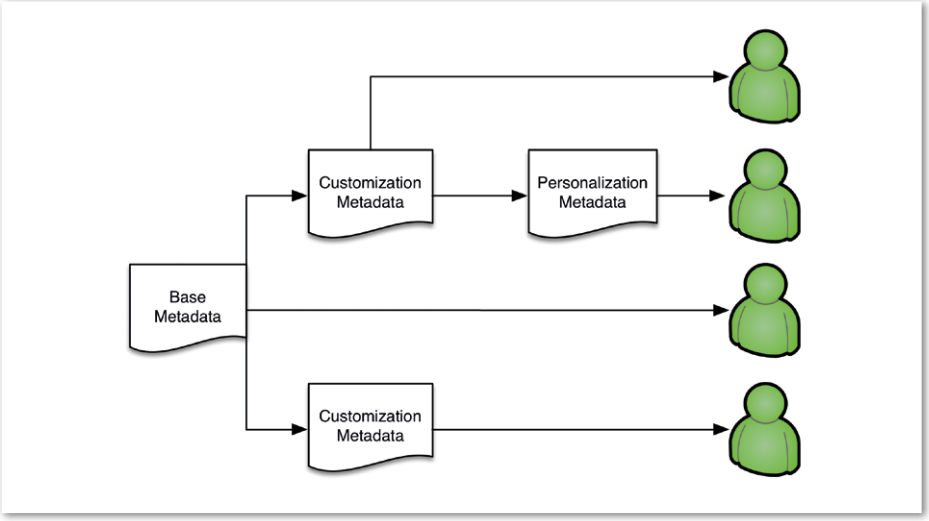

Abbildung 1: ADF Customization und Personalization

tanzen einer ADF-Anwendung das gleiche MDS-Repository verwenden.

Wenn in einer Anwendung Customization-Änderungen durchgeführt werden, werden diese durch Polling auch auf anderen Instanzen der Anwendung sichtbar. Allerdings muss bei der Nutzung der Datenbank-basierten Variante natürlich auch zusätzlicher Aufwand betrieben werden. Es muss beispielsweise geklärt sein, wie viele Repositories notwendig sind und welche Purging-Strategie für die gewählte Architektur genutzt wird.

Innerhalb des MDS-Repository sind die Metadaten in einer Verzeichnis-Struktur abgelegt. Jede abgelegte Datei ist einem bestimmten Pfad zuzuordnen. Zusätzlich kann das MDS-Repository noch in mehrere Partitionen aufgeteilt sein. Partitionen ermöglichen die logische Trennung von Metadaten entsprechend ihrem Nutzen. Im Hinblick auf ADF-Anwendungen gilt es etwa als Best Practice, für jede Anwendung eine eigene Partition anzulegen. Durch diese Strategie sind die Metadaten einzelner Anwendungen voneinander getrennt.

Die zweite Komponente des MDS-Repository ist die Runtime Engine. Diese bietet dem Fusion-Middleware-Stack und dessen Anwendungen die Interaktion mit den Metadaten innerhalb des MDS-Repository. Sie kontrolliert den Zugriff auf darin enthaltene Metadaten und ermöglicht das Importieren von Metadata-Archiv-Dateien (MAR). Außerdem beinhaltet die MDS-Runtime-Engine einen integrierten Cache, der den Zugriff auf häufig angefragte Metadaten optimiert.

Bei der letzten Komponente handelt es sich um MDS-spezifische MBeans, die zur Verwaltung des MDS verwendet werden können. Sie ermöglichen das Entleeren des MDS-Runtime-Engine-Cache, den Im- und Export von anwendungsspezifischen Metadaten sowie die Konfiguration einiger MDS-Repository-Funktionen *(siehe Abbildung 3)*.

Das MDS-Repository in der Service-Entwicklung

Bisher wurde aufgezeigt, wie tief das MDS-Repository im Fusion-Middleware-Stack verankert ist. Gerade im Bereich der SOA Suite besteht jedoch auch die Möglichkeit, es aktiv zu nutzen und in seine Service-Architektur einzubinden. Zunächst einmal kann das MDS-Repository als zentrales

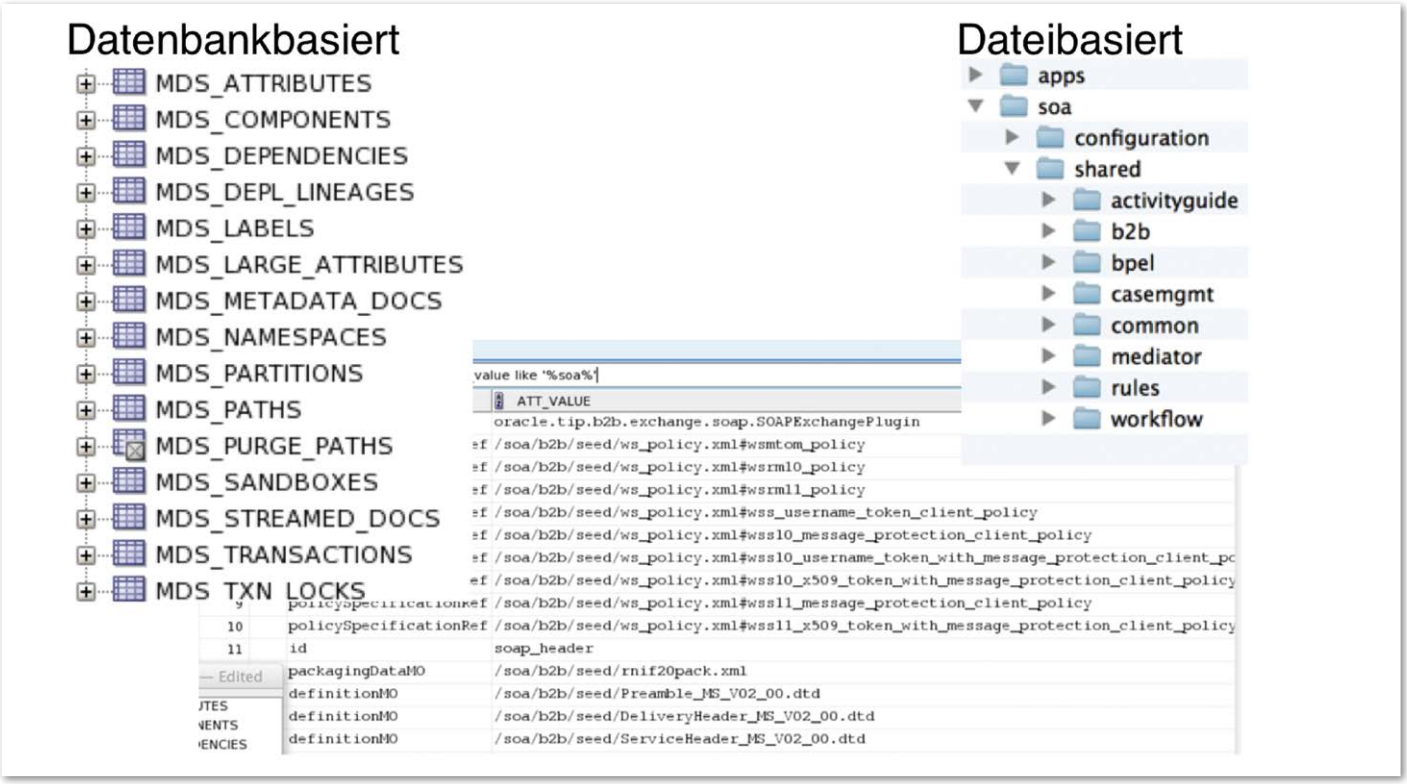

Abbildung 2: Datenbank- und Datei-basierte Variante des MDS-Repository

Repository für die Serviceschnittstellen-Definitionen zum Einsatz kommen *(siehe Abbildung 4)*.

Hier sind WSDL-, XSD- und XSLT-Dateien in einem zusätzlichen Projekt abgelegt. Dieses wird dann in das MDS-Repository eingerichtet. Die Dateien sind anschließend unterhalb des MDS-Repository im „apps"-Ordner zu finden. SCAs und deren Komponenten können diese Artefakte dann über „oramds:/apps/Verzeichnis_1/Verzeichnis_2/fooService.wsdl" referenzieren.

Eine weitere Möglichkeit, das MDS-Repository zu nutzen, ist die Verwendung von Domain-Value-Maps (DVMs) und Cross References (XRef). Innerhalb der Service-Entwicklung werden diese Dateien verwendet, um ein Mapping zwischen verschiedenen Werten zu ermöglichen. Sowohl DVMs als auch XRef -Dateien werden beim Deployment eines SCA im MDS-Repository abgelegt. Die darin enthaltenen Werte lassen sich dann über Xpath-Funktionen oder innerhalb von Java-Code abfragen.

Ein Anwendungsbeispiel

Für MDS-Zugriff auf eine DVM mit Java wird das MDS-Repository verwendet, um konfigurierbare Werte im MDS in Form von DVMs abzulegen. Dies ermöglicht es, die Werte innerhalb einer Anwendung auch während der Laufzeit zu ändern, und das ohne ein erneutes Deployment der Anwendung auf dem Server. Als konkretes Beispiel dient hier die Ablage und Konfiguration von Fehler-Informationen. Ziel ist es, Informationen wie Fehlercode, Fehlername und Fehlerbeschreibung als DVM im MDS abzulegen und aus Java darauf zuzugreifen. Um dies zu erreichen, wurde die DVM aus *Abbildung 5* im MDS abgelegt. *Abbildung 6* zeigt die Ablage-Struktur innerhalb des MDS.

Aus einer anderen Komponente werden die Fehler-IDs in der ersten Spalte geliefert. Um dem Benutzer der Anwendung eine möglichst aussagekräftige Fehlermeldung anzuzeigen, werden die übrigen Werte mit der ID aus der DVM abgefragt. *Listing 1* zeigt, wie Java-Werte von der DVM „Errors.dvm" innerhalb des MDS abgefragt werden.

Im Code werden die relevanten Werte aus der DVM abgefragt und in einer Fehlermeldung dem Anwender ange-

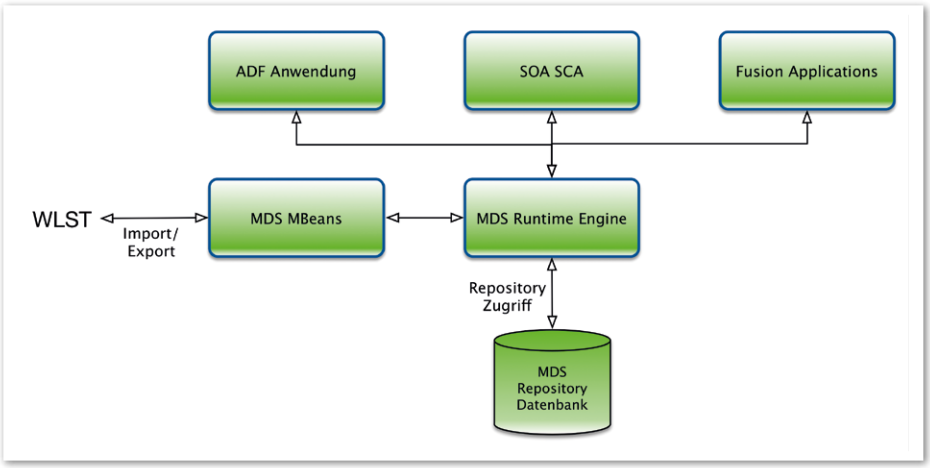

Abbildung 3: MDS Repository Runtime Architecture

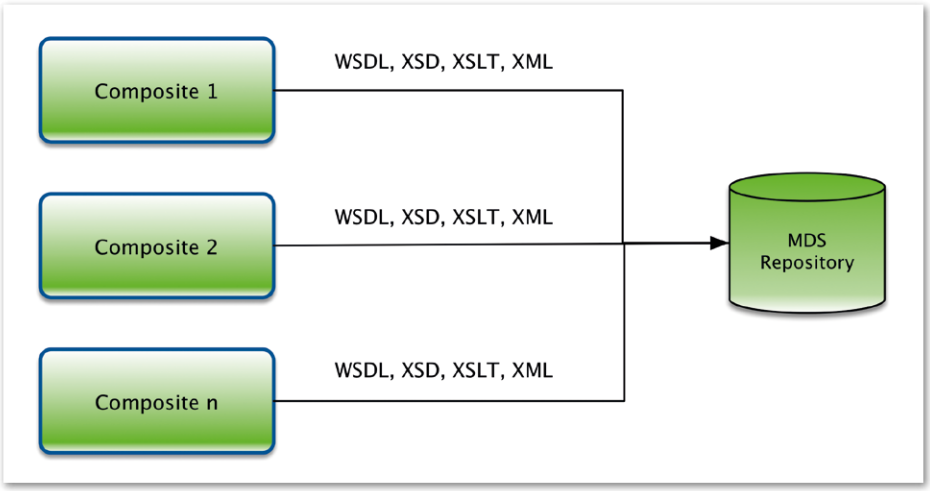

Abbildung 4: Verwendung des MDS-Repository in der Service-Entwicklung

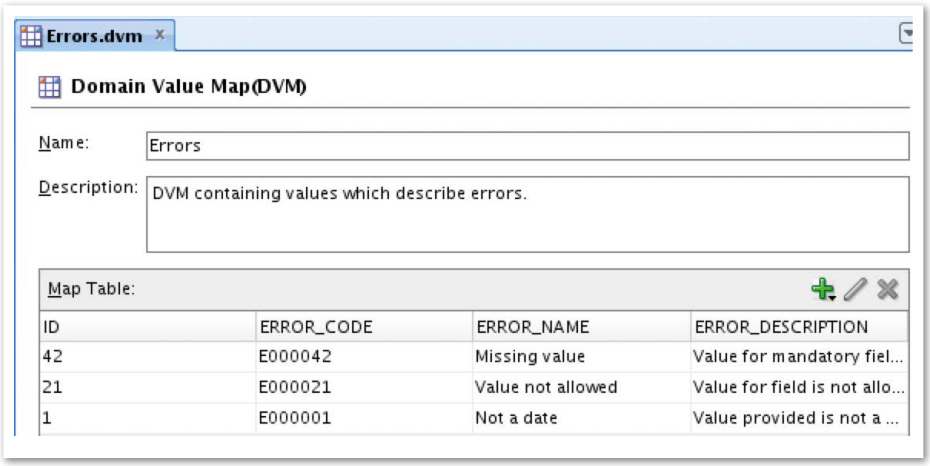

Abbildung 5: DVM mit Fehler-Informationen

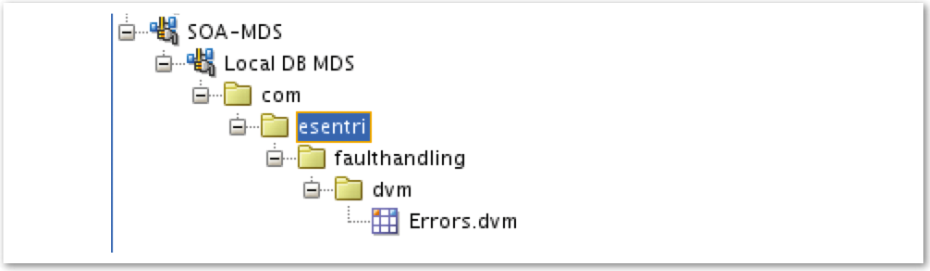

Abbildung 6: Im MDS-Repository abgelegte DVM

```
DVMManager dvmManager = DVMManagerFactory.getDVMManager();

String errorCode;
String errorName;
String errorDescription;

try {
   errorCode = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_CODE", null);

   errorName = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_NAME", null);

   errorDescription = dvmManager.lookupValue(
               "oramds:/com/esentri/faulthandling/dvm/Errors.dvm"),
               "ID", "42",
               "ERROR_DESCRIPTION", null);
} catch (DVMException e) {
   // Handle Exception
}
```

Listing 1

zeigt. Sollte sich ein bestimmter Wert ändern, ist dieser lediglich in der DVM innerhalb des MDS-Repository zu ändern. Beim nächsten Zugriff auf die DVM verwendet die Anwendung den neuen Wert.

Fazit

Das MDS-Repository wird oftmals als eine einfache Komponente innerhalb des Fusion-Middleware-Stacks dargestellt. Wie beschrieben, stellt sie jedoch einen zentralen Bestandteil für die Funktionsweise und Interaktion vieler Oracle-Produkte dar, sowohl während der Entwicklung als auch zur Laufzeit. Daher ist bei der Entwicklung von Enterprise-Applikationen auf Basis von Oracle-Produkten ein grundlegendes Verständnis der MDS-Repository-Architektur und der damit verbundenen Prinzipien von großem Wert.

Carsten Wiesbaum
carsten.wiesbaum@esentri.com

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Alexander Alers
Henriette Cebulla
Klaus Debus
Christina Eicher
Richard Haberkorn
André Pittel
Per Reinholdt
Gerd Schoen
Hermann von Kobylinski
Thorsten Ahlbrecht
Artur Czudnochowski
Martin Götz
Houssem Eddine Hariz
Christian Höcherl

Firmenmitglieder DOAG

Kundan Lamsal, Cleverbridge AG
Friedrich Mayerhofer, GRZ IT Center GmbH
Klaus Rieck, S&H EDV-Beratung GmbH
Erwin Rossgoderer, ISE GmbH
Martin Waiblinger, TransnetBW GmbH

Neumitglieder SOUG

Richard Koller, CSS Versicherung
Laszlo Hadhazy, Accarda AG
Christina Pavalache, Diso AG
Karl-Heinz Sütterlin, Microsoft Schweiz GmbH
Jacques Kostic, Trivadis AG
Thomas Rieder, Atos AG
Dr. Stefan Meyer, Prolicense Schweiz GmbH
Tobias Schaller, Zürich Kantonalbank
Marco Kurmann, bbi software ag