SAP PRESS

SAP® Gateway und OData

- Installation, Konfiguration und Systemanbindung
- OData-Services erstellen und erweitern
- SAP Fiori, mobile Apps und Unternehmens-anwendungen anbinden

3., aktualisierte und erweiterte Auflage

Bönnen · Drees · Fischer
Heinz · Strothmann

Rheinwerk Publishing

Kapitel 5

Einführung in die Erstellung von OData-Services mit SAP Gateway

Die Serviceentwicklung ist eine der zwei wesentlichen Methoden zur Erstellung von OData-Services. Wenn Sie Services selbst entwickeln, sind Sie grundsätzlich flexibler als bei der Servicegenerierung, der zweiten Methode.

Wie Sie in Kapitel 2, »Einführung in OData«, und Kapitel 3, »Architektur und Integration«, gesehen haben, werden OData-Services im SAP-Backend-System implementiert und als URI über den Gateway-Server publiziert, der einen Zugriff auf die Daten des SAP-Backend-Systems ermöglicht.

Die Anzahl der OData-Services, die mit SAP Gateway als Teil des Standards ausgeliefert werden, ist klein. Dies wird sich auch in Zukunft nicht ändern, da OData-Services von Natur aus sehr granular und auf einen bestimmten Anwendungsfall zugeschnitten sind. OData-Services, die auf der Gateway-Technologie basieren, werden daher als Teil von SAP-Produkten wie SAP Fiori, SAP S/4HANA oder mobilen SAP-Anwendungen ausgeliefert.

Bei der Entwicklung von Anwendungen ist es häufig so, dass ein großer Teil der gesamten Entwicklungszeit in die Erstellung der geeigneten OData-Services investiert werden muss. Daher ist ein gutes Verständnis des Prozesses der Serviceerstellung besonders wichtig.

SAP Gateway Service Builder

Das zentrale Werkzeug, um Services in SAP Gateway zu definieren und zu implementieren, ist der *SAP Gateway Service Builder* (Service Builder), der mit der Transaktion SEGW gestartet wird. Nachdem Sie einen Service im Service Builder erstellt und diesen auf dem Gateway-Server publiziert haben, kann dieser direkt in jedem beliebigen Konsumenten verwendet werden. Der Service Builder bietet alle Funktionalitäten für die Gateway-Serviceentwicklung »aus einer Hand« und wird durch zusätzliche Support-Werkzeuge ergänzt.

ABAP Development Tools for SAP NetWeaver

In bestimmten Fällen können Sie auch ausgewählte Schritte in anderen Werkzeugen ausführen und die Ergebnisse dann in den Service Builder importieren (so z. B. die Nutzung eines OData-Modell-Editors für die Erstel-

lung eines OData-Modells). Neben dem Service Builder spielen im ABAP-Programmiermodell für SAP Fiori die Eclipse-basierten *ABAP Development Tools for SAP NetWeaver* (ADT, auch bekannt als ABAP in Eclipse) eine wichtige Rolle, da sie für die Entwicklung von CDS Views verwendet werden (siehe Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«).

Das Hauptziel dieses Kapitels ist es, Ihnen einen Überblick über den Prozess der Erstellung von OData-Services zu geben (siehe Abschnitt 5.2), den wir dann in Kapitel 6, »Serviceentwicklung«, Kapitel 7, »Servicegenerierung«, und Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«, ausführlicher diskutieren. Dazu bieten wir Ihnen in Abschnitt 5.1, »Serviceerstellung – Möglichkeiten«, eine kurze Übersicht über die einzelnen Schritte, die in den beiden Arten der Serviceerstellung (Serviceentwicklung und Servicegenerierung) ausgeführt werden.

In Abschnitt 5.3, »SAP Gateway – Entwicklungswerkzeuge«, stellen wir Ihnen dann das wichtigste Werkzeug für die Serviceerstellung vor: den SAP Gateway Service Builder. Danach werden weitere Werkzeuge beschrieben, die für die Erstellung und Verwaltung von Gateway-Services eingesetzt werden.

In Abschnitt 5.4, »Serviceerstellung – Schritt für Schritt«, werden wir uns dann detailliert mit den drei Hauptschritten der Serviceerstellung befassen: der Datenmodelldefinition, Serviceimplementierung und Serviceverwaltung.

Außerdem betrachten wir folgende Themen der Serviceerstellung: das Redefinieren von bestehenden Geschäftsobjekten und die Wiederverwendung und Erweiterung vorhandener Gateway-Services in Erweiterungsszenarien. Das Entwicklungsmodell, das der Entwicklung von Services in SAP Gateway zugrunde liegt, ist der sogenannte OData-Channel. Diesen stellen wir Ihnen in Abschnitt 5.5 vor.

5.1 Serviceerstellung – Möglichkeiten

Es gibt zwei Möglichkeiten, um OData-Services mit SAP Gateway zu erstellen:

Serviceentwicklung

- **Serviceentwicklung**
  Der klassische Ansatz ist die Programmierung von Gateway-Services in ABAP. Die Programmierung in ABAP ist einerseits äußerst flexibel und ermöglicht die Erstellung von sehr effizienten Services. Sie erfordert andererseits jedoch ein gewisses Maß an technischem Know-how, ähnlich dem, das für die Entwicklung von Klassen und RFC-Funktionsbausteinen (RFC – Remote Function Call) benötigt wird.

Servicegenerierung

- **Servicegenerierung**
  Ein zweiter Weg ist die Generierung von Gateway-Services, bei der es vier Arten gibt:
  - *Mapping einer Datenquelle*: Hier können Sie einen Service durch das Mapping der CRUD-Q-Vorgänge einer Entitätsmenge auf die Methoden einer Datenquelle generieren. Dieses Vorgehen wird für folgende Datenquellen unterstützt:
    - RFC-Funktionsbausteine oder BAPIs mit dem *RFC-/BOR-Generator* (BOR – Business Object Repository)
    - Suchhilfen (nur `READ`- und `QUERY`-Methode)
    - *Core Data Services* (CDS) Views (nur `READ`- und `QUERY`-Methode)
  - *Redefinieren (Redefinition)*: Das Redefinieren erlaubt die Definition eines Gateway-Service auf Basis einer existierenden Datenquelle oder eines existierenden Gateway-Service.
    (Beachten Sie: Diese Option wird im Service Builder als *Überdefinieren* bezeichnet.)
  - *Referenzierte Datenquellen*: Exponieren eines oder mehrerer CDS Views, Assoziationen und Aktionen als OData-Service
  - *Anlegen von CDS Views mit Eclipse*: Durch eine einfache Annotation (`OData.publish: true`) in einem CDS View können OData-Services auch ohne Verwendung des Service Builders direkt aus Eclipse heraus angelegt werden.

Vor- und Nachteile

Die Servicegenerierung führt üblicherweise schneller zu Ergebnissen und ist weniger aufwendig als die Serviceentwicklung. Die Generierung von Services ist jedoch nicht so flexibel wie die Serviceentwicklung und daher vor allem für die Erstellung von einfachen Services geeignet. Die Optimierung von generierten Services ist nur eingeschränkt möglich, wenn keine ABAP-Programmierung erfolgen soll.

In den meisten Fällen wird man sich für die Serviceentwicklung entscheiden, da die Vorteile den erhöhten Entwicklungsaufwand rechtfertigen. Der Entwicklung von OData-Services mit SAP Gateway haben wir mit Kapitel 6, »Serviceentwicklung«, einen ausführlichen Praxisteil gewidmet.

Die Generierung von Gateway-Services ist jedoch dann interessant, wenn Sie über geeignete Datenquellen verfügen, wie z. B. Suchhilfen, GenIL-Objekte (GenIL – Generic Interaction Layer), Service-Provider-Objekte, BW Queries wie eine Easy Query oder aber geeignete RFC-Funktionsbausteine oder BAPIs. Auch die Generierung von Gateway-Services auf Basis der zuvor genannten SAP-Geschäftsobjekte stellen wir in einem eigenen Praxisteil (Kapitel 7, »Servicegenerierung«) vor.

CDS Views

Mit SAP S/4HANA können aber nun OData-Services auf Basis von CDS Views generiert werden, die auch die *Draft-Infrastruktur* unterstützen. Dies wird zukünftig die bevorzugte Art sein, mit der OData-Services erstellt werden (❶ in Abbildung 5.1).

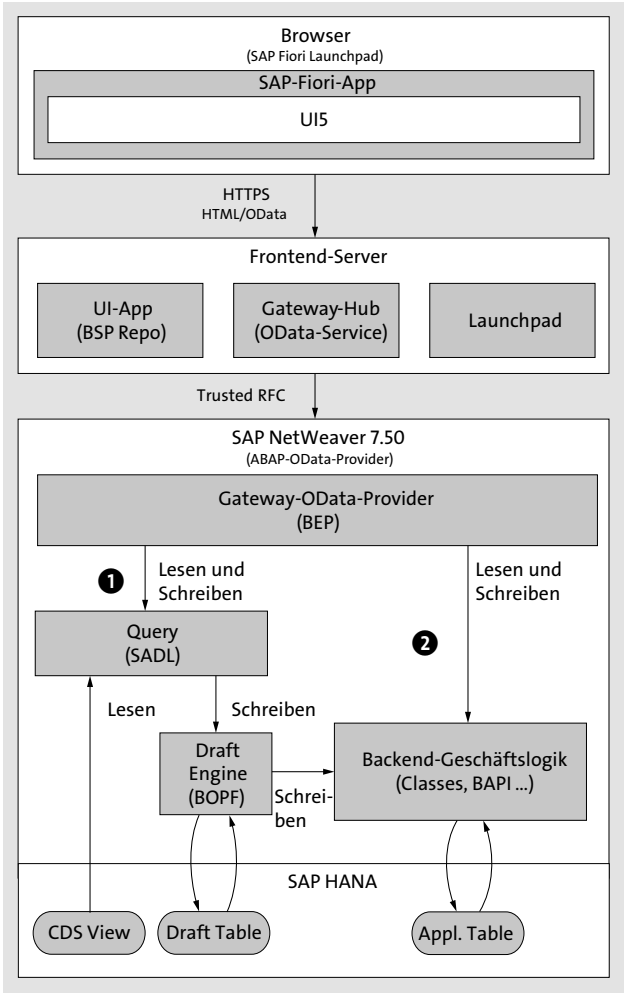


Abbildung 5.1 SAP-Gateway-Service-Erstellung für SAP Fiori – das neue Programmiermodell in SAP S/4HANA

Draft

Eine transaktionale Anwendung mit Draft-Funktionen ermöglicht es dem Benutzer, ein neues Dokument zu erstellen oder ein bestehendes Dokument zu bearbeiten, ohne die Änderungen sofort übernehmen zu müssen und ohne dabei Gefahr zu laufen, die im Entwurfsmodus erstellten oder veränderten Daten zu verlieren. Die im Entwurfsmodus erstellten oder überarbeiteten Daten sind dabei für den Zugriff durch andere Benutzer

gesperrt und werden durch das Framework in den Entwurfstabellen zwischengespeichert, die mit der Annotation `writeDraftPersistence` ausgezeichnet sind.

SAP Fiori Elements

Da für OData-Services, die auf diese Weise erstellt wurden, mit SAP Fiori Elements (früher als Smart Templates bekannt) Benutzeroberflächen generiert werden können, wird es in der Mehrzahl der Anwendungsszenarien in SAP S/4HANA nicht mehr erforderlich sein, SAPUI5-Code zu entwickeln. Es wird genügen, hier geeignete CDS Views und BOPF-Objekte (BOPF – Business Object Processing Framework) zu entwickeln.

Auch wenn OData-Services auf Basis von CDS Views generiert wurden, ist es möglich, zusätzliche Anwendungslogik in der Daten-Provider-Erweiterungsklasse mit ABAP-Code zu implementieren (diese Option werden wir detailliert in Kapitel 7, »Servicegenerierung«, und Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«, erläutern).

In Systemen, die auf SAP NetWeaver 7.50 beruhen, wird es aber auch weiterhin möglich sein, OData-Services zu entwickeln oder über das Mapping einer Datenquelle zu implementieren ❷. Auf diese Weise werden Kunden in der Lage sein, bereits vorhandene Geschäftsprozesslogik in Form von ABAP-Klassen und RFC-Funktionsbausteinen für die Erstellung von OData-Services zu nutzen, auch wenn Sie SAP Business Suite EHP 8 oder aber SAP S/4HANA (on premise) verwenden.

Unabhängig davon, ob Sie sich für die Entwicklung oder die Generierung von Services entscheiden, folgen Sie in beiden Fällen dem Serviceerstellungsprozess in SAP Gateway.

5.2 Prozess der Serviceerstellung

Dieser Abschnitt gibt Ihnen einen Überblick über die Vorgehensweise bei der Serviceerstellung. Die Serviceerstellung wird dabei, bewusst vereinfacht, als eine Abfolge sequenzieller Schritte dargestellt, das heißt als *Wasserfallmodell*.

In der Realität erfolgt die Ausführung der einzelnen Schritte nicht sequenziell, sondern vielmehr iterativ. Zum besseren Verständnis beginnen wir mit der Darstellung der vereinfachten Vorgehensweise, bevor wir den detaillierten Prozess eingehend vorstellen.

Der Prozess der Serviceerstellung teilt sich in drei Hauptphasen auf: in die *Definition des Datenmodells*, die *Serviceimplementierung* und die *Serviceverwaltung*. Je nachdem, ob Sie sich für die Generierung oder die Entwick-

lung eines Service entschieden haben, müssen Sie innerhalb dieser Phasen unterschiedliche Schritte ausführen, es gibt jedoch auch Schritte, die sowohl bei der Entwicklung als auch bei der Generierung von Services ausgeführt werden müssen.

Bevor Sie mit der Erstellung eines Service beginnen können, muss dieser zunächst definiert werden. Hierbei werden Art und Umfang des Service festgelegt. Dies geschieht idealerweise in Abstimmung mit dem Entwickler der Client-Applikation, sodass Sie wissen, welche Daten in der Anwendung benötigt werden und wie Sie diese aus dem SAP-Backend-System erhalten. Sind diese Voraussetzungen geklärt, können Sie mit den drei Phasen des Serviceerstellungsprozesses beginnen.

Phase der Datenmodelldefinition

In der Phase der Datenmodelldefinition definieren Sie das Datenmodell Ihres Service. Dabei werden die notwendigen Artefakte wie Entitätstypen, Entitätsmengen, Assoziationen und weitere Komponenten angelegt, die Ihr Service nutzen wird (die zuvor genannten Komponenten wurden in Kapitel 2, »Einführung in OData«, näher erläutert). Nach der Definition des Datenmodells müssen zunächst die Repository-Objekte generiert und im SAP-Backend-System registriert werden. Danach können Sie mit der nächsten Phase der Serviceerstellung fortfahren, der Serviceimplementierung.

Phase der Serviceimplementierung

In der Phase der Serviceimplementierung werden die Methoden implementiert, die von dem Service unterstützt werden sollen. Abhängig davon, ob dies durch Serviceentwicklung oder Servicegenerierung geschieht, folgen Sie unterschiedlichen Implementierungspfaden:

- Bei der Serviceentwicklung werden die Methoden, die vom Service unterstützt werden, mithilfe von ABAP-Programmierung implementiert.
- Bei der Servicegenerierung haben Sie die folgenden vier Möglichkeiten, einen Service zu generieren:
  - Wenn Sie das Mapping einer Datenquelle verwenden, erfolgt die Implementierung durch das Mapping der Vorgänge einer Entitätsmenge auf die Methoden eines RFC-Funktionsbausteins, eines BOR-Objekts, einer Suchhilfe oder eines CDS Views.
  - Im Fall der Redefinition gibt es keinen separaten Schritt für die Implementierung des Service. Sie müssen lediglich den Datenmodellierungsschritt ausführen. Die Implementierung des Service wird dann auf Basis des Customizings generiert, das im Datenmodellierungsschritt durchgeführt wurde.
  - Auch bei der Verwendung von referenzierten Datenquellen gibt es keinen Serviceimplementierungsschritt, da hier ein oder mehrere Entitätsmengen oder Assoziationen eines CDS Views in das Datenmodell übernommen werden.

  - Wenn Sie einen CDS View mit Eclipse anlegen und durch die Anmerkung (`OData.publish: true`) als OData-Service im Backend-System registrieren, gibt es ebenfalls keinen Serviceimplementierungsschritt. Hier erfolgt die Nutzung einer generischen Serviceimplementierung auf Basis der CDS-View-Definition.

Phase der Serviceverwaltung

In der dritten Phase des Prozesses der Serviceerstellung, der Serviceverwaltung, wird der Service im Gateway-Server aktiviert, sodass er im Servicekatalog auffindbar ist. Dies bedeutet, dass der Service von einem Client konsumiert werden kann.

Die drei Phasen – Datenmodelldefinition, Serviceimplementierung und Serviceverwaltung – sind in Abbildung 5.2 dargestellt.

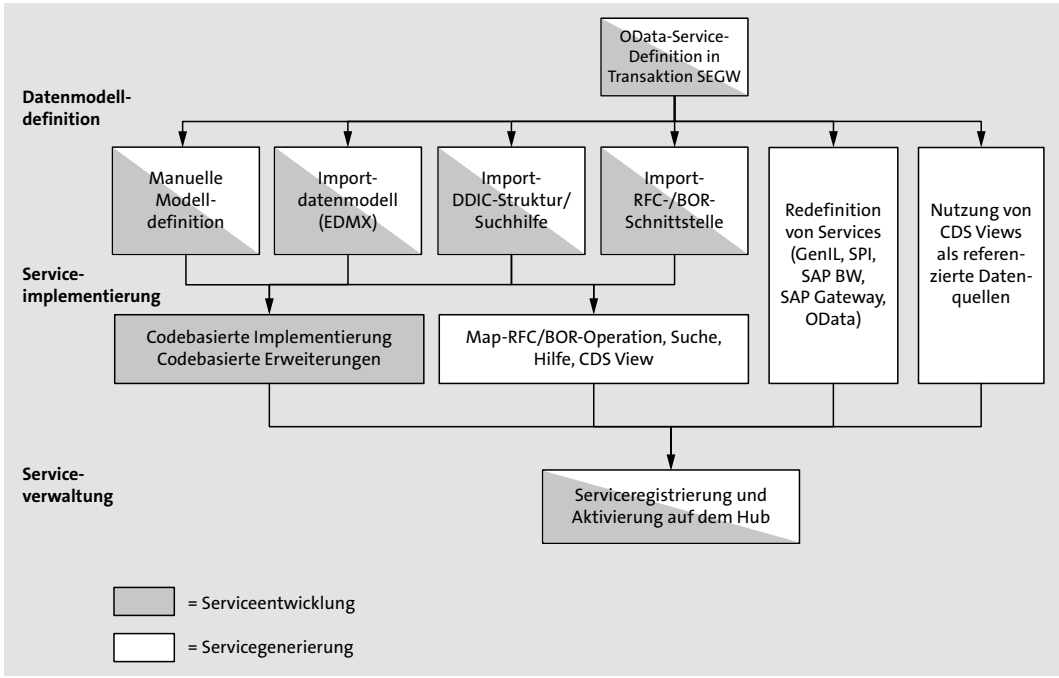


Abbildung 5.2 Prozess der Serviceerstellung

Dabei sind die Schritte, die in der Serviceentwicklung und in der Servicegenerierung durchgeführt werden, mit unterschiedlichen Farben gekennzeichnet. Die Schritte, die sowohl bei der Serviceentwicklung als auch bei der Servicegenerierung durchgeführt werden, sind mit zwei Farben gekennzeichnet. Obwohl wir die beiden Methoden der Serviceerstellung (Servicegenerierung und Serviceentwicklung) getrennt dargestellt haben, ist es möglich, beide Ansätze zu kombinieren.

So können Sie z. B. einen OData-Service erstellen, bei dem eine Entitätsmenge mithilfe des RFC-/BOR-Generators durch Mapping (Servicegenerierung) implementiert wird, während eine zweite Entitätsmenge durch ABAP-Programmierung (Serviceentwicklung) implementiert wird. Es ist auch möglich, einen Service auf Basis eines CDS Views zu generieren, der zunächst nur einen lesenden Zugriff auf Daten ermöglicht. Diesen Service können Sie dann durch ABAP-Code so erweitern, dass er auch einen ändernden Zugriff auf Geschäftsdaten erlaubt.

Inkrementeller Serviceerstellungsprozess

Wie bereits erwähnt, haben wir den Prozess der Serviceerstellung der Übersichtlichkeit halber zunächst klar strukturiert und als Abfolge sequenzieller Schritte dargestellt. Dieses Wasserfallmodell eignet sich gut, um die drei verschiedenen Phasen der Serviceerstellung zu erläutern. In realen Entwicklungsprojekten wird man die Reihenfolge, in der die einzelnen Schritte ausgeführt werden, so anpassen, wie es für die Serviceerstellung am besten ist.

Lediglich die Tätigkeiten der Serviceverwaltungsphase werden nur einmal ausgeführt. Sobald ein Service im Backend registriert und im Gateway-Hub aktiviert (das heißt veröffentlicht) ist, müssen diese Einstellungen in der Regel nicht mehr angepasst werden.

In allen anderen Phasen der Serviceerstellung sollten Sie hingegen einen inkrementellen Ansatz verfolgen: Legen Sie einen Service an – oder Teile eines Service –, führen Sie diesen aus, und testen Sie ihn. Danach können Sie den Service so lange verändern, bis er schließlich allen Anforderungen genügt. Innerhalb des Prozesses der Erstellung eines OData-Service können das Datenmodell und auch die Serviceimplementierung mehrfach geändert werden.

Serviceverwaltung

Das Publizieren eines Service ist eine einmalige Angelegenheit, wenn nicht weitreichende Änderungen am Service vorgenommen werden. Das Registrieren eines Service für weitere SAP-Backend-Systeme ist ein Beispiel für solch eine weitreichende Änderung.

Änderungen in der Serviceimplementierung oder dem Datenmodell eines Service, der bereits publiziert wurde, können in Entwicklungssystemen hingegen ohne zusätzliche Aktivitäten umgesetzt werden.

Beachten Sie: Da die Datenmodelle in Produktivsystemen aus Performancegründen gecacht werden, müssen bei Änderungen in der Datenmodelldefinition die Caches im Backend und im Gateway-Server aktualisiert werden.

Reihenfolge der Phasen ändern

In realen Entwicklungsprojekten werden die Phasen der Serviceimplementierung und Serviceverwaltung oft in umgekehrter Reihenfolge ausführt. Generiert man nach der Datenmodelldefinition zunächst das Service-Builder-Projekt und führt anschließend direkt den Schritt der Serviceverwaltung aus, um den Service zu aktivieren, kann man zunächst die Metadaten des Service (Servicedokument und Servicemetadatendokument) kontrollieren, obwohl der Service noch keine Funktionalität bietet. Anschließend kann man den Service-Stub inkrementell implementieren.

Abbildung 5.3 stellt diesen inkrementellen Prozess der Serviceerstellung dar. Der Prozess basiert auf Abbildung 5.2; zusätzlich haben wir noch inkrementelle Schritte hinzugefügt. Die inkrementellen Schritte sind dabei durch die durchgezogenen Pfeile gekennzeichnet, die die Übergänge zwischen den Phasen der Datenmodellierung und der Serviceimplementierung darstellen. Die Phasen sind durch horizontale Kästen dargestellt. Die gepunktete Linie zeigt dagegen die einmalige Aktivierung eines Service in der Phase der Serviceverwaltung.

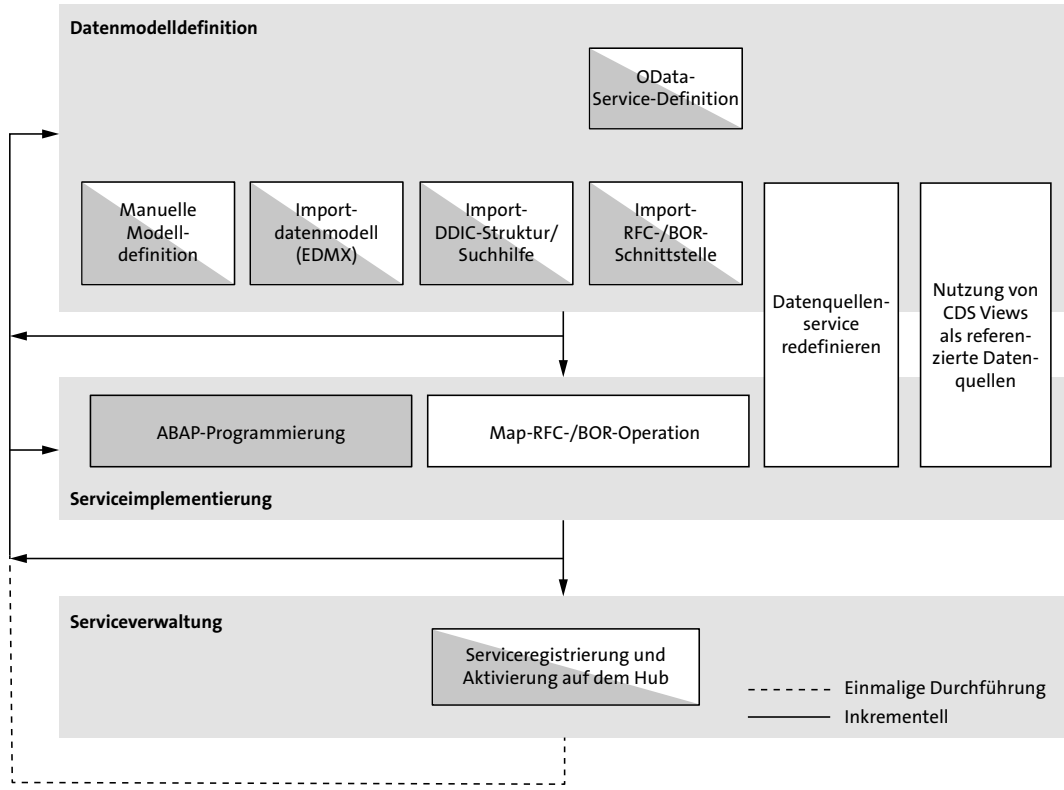


Abbildung 5.3 Inkrementeller Prozess der Serviceerstellung

5.3 SAP Gateway – Entwicklungswerkzeuge

SAP Gateway stellt eine Reihe von Werkzeugen zur Verfügung, um die unterschiedlichen Anforderungen aus den Bereichen Entwicklung, Test und Betrieb von OData-Services zu adressieren.

An dieser Stelle werden wir die Werkzeuge für den Betrieb von SAP Gateway nicht näher erläutern und uns stattdessen auf die Werkzeuge fokussieren, die im Rahmen der Serviceentwicklung zum Einsatz kommen. Wir werden daher zunächst im folgenden Abschnitt den SAP Gateway Service Builder näher betrachten. Anschließend werden wir in Abschnitt 5.3.2 weitere Werkzeuge vorstellen, die Sie beim Prozess der Erstellung von OData-Services unterstützen und die in optimaler Weise auf das Zusammenspiel mit dem Service Builder abgestimmt sind. In Abschnitt 5.3.3 werden wir Ihnen dann die Eclipse-basierten ABAP Development Tools for SAP NetWeaver vorstellen, mit denen CDS Views erstellt werden können.

5.3.1 SAP Gateway Service Builder

Modellierung und die Entwicklung von OData-Services

Der Service Builder bietet Ihnen als Entwickler alle Funktionen, die Sie für die Modellierung und die Entwicklung von OData-Services in SAP Gateway benötigen. Dies umfasst sowohl die Programmierung als auch die Generierung von OData-Services. Darüber hinaus bietet der Service Builder einen direkten Zugriff auf zusätzliche Funktionen, die für Sie interessant sind, wie z. B. die Serviceverwaltung und die Servicevalidierung. Der Service Builder unterstützt den gesamten Entwicklungszyklus (*Development Lifecycle*) eines OData-Service in SAP Gateway und kann über die Transaktion SEGW gestartet werden (siehe Abbildung 5.4).

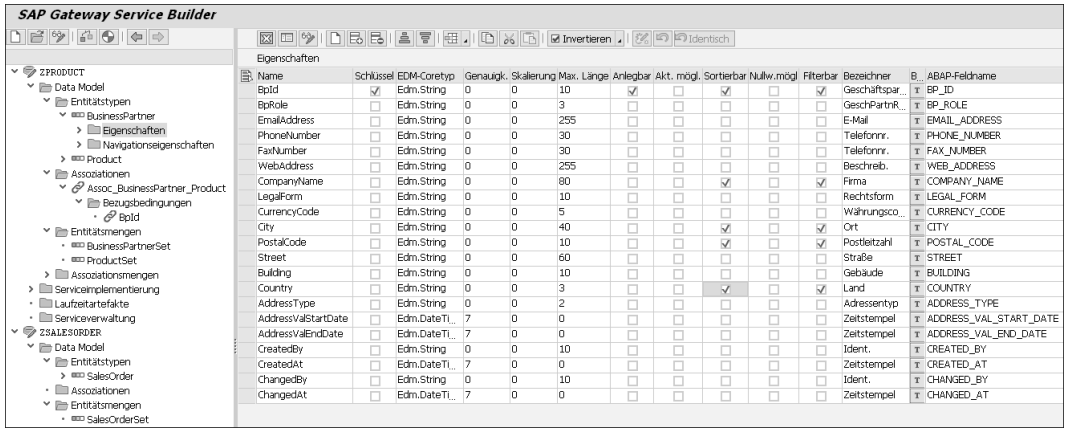


Abbildung 5.4 SAP Gateway Service Builder

Zielgruppe

Dabei eignet sich der Service Builder sowohl für erfahrene Entwickler als auch für Anfänger in der ABAP-Programmierung und für Nicht-Entwickler. Ihnen als erfahrenem Entwickler bietet der Service Builder die Möglichkeit der Entwicklung von Services durch ABAP-Programmierung und damit ein Maximum an Flexibilität bei der Serviceimplementierung. Für die Datenmodellierung können Sie den *OData Model Editor* des Service Builders nutzen, der den ABAP-Code der Modell-Provider-Klasse generiert.

Als Anfänger in der ABAP-Programmierung und Nicht-Entwickler hingegen werden Sie die Möglichkeiten der Servicegenerierung zu schätzen wissen, die die Erstellung von OData-Services erlauben, ohne dass Sie hierfür auch nur eine Zeile ABAP-Code schreiben müssen.

Der Service Builder ermöglicht die Entwicklung und Definition von OData-Services an zentraler Stelle. Dies umfasst die Laufzeitartefakte (Modell-Provider-Klasse – MPC, Daten-Provider-Klasse – DPC, Modell- und Servicedefinition), OData-Artefakte (Entitätsmenge, Entitätstypen und Attribute) sowie die verwendeten Datenquellen und Modelle.

Projektbasierte Entwicklung

Die Modellierungsumgebung des Service Builders unterstützt die projektbasierte Entwicklung. Alle entwicklungsrelevanten Objekte werden in Projekten zusammengefasst. Das Anlegen eines Projekts im Service Builder ist der Startpunkt für jede Art von Serviceerstellung mit dem Service Builder. Damit haben Sie als Serviceentwickler die Möglichkeit, alle entwicklungsrelevanten Artefakte in Projekten an einer zentralen Stelle zu bündeln. Der Service Builder erlaubt es Ihnen darüber hinaus, wie in Abbildung 5.5 gezeigt, mehrere Projekte zu öffnen und zu bearbeiten. In dem Beispiel wurden die Projekte `ZPRODUCT` und `ZSALESORDER` geöffnet.

In welchem System wird der Service Builder verwendet?

Der Service Builder wird in dem System verwendet, in dem die BEP-Komponente (*Business Enablement Provisioning*) installiert ist. Dies ist üblicherweise das SAP-Backend-System (siehe hierzu Kapitel 4, »Deployment-Optionen, Installation und Konfiguration«, in dem wir die verschiedenen Deployment-Optionen dargestellt haben).

Die BEP-Komponente wird bis SAP NetWeaver 7.31 als Add-on `IW_BEP` ausgeliefert. Mit SAP NetWeaver 7.40 SP02 ist die BEP-Komponente Teil der SAP-Basis und wird über die Softwarekomponente `SAP_GWFND` ausgeliefert. Daher ist es ohne besonderen Aufwand möglich, in auf SAP NetWeaver 7.40 basierenden oder neueren Systemen mit dem Service Builder OData-Services zu entwickeln. Dies gilt für alle SAP-Business-Suite-Systeme mit

EHP 7 oder höher wie auch für alle SAP-S/4HANA-Systeme, die on premise betrieben werden.

Da der Service Builder als Teil der BEP-Komponente in der Regel (aber nicht zwingend) auf dem SAP-Backend-System installiert ist, werden sowohl das Servicemodell (die Modell-Provider-Klasse, MPC) als auch die Serviceimplementierung (Daten-Provider-Klasse, DPC) auf diesem System definiert. Um zu verstehen, welche ABAP-Repository-Objekte – z. B. Dictionary-Elemente (DDIC), Strukturen und Datenelemente – referenziert werden können, wenn RFCs, BAPIs oder Klassenmethoden aufgerufen werden, ist dies wichtig zu wissen.

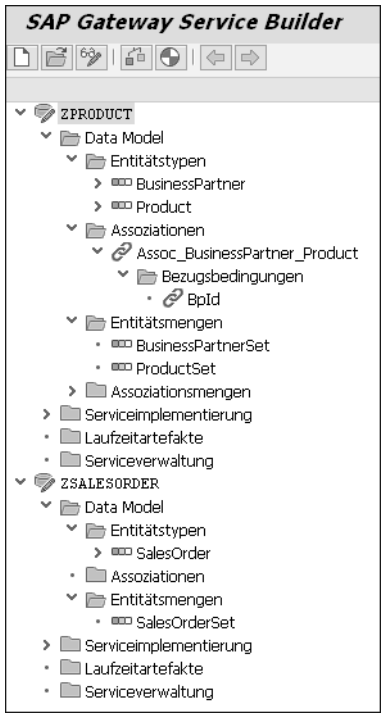


Abbildung 5.5 Projektbasierte Entwicklung

Bestmögliche Unterstützung

Mit dem Service Builder soll Ihnen die bestmögliche Unterstützung bei der Erstellung von OData-Services geboten werden, sowohl programmatisch als auch über die Generierung von OData-Services. Es gibt jedoch technische Restriktionen, was mit dem Service Builder modelliert, entwickelt oder generiert werden kann. Bestimmte OData-Features müssen gegebenenfalls manuell implementiert werden, und bestimmte Methoden sind gegebenenfalls nicht in einem redefinierten Geschäftsobjekt verfügbar.

Das Ergebnis der Serviceentwicklung bzw. Servicegenerierung sind in jedem Fall ABAP-Klassen, die auf dem Programmiermodell des OData-Channels basieren (siehe Abschnitt 5.5). Sie haben jederzeit die Möglichkeit, den Quellcode der ABAP-Klassen zu analysieren, um die Funktionsweise eines Service besser zu verstehen. Dann können Sie den (generierten) Code gegebenenfalls anpassen, indem Sie die entsprechenden Methoden der Modell- bzw. der Daten-Provider-Erweiterungsklasse redefinieren, die ihre Funktionalität von den Basisklassen erben und bei der Neugenerierung eines Service im Gegensatz zur Basisklasse nicht überschrieben werden.

5.3.2 Weitere Werkzeuge zur Unterstützung des Serviceerzeugungsprozesses

Wie bereits erwähnt, ist das zentrale Werkzeug für die Erstellung von Services der SAP Gateway Service Builder. Darüber hinaus bietet SAP Gateway eine Reihe von Werkzeugen an, die den Entwicklungsprozess von Gateway-Services unterstützen. Mithilfe dieser Werkzeuge ist es z. B. möglich, Services frühzeitig zu testen oder die Kommunikation eines Service zu tracen. Aus diesem Grund geben wir Ihnen in diesem Abschnitt einen kurzen Überblick über einige dieser Funktionalitäten. Eine ausführliche Darstellung der Entwicklungs- und Administrationswerkzeuge von SAP Gateway finden Sie in Kapitel 15, »Lifecycle Management: Qualitätssicherung, Service-Deployment und Operations«.

SAP Gateway Client

Testen und Troubleshooting

Der *SAP Gateway Client* kann sowohl für das Testen als auch für das Troubleshooting verwendet werden und ist ein im Gateway-Server eingebauter REST-Client. Der SAP Gateway Client kann aus dem SAP GUI heraus über die Transaktion /IWFND/GW_CLIENT gestartet werden. Nachdem man einen Service auf dem Gateway-Server aktiviert hat, kann dieser dort umgehend mit dem SAP Gateway Client getestet werden, wie in Abbildung 5.6 gezeigt.

Hierzu wählen Sie zuerst die HTTP-Methode, wie z. B. **GET**, **PUT**, **POST**, **PATCH** oder **DELETE**, aus ❶. Dann geben Sie den URI Ihres Requests im Eingabefeld **Request-URI** ein ❷. Falls erforderlich, können Sie auch bestimmte HTTP-Header setzen. Der Body des HTTP-Requests kann entweder manuell eingegeben werden, oder Sie laden ihn aus einer Datei hoch ❸. Darüber hinaus ist es möglich, die Funktionalität **Als Anf. verwenden** zu nutzen. Damit kann der Inhalt einer `CREATE`- oder `UPDATE`-Anforderung auf Basis der HTTP-Response einer Leseanforderung ❹ erzeugt werden. Der HTTP-Request kann nun ausgeführt werden, wenn Sie **Ausführen** auswählen ❺.

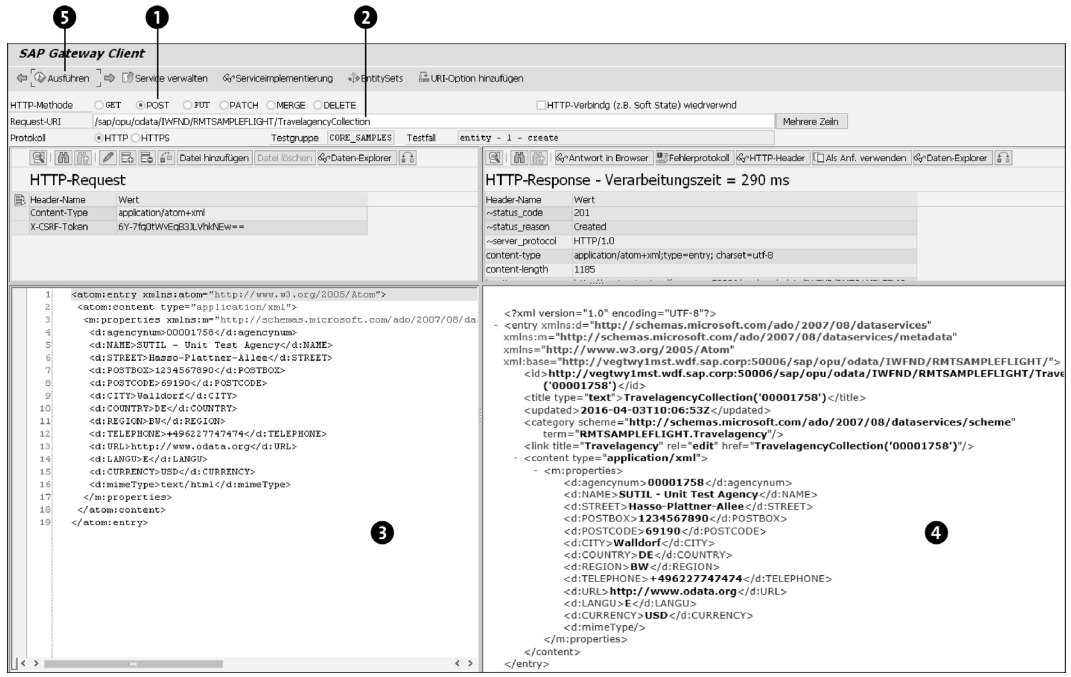


Abbildung 5.6 SAP Gateway Client – Anforderung erstellen

Testfälle speichern

Eine besonders nützliche Funktionalität des SAP Gateway Clients ist, dass Sie Testfälle in einer Datenbank speichern können. Der Testfall, der in Abbildung 5.6 gezeigt wird, ist einer von mehr als 250 Testfällen, die als Testgruppe mit dem Namen `CORE_SAMPLES` ausgeliefert werden. Diese Testgruppe enthält Testfälle für die Standardtest-Gateway-Services `TEA_TEST_APPLICATION` und `RMTSAMPLEFLIGHT`. Beachten Sie, dass die Testfälle der Testgruppe `CORE_SAMPLES` in Ihrem System wie in Abbildung 5.7 angelegt werden müssen. Wählen Sie dazu im Menü des SAP Gateway Clients **SAP Gateway Client ❶ • Core Samples anlegen ❷**.

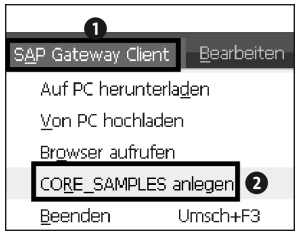


Abbildung 5.7 Anlegen der Testfälle der Testgruppe CORE_SAMPLES

Wenn Sie einen HTTP-Request als Testfall gespeichert haben, können Sie anschließend den HTTP-Rückgabecode festlegen, der bei der Ausführung

des HTTP-Requests vom Service zurückgeliefert werden soll. Ein HTTP-Request kann dabei mehrere gültige Rückgabecodes zurückliefern (z. B. 200, 401, 402 und 403). Aus diesem Grund können auch im SAP Gateway Client mehrere gültige Rückgabewerte sowie Intervalle spezifiziert werden (z. B. 201, 401–403).

Darüber hinaus ist es möglich, die Payload-Validierung zu nutzen. Dabei kann der Payload eines HTTP-Requests mit dem erwarteten Payload eines Service verglichen werden und nicht nur mit dem HTTP-Rückgabecode.

Ein oder mehrere Testfälle können dann mit dem SAP Gateway Client ausgeführt werden. Die Ergebnisse werden in einer Tabelle **SAP Gateway Client – Mehrfachtest** mithilfe von Ampeln angezeigt, wobei sowohl der erwartete als auch der tatsächliche HTTP-Rückgabecode in der Tabelle aufgeführt werden.

Fehlerprotokoll

Das *Fehlerprotokoll* (*Error Log*) ist ein weiteres Werkzeug, das für Sie als Entwickler bei der Fehlerbehandlung von großem Nutzen ist. Das Fehlerprotokoll wird im Gateway-Server über die Transaktion /IWFND/ERROR_LOG aufgerufen. Im SAP-Backend-System gibt es auch ein Backend-Fehlerprotokoll mit einer sehr ähnlichen Benutzeroberfläche. Mithilfe des Backend-Fehlerprotokolls können Fehler analysiert werden, die im Business-Suite-Backend-System auftreten. Das Backend-Fehlerprotokoll wird über die Transaktion /IWBEP/ERROR_LOG im SAP-Backend-System aufgerufen.

Das Fehlerprotokoll ist gut mit dem SAP Gateway Client integriert. Dadurch ist es beispielsweise möglich, einen HTTP-Request, der von einem OData-Konsumenten versendet wurde und der zu einem Verarbeitungsfehler führte, erneut auszuführen. Wählen Sie dazu wie in Abbildung 5.8 **Wiedergabe • SAP Gateway Client**.

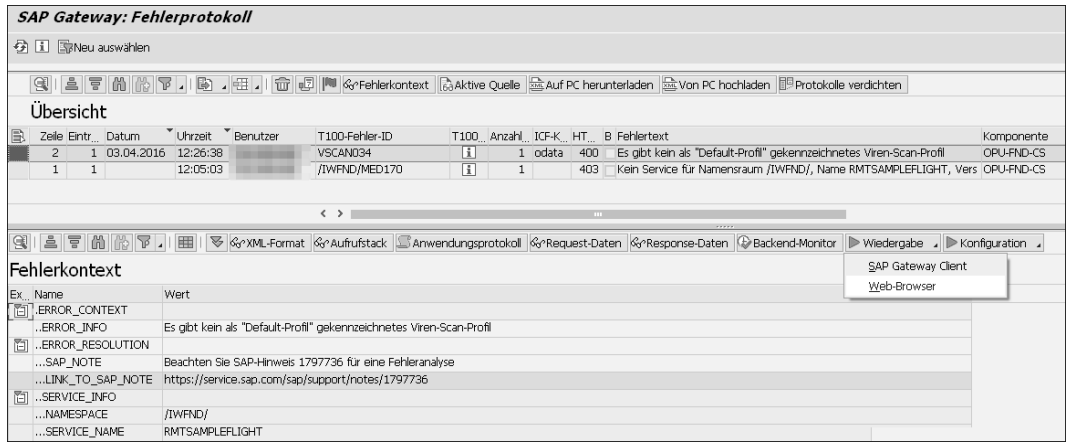


Abbildung 5.8 Transaktion /IW_FND/ERROR_LOG

Logging und Tracing

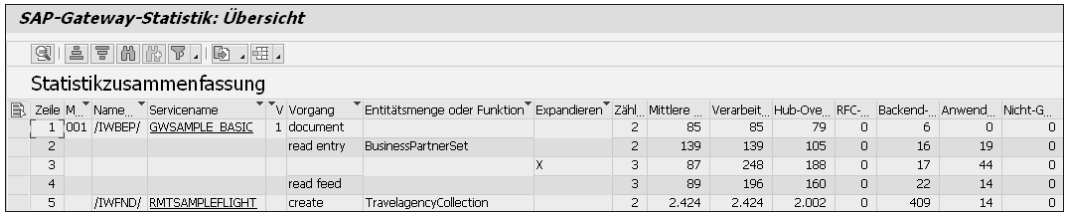
Eine weitere Möglichkeit der Fehlerbehandlung bietet die Analyse von Protokolleinträgen, die für das *SysLog* und das *Anwendungsprotokoll* von SAP Gateway von einem OData-Service erzeugt werden. Das SysLog kann über die Transaktion SM21 aufgerufen werden, während der Gateway-Anwendungsprotokoll-Viewer über Transaktion /IWFND/APPS_LOG aufgerufen werden kann.

SAP-Gateway-Statistikdaten

Bei der Entwicklung eines OData-Service oder einer Client-Anwendung sind Sie als Entwickler natürlich an Performancedaten zu diesem Service interessiert. Sie können die Statistikdaten für einen HTTP-Request im SAP Gateway Client anzeigen, wenn Sie an den Request-URI den URL-Parameter `?sap-statistics=true` anhängen oder den HTTP-Request-Header `sap-statistics=true` setzen. Das SAP Gateway Framework stellt dem Client die Statistikdaten im HTTP-Header `sap-statistics` bereit. Die Antwortzeiten werden durch das SAP Gateway Framework darüber hinaus für jeden eingehenden HTTP-Request im SAP-Gateway-Server gespeichert.

Performance kontrollieren

Auf Basis dieser Daten können Sie über die Transaktion /IWFND/STATS (SAP-Gateway-Statistik, siehe Abbildung 5.9) auch auf die Statistikdaten einzelner Serviceaufrufe zugreifen. Die Statistikdaten werden in regelmäßigen Abständen verdichtet, sodass die Performance jedes Service einfach ausgewertet werden kann. In Produktivsystemen ist diese Transaktion für den Administrator von großem Nutzen, wenn dieser die Performance von OData-Services kontrollieren möchte.



Zeile	M...	Name...	Servicename	V	Vorgang	Entitätsmenge oder Funktion	Expandieren	Zähl...	Mittlere ...	Verarbeit...	Hub-Ove...	RFC-...	Backend-...	Anwend...	Nicht-G...
1	001	/IWBEP/	GWSAMPLE_BASIC	1	document			2	85	85	79	0	6	0	0
2					read entry	BusinessPartnerSet		2	139	139	105	0	16	19	0
3							X	3	87	248	188	0	17	44	0
4					read feed			3	89	196	160	0	22	14	0
5		/IWFND/	RMTSAMPLEFLIGHT		create	TravelagencyCollection		2	2.424	2.424	2.002	0	409	14	0

Abbildung 5.9 Transaktion /IWFND/STATS

Über die Transaktion /IWFND/TRACES können Sie nicht nur die Performance eines einzelnen Serviceaufrufes im SAP-Gateway-Server und Backend-System aufzeichnen, sondern dies auch mit der Payload eines Serviceaufrufes tun (siehe Abbildung 5.10).

Mithilfe des Payload-Trace ist es möglich, sowohl die Payload zu überwachen, die vom Client zum Server gesendet wird, als auch den Inhalt der Response, die vom Server zurück an den Client geschickt wird. Die aufgezeichneten Daten können dann dazu verwendet werden, erneut einen Request an den Server zu senden.

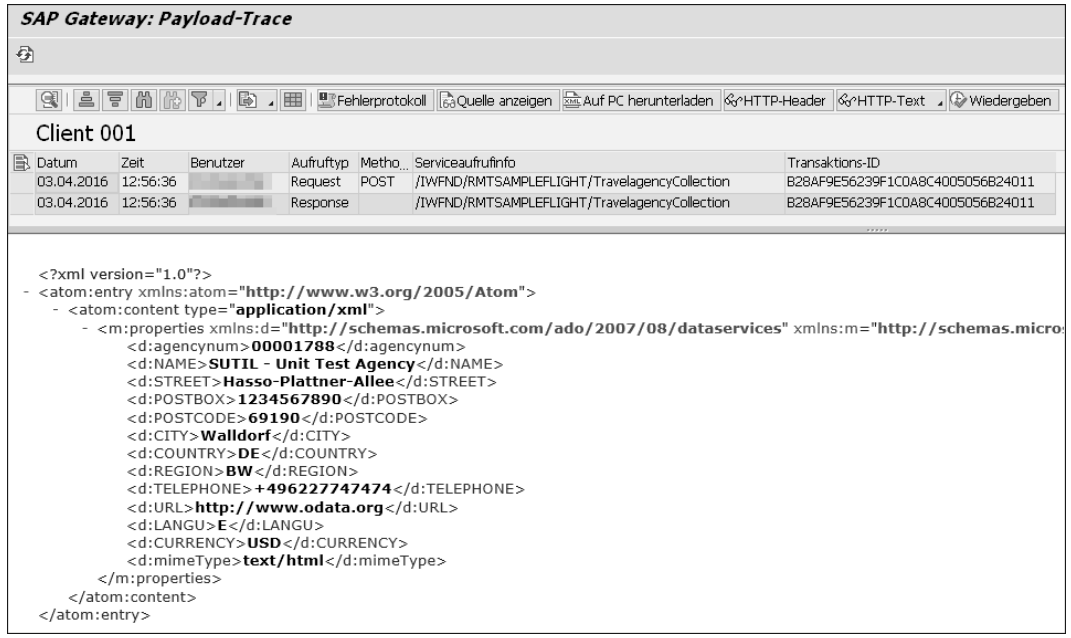


Abbildung 5.10 Transaktion /IWFND/TRACES – Payload-Trace

Testfälle erstellen

Mit den im Payload-Trace aufgezeichneten Daten ist es darüber hinaus sehr einfach, Testfälle im SAP Gateway Client zu erstellen. Dies ist vor allem im Support-Fall von großem Nutzen. Um den Payload- und den Performance-Trace zu verwenden, müssen diese in der Transaktion /IWFND/TRACES aktiviert werden. Wir betrachten den Payload- und den Performance-Trace detailliert in Anhang A, »Weiterführende Konzepte«.

Katalogservice

Jedes Gateway-System bietet einen *Katalogservice* (*Service Catalog*), über den man eine Liste aller Services eines Gateway-Servers erhalten kann (siehe Abbildung 5.11). Der Katalog ist selbst ein OData-Service, und die Liste der verfügbaren Services kann man über die folgende URL abfragen:

http://<server>:<port>/sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/CatalogCollection

OpenSearch

Der Katalogservice unterstützt das OpenSearch-Protokoll. Entwickler oder Entwicklungsumgebungen wie die SAP Web IDE können daher Services über eine Freitextsuche anhand der Servicebeschreibung finden. Durch

den Aufruf der folgenden URL können sie nach Services suchen, bei denen das Wort »Produkt« in der Servicebeschreibung enthalten ist:

http://<server>:<port>/sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/Service Collection?search=Produkt

```
<?xml version="1.0" encoding="UTF-8"?>
<app:service xml:lang="de" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xml:base="http://vegtwy1mst.wdf.sap.corp:50006/sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/"
xmlns:sap="http://www.sap.com/Protocols/SAPData" xmlns:atom="http://www.w3.org/2005/Atom"
xmlns:app="http://www.w3.org/2007/app">
  - <app:workspace>
        <atom:title type="text">Data</atom:title>
    - <app:collection href="Vocabularies" sap:content-version="2" sap:deletable="false" sap:updatable="false" sap:addressable="false">
        <atom:title type="text">Vocabularies</atom:title>
        <sap:member-title>Vocabulary</sap:member-title>
      </app:collection>
    - <app:collection href="ServiceCollection" sap:content-version="2" sap:searchable="true" sap:deletable="false" sap:updatable="false"
      sap:creatable="false">
        <atom:title type="text">ServiceCollection</atom:title>
        <sap:member-title>Service</sap:member-title>
        <atom:link title="searchServiceCollection" type="application/opensearchdescription+xml" rel="search"
            href="ServiceCollection/OpenSearchDescription.xml"/>
      </app:collection>
    - <app:collection href="TagCollection" sap:content-version="2" sap:updatable="false" sap:creatable="false">
        <atom:title type="text">TagCollection</atom:title>
        <sap:member-title>Tag</sap:member-title>
      </app:collection>
    - <app:collection href="EntitySetCollection" sap:content-version="2" sap:deletable="false" sap:updatable="false" sap:creatable="false">
        <atom:title type="text">EntitySetCollection</atom:title>
        <sap:member-title>EntitySet</sap:member-title>
      </app:collection>
    - <app:collection href="CatalogCollection" sap:content-version="2">
        <atom:title type="text">CatalogCollection</atom:title>
        <sap:member-title>Catalog</sap:member-title>
      </app:collection>
    - <app:collection href="Annotations" sap:content-version="2" sap:deletable="false" sap:updatable="false">
        <atom:title type="text">Annotations</atom:title>
        <sap:member-title>Annotation</sap:member-title>
      </app:collection>
    </app:workspace>
    <atom:link rel="self" href="http://[illegible]/sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/"/>
    <atom:link rel="latest-version" href="http://[illegible]/sap/opu/odata/iwfnd/CATALOGSERVICE;v=2/"/>
</app:service>
```

Abbildung 5.11 Servicekatalog – Servicedokument

5.3.3 ABAP Development Tools for SAP NetWeaver und Core Data Services Views

Ein Core Data Services View (CDS View) ist, wie der Name vermuten lässt, ein View, der definiert wird, um Geschäftsdaten für die Anwendungsebene durch eine Projektion aufzubereiten. Dies ist erforderlich, da die Geschäftsdaten in der Regel über mehrere Datenbanktabellen verteilt sind.

CDS stellt eine Spezifikation für eine SQL-basierte Datenbanksprache (*Data Definition Language*, DDL) bereit. Mit *SAP HANA CDS* und *ABAP CDS* gibt es zwei Dialekte dieser Datenbanksprache. Während mit SAP HANA CDS Views speziell für SAP HANA erstellt werden können, müssen ABAP CDS Views mehrere unterschiedliche Datenbanken unterstützen. Diese unterschiedlichen Anforderungen sind vergleichbar mit den Anforderungen an die ABAP-Open-SQL-Syntax. Auch diese unterstützt nur diejenigen SQL-

Artefakte, die von allen Datenbanken unterstützt werden, die für SAP NetWeaver ABAP freigegeben sind.

> **ABAP und HANA CDS Views**
>
> Einen ausführlichen Vergleich zwischen ABAP CDS Views und SAP HANA CDS Views finden Sie in folgendem Blog in der SAP Community: *http://s-prs.de/v671711*

Werfen wir nun einen Blick auf den Code eines einfachen CDS Views in Listing 5.1, wie Sie ihn auch in der SAP-Onlinehilfe finden: *http://s-prs.de/v671712*

```
@AbapCatalog.sqlViewName: 'CUSTOMER_VW'
DEFINE VIEW cust_book_view_entity AS SELECT FROM scustom
     JOIN sbook
     ON scustom.id = sbook.customid
     {
           scustom.id,
           scustom.name,
           sbook.bookid
     }
```

Listing 5.1 Beispiel für einen ABAP CDS View

Der CDS View `cust_book_view_entity` erzeugt einen Join auf Basis der beiden Datenbanktabellen `sbook` und `scustom`, die beide Teil des bekannten SFLIGHT-Datenmodells sind. Über den oben gezeigten CDS View kann man nun auf die in Listing 5.2 gezeigte ABAP-SQL-Anweisung zugreifen.

```
SELECT id name bookid
FROM cust_book_view_entity
INTO TABLE @DATA(result_data)
WHERE ... .
```

Listing 5.2 Nutzung von CDS Views in ABAP-Code

ABAP Development Tools for SAP NetWeaver

ABAP CDS Views können mithilfe der Eclipse-basierten *ABAP Development Tools for SAP NetWeaver* (ADT) und dem ABAP-CDS-Schlüsselwort `DEFINE VIEW` erstellt werden. Wie in Abbildung 5.12 zu sehen, werden hierdurch zwei Objekte im ABAP Dictionary (hier abgekürzt als DDIC) erstellt, ein SQL View und eine CDS-Entität.

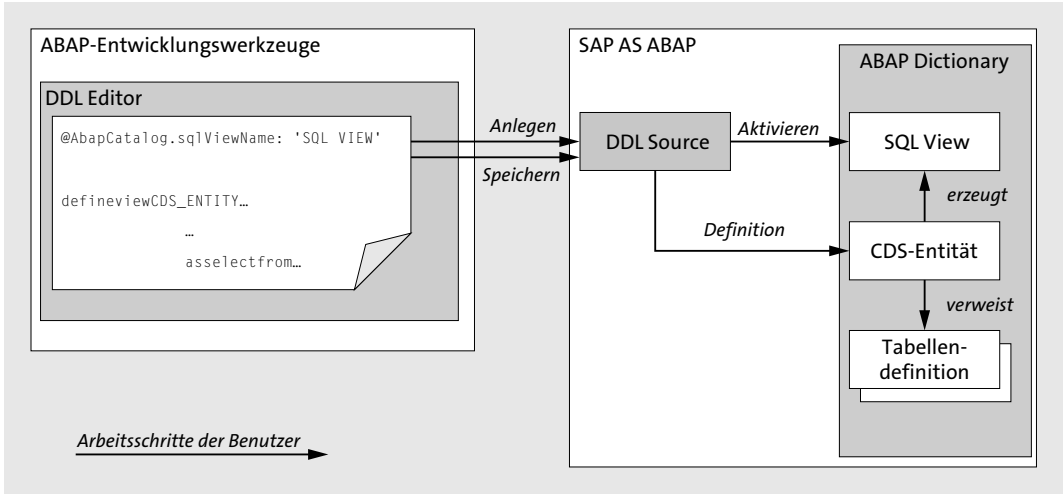


Abbildung 5.12 ABAP-CDS-View-Entwicklung – Architektur

Name des SQL Views und der CDS-Entität

Der SQL View und die CDS-Entität werden im selben Namensraum im DDIC angelegt. Daher müssen die beiden Namen unterschiedlich sein. In dem in Listing 5.1 gezeigten CDS View hat daher der SQL View den Namen `CUSTOMER_VW`, wohingegen die CDS-Entität den Namen `cust_book_view_entity` trägt.

5.4 Serviceerstellung – Schritt für Schritt

Zu Beginn dieses Kapitels haben wir in Abschnitt 5.2 den Prozess für die Erstellung von Gateway-Services vorgestellt. Dieser Prozess besteht aus den drei Phasen: Datenmodellierung, Serviceimplementierung und Serviceverwaltung.

Abhängig davon, ob Sie sich für die Serviceentwicklung oder die Servicegenerierung entscheiden, können Sie unterschiedliche Vorgehensweisen wählen. Wir betrachten nun diese verschiedenen Vorgehensweisen und die darin enthaltenen einzelnen Schritte eingehend aus technischer Sicht.

5.4.1 Datenmodellierung im Service Builder

Die erste Phase bei der Erstellung eines Service in SAP Gateway ist die Datenmodellierung. In dieser Phase soll mithilfe des Service Builders das

OData-Datenmodell des Service modelliert werden, das alle Artefakte des OData-Service beschreibt, wie z. B. Entitätstypen, komplexe Typen, Attribute und Assoziationen.

Bei der Entwicklung eines Gateway-Service (Serviceentwicklung) oder bei der Generierung eines Gateway-Service mithilfe des Mappings einer Datenquelle (eine der verschiedenen Möglichkeiten der Servicegenerierung) startet man mit dem Anlegen eines Datenmodells.

Redefinition

Wenn Sie die zweite Methode der Servicegenerierung wählen, also die Redefinition eines vorhandenen Service, wird kein neues Datenmodell definiert, sondern das vorhandene Datenmodell des Service- oder Geschäftsobjekts wird redefiniert. Die Servicegenerierung mittels Redefinition wird in Abschnitt 5.4.5 beschrieben. Beachten Sie: Im Service Builder wird die Redefinition als »Überdefinieren« bezeichnet.

Fünf Möglichkeiten für die Definition

Der Service Builder bietet Ihnen verschiedene Möglichkeiten, ein Datenmodell zu definieren. Jede dieser Möglichkeiten adressiert dabei einen bestimmten Anwendungsfall:

- Die erste Möglichkeit zur Anlage eines Datenmodells ist, die verschiedenen Artefakte eines OData-Modells manuell im Service Builder anzulegen. Diese Vorgehensweise wird in der Dokumentation als *deklarative Modelldefinition* bezeichnet. Entitätstypen, Assoziationen und Assoziationsmengen werden hierbei manuell mit dem Service Builder angelegt.
- Die zweite Möglichkeit ist, ein komplettes Datenmodell im EDMX-Format zu importieren, das entweder mit dem OData Model Editor der SAP Web IDE oder mithilfe des Entity Data Modelers von Microsoft Visual Studio angelegt wurde. Darüber hinaus kann das Servicemetadatendokument eines bestehenden OData-Service in den Service Builder importiert werden.
- Bei der dritten und vierten Möglichkeit wird ein Entitätstyp auf Basis einer Datenstruktur angelegt, die schon im SAP-Backend-System existiert. Bei dieser für Sie als ABAP-Entwickler sehr bequemen Variante werden Entitätstypen entweder auf Basis von DDIC-Strukturen und Tabellen oder aber auf Basis von RFC-/BOR-Schnittstellen generiert.
- Des Weiteren gibt es die Möglichkeit, die F4-Hilfen zu nutzen, die im DDIC definiert wurden. Die F4-Hilfen können dabei, wie auch RFC-/BOR-Schnittstellen, als Datenquelle genutzt werden und bieten eine Service-

implementierung durch reines Mapping an, das heißt Servicegenerierung.

Im Folgenden werden wir alle fünf genannten Optionen detaillierter beschreiben.

Deklaratives Anlegen eines Datenmodells

Entitätstypen

Ein Datenmodell kann manuell mit dem Service Builder angelegt werden. Diese Methode kann verwendet werden, um Entitätstypen anzulegen, die auf manuell angelegten Attributen basieren. Die so angelegten Attribute können auf existierenden DDIC-Typen basieren.

Um einen OData-Service von Grund auf in einem WYSIWYG-Stil (*what you see is what you get*) zu modellieren, sind jedoch OData-Modellierungswerkzeuge, wie beispielsweise der OData Model Editor der SAP Web IDE und Microsoft Visual Studio, besser geeignet. In diesem Fall ist es jedoch erforderlich, das so angelegte Modell in den Service Builder zu importieren, wie im folgenden Absatz beschrieben wird. Ein Nachteil dieses Ansatzes besteht darin, dass die auf diese Weise angelegten Entitätstypen nicht an vorhandene DDIC-Strukturen oder RFC/BOR-Schnittstellen gebunden werden.

Datenmodell aus Datei

Mit dieser Option können Sie ein komplettes Datenmodell importieren, das entweder in einer EDMX-Datei oder aber in einem Metadatendokument eines existierenden OData-Service gespeichert ist. Dies beinhaltet die Definition sämtlicher OData-Artefakte, wie z. B. Entitätstypen, Entitätsmengen, Assoziationen und andere Komponenten.

Wenn Sie ein Datenmodell in ein bereits existierendes Projekt im Service Builder importieren wollen, so bietet der Service Builder die Möglichkeit des Reimports einer Datenmodelldatei. Dabei erscheint ein Dialog, der Ihnen anzeigt, welche Artefakte dem Datenmodell hinzugefügt werden und welche gelöscht werden würden.

Import eines Datenmodells via DDIC-Import

DDIC-Typunterstützung

Um schnell Entitätstypen und komplexe Typen in einem Datenmodell anzulegen, können die in einem SAP-Backend-System bereits vorhandenen Datenstrukturen genutzt werden. Dazu können die folgenden DDIC-Typen in den Service Builder importiert werden:

- Views
- Datenbanktabellen
- Strukturen

Bearbeitung der Attribute eines Entitätstyps – Beautification

Wird ein Entitätstyp auf Basis eines DDIC-Typs angelegt, kann schon während des Imports ein leicht verständlicher Name gewählt werden. Auch ist es möglich, automatisch eine Default-Entitätsmenge anlegen zu lassen, wobei dem Namen des Entitätstyps als Vorschlag die Endung »Set« angehängt wird.

Der Name des Entitätstyps wird vom Service Builder vorgeschlagen und dabei vom Namen des DDIC-Typs abgeleitet, indem die Unterstriche entfernt werden. Die durch Unterstriche getrennten Namensteile werden zu einem Namen in CamelCase-Notation zusammengesetzt. Wird beispielsweise eine Struktur mit dem Namen `BAPI_EPM_PRODUCT_HEADER` importiert, wird der Service Builder als Namen des Entitätstyps `BapiEpmProductHeader` vorschlagen. Dieser kann durch einen sprechenderen Namen ersetzt werden.

Diese Namenskonvention wird auch für die Attribute der so generierten Entitätstypen verwendet, sodass statt des originalen Feldnamens `SUPPLIER_NAME` der Name der Eigenschaft im generierten Entitätstyp `SupplierName` lauten wird.

Insbesondere die Namen der Entitätsmengen und ihrer Attribute sollten einfach zu verstehen sein. Dies ist wichtig, da es die Namen der Entitätsmengen und ihrer Attribute sind, die ein externer Konsument sieht.

Während des Imports einer DDIC-Struktur oder auch direkt danach können Sie einen als *Beautification* bezeichneten Prozess starten. Hierbei kann die Anzahl der Attribute eines Entitätstyps reduziert werden, indem diese einfach aus der Definition des Entitätstyps entfernt werden. Darüber hinaus ist es auch möglich, die generierten Namen von Attributen zu ändern, um sie so verständlicher zu gestalten.

Die Reduzierung der Anzahl von Attributen auf das notwendige Maß und das Umbenennen von Attributen, die nach außen sichtbar sind, sind essenziell, wenn es darum geht, Services zu erstellen, die einfach zu konsumieren sind. Das Publizieren existierender DDIC-Strukturen ohne verständliche Namen ist kontraproduktiv.

Das Thema Beautification wird eingehend in Abschnitt 7.3.1, »SAP BW Easy Query«, dargestellt.

Import eines Datenmodells via RFC/BOR

Funktionsmodul und BAPI-Parameter

Eine weitere Möglichkeit, mithilfe des Service Builders Entitätstypen und Entitätsmengen anzulegen, ist es, die Schnittstellen (Interfaces) von RFC-

Funktionsbausteinen oder BAPIs zu importieren. Auch hier bietet der Service Builder einen Wizard an, der Sie durch den Importprozess führt.

Die Verwendung von Schnittstellen von RFC-Funktionsbausteinen oder BOR-Objekten ist hilfreich, wenn auf Daten aus dem SAP-Backend-System zugegriffen werden soll.

Dabei kann die Implementierung der Schnittstellen durch ABAP-Programmierung (Serviceentwicklung) oder durch reines Mapping (Servicegenerierung) erfolgen. Beim Mapping können mithilfe des RFC-/BOR-Generators die Vorgänge einer Entitätsmenge auf die Methoden einer RFC-/BOR-Schnittstelle gemappt werden.

Import einer Suchhilfe (F4-Hilfe)

Als fünfte Option bietet der Service Builder die Möglichkeit, Entitätstypen auf Basis existierender F4-Suchhilfen anzulegen. Ähnlich wie bei BOR-/RFC-Schnittstellen können hier Entitätstypen und Entitätsmengen auf Basis von Suchhilfen erzeugt werden. Der Assistent erstellt dabei auch das Mapping der Lesevorgänge (`QUERY`, `READ`), sodass kein separater Schritt für die Serviceimplementierung notwendig ist.

5.4.2 Serviceregistrierung im SAP-Backend-System

Nachdem ein Datenmodell angelegt wurde, muss dieses noch registriert werden. Mit der Registrierung eines Service im SAP-Backend-System wird das Ergebnis der Phase der Datenmodellierung persistiert.

Dies bedeutet, dass nun Laufzeitobjekte, die für einen Gateway-Service benötigt werden, mit dem Service Builder generiert werden. Der Service Builder übernimmt dabei für Sie auch die Konfigurationsschritte, die notwendig sind, um den Service im SAP-Backend-System zu registrieren.

Serviceregistrierung und Serviceverwaltung

Wie Sie in Abschnitt 5.1, »Serviceerstellung – Möglichkeiten«, gesehen haben, beinhaltet die Phase der Serviceverwaltung innerhalb der Serviceerstellung die Aktivitäten, um den Service im Gateway-Server zu registrieren und zu aktivieren. Dieser Prozess darf nicht mit der Serviceregistrierung im SAP-Backend-System verwechselt werden, die nach der Datenmodelldefinition im Backend erfolgt.

In diesem Abschnitt werden wir uns auf die Serviceregistrierung im SAP-Backend-System konzentrieren. In Abschnitt 5.4.4 werden wir die Serviceverwaltung vorstellen. Serviceregistrierung und Serviceverwaltung unterscheiden sich dabei wie folgt:

- Die Serviceregistrierung ist eine Aktivität im Rahmen der Serviceerstellung, die im SAP-Backend ausgeführt wird. Als Ergebnis der Serviceregistrierung werden die für die Implementierung nötigen Laufzeitartefakte im Backend-System generiert.
- Die Serviceverwaltung ist eine Aktivität, die im Gateway-Server ausgeführt wird. Hierbei wird der zuvor im Backend registrierte Service aktiviert, sodass er konsumiert werden kann.

Stub-Class-Erzeugung

Auf der Basis des Datenmodells werden vom Service Builder die korrespondierende Modell- (MPC) und die Daten-Provider-Klasse (DPC) sowie deren Erweiterungsklassen generiert.

Die Basisklasse der Modell-Provider-Klasse enthält dabei den ABAP-Code, mit dem das Datenmodell des Service definiert wird. Die Implementierung der Serviceoperationen erfolgt in der Daten-Provider-Klasse. Die Erweiterungsklassen, die vom Service Builder generiert werden, können dazu verwendet werden, die entsprechenden Methoden der generierten Basisklasse zu redefinieren, das heißt mit kundeneigenem Code zu implementieren. Während die Methoden der Basisklassen der Daten-Provider-Klasse und der Modell-Provider-Klasse bei der Neugenerierung des Service-Builder-Projekts überschrieben werden, ist dies bei den Erweiterungsklassen nicht der Fall (weiterführende Informationen zum Thema Modell- und Daten-Provider-Klasse finden Sie in Abschnitt 5.5, »OData-Channel«).

Serviceregistrierung

Damit die Laufzeitartefakte als Service genutzt werden können, müssen einige Konfigurationsschritte ausgeführt werden. Die Ausführung dieser Schritte wird vom Service Builder unterstützt.

Wird ein Projekt im Service Builder zum ersten Mal generiert, müssen Sie die Namen der Modell- und Daten-Provider-Erweiterungslasse sowie deren Basisklassen festlegen (siehe Abbildung 5.13). Darüber hinaus müssen Sie den **Techn. Modell Name**(n) (*Technical Model Name*) und den **Name**(n) **des techn. Service** (*Technical Service Name*) festlegen. Letzterer wird beim Publizieren des Service im Gateway-Service als Servicename verwendet und kann nicht geändert werden.

Modell- und Daten-Provider-Klasse

Modell- und Daten-Provider-Klasse werden daher durch Customizing und nicht programmatisch zu einem Gateway-Service zusammengefügt. Diese Konfigurationsschritte werden, wie bereits erwähnt, durch den Service Builder ausgeführt, wenn ein Projekt zum ersten Mal generiert wird. Der Prozess der Modell- und Servicedefinition ist in Abbildung 5.14 dargestellt.

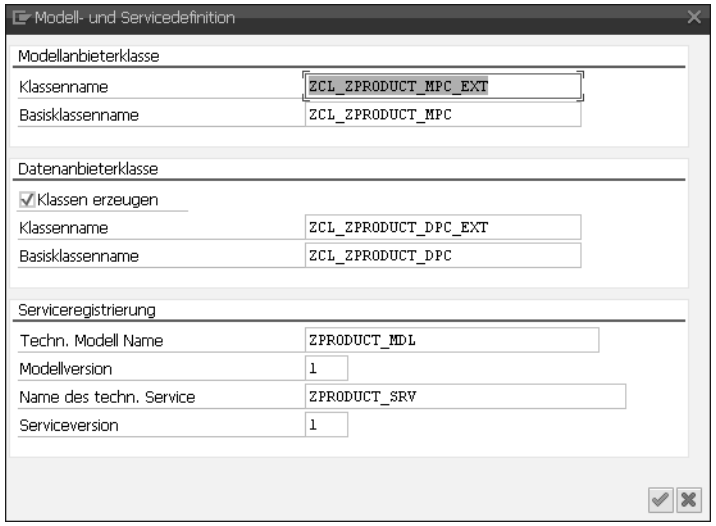


Abbildung 5.13 Dialog der Modell- und Servicedefinition (bei Anmeldung in deutscher Sprache)

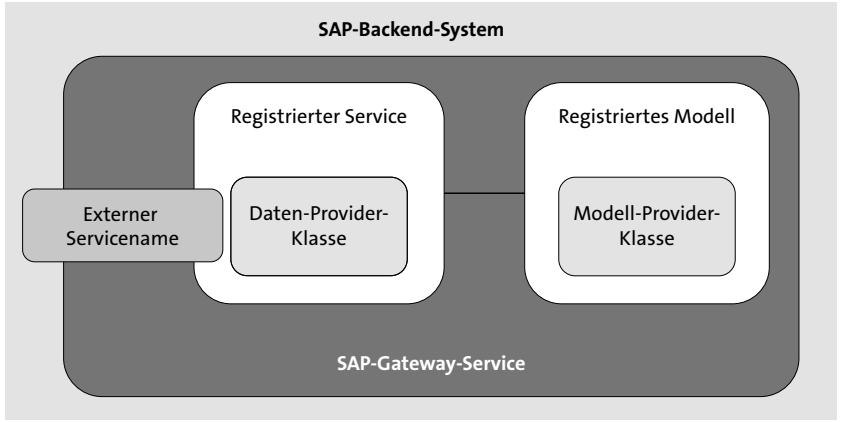


Abbildung 5.14 Service- und Modellregistrierung

Neben der Modell-Provider-Klasse (siehe Abschnitt 5.5.1) und Daten-Provider-Klasse (siehe Abschnitt 5.5.2) werden zwei weitere Repository-Objekte für das Modell und den Service angelegt, wenn der Service im SAP-Backend-System registriert wird.

5.4.3 Serviceimplementierung

Während der Phase der Serviceimplementierung werden die Methoden eines Gateway-Service implementiert. Die Implementierung erfolgt dabei

entweder durch ABAP-Programmierung oder durch das Mappen der Methoden eines RFC-Funktionsbausteins, eines BAPIs oder einer Suchhilfe auf die Attribute eines OData-Modells.

Die Vorgänge, die für eine Entitätsmenge zur Laufzeit ausgeführt werden können, umfassen Vorgänge zum Anlegen, Lesen, Aktualisieren und Löschen von Daten. Für diese werden die folgenden englischen Bezeichnungen verwendet: `CREATE`, `READ`, `UPDATE`, `DELETE` und `QUERY`, die daher auch als CRUD-Q-Methoden bezeichnet werden.

An dieser Stelle ist es wichtig, darauf hinzuweisen, dass die Serviceimplementierungsphase nur für die Serviceentwicklung und nur für eine der möglichen Optionen der Servicegenerierung relevant ist, nämlich für das Mapping einer Datenquelle. Für die Generierung von Services durch Redefinition (im Service Builder als Überdefinieren bezeichnet) und für referenzierte Datenquellen (CDS Views) muss der Schritt der Implementierung von Services nicht ausgeführt werden. Dies liegt daran, dass die Serviceimplementierung auf Basis des Customizings generiert wird, das im Schritt der Modelldefinition ausgeführt wurde. Zu einem späteren Zeitpunkt ist jedoch eine Implementierung der Methoden der Erweiterungsklassen möglich. Dadurch kann beispielsweise eine Unterstützung für das Anlegen, Ändern und Löschen von Daten für einen Service, der auf Basis eines CDS Views als rein lesender Zugriff generiert wurde, implementiert werden.

Servicegenerierung

Die Servicegenerierung durch Redefinieren (Überdefinieren) wird in Abschnitt 5.4.5 und die Servicegenerierung durch referenzierte Datenquellen in Abschnitt 5.4.6 näher beschrieben.

Im Folgenden geben wir Ihnen für die Szenarien einen kurzen Überblick über die Phase der Serviceimplementierung, für die sie relevant ist: die Serviceentwicklung und die Servicegenerierung durch Mapping einer Datenquelle.

Serviceimplementierung durch ABAP-Programmierung

Wie Sie sich erinnern werden, wurden während der Serviceregistrierung am Ende der Datenmodellierungsphase eine Daten-Provider-Klasse und eine Erweiterungsklasse erzeugt. Während der Phase der Serviceimplementierung werden diejenigen Vorgänge (Methoden) implementiert, die vom Gateway-Service unterstützt werden sollen.

Bei der Implementierung der Vorgänge durch ABAP-Programmierung müssen Sie die entsprechenden Methoden der Daten-Provider-Erweiterungsklasse (DPC_EXT) implementieren. Die Namen dieser Methoden werden Sie an die zuvor erwähnten CRUD-Q-Methoden erinnern:

- <ENTITY_SET_NAME>_CREATE_ENTITY
- <ENTITY_SET_NAME>_GET_ENTITY
- <ENTITY_SET_NAME>_UPDATE_ENTITY
- <ENTITY_SET_NAME>_DELETE_ENTITY
- <ENTITY_SET_NAME>_GET_ENTITYSET

CRUD-Q-Methoden

Der Service Builder bietet einen bequemen Zugriff auf diese Methoden. Dazu muss der Knoten **Serviceimplementierung** wie in Abbildung 5.15 aufgeklappt werden.

Von hier aus ist eine einfache Navigation zu dem entsprechenden Eintrag einer Entitätsmenge möglich, wobei alle CRUD-Q-Vorgänge der Entitätsmenge angezeigt werden. Durch die Auswahl von **Zur ABAP Workbench** aus dem Kontextmenü gelangen Sie direkt in den Class Builder (Transaktion SE24), in dem Sie die Methode der Daten-Provider-Erweiterungsklasse implementieren können, die dem Vorgang der Entitätsmenge entspricht. Wenn der Service Builder aus den ABAP Development Tools for SAP NetWeaver heraus gestartet wird, wird statt der SE24 der Eclipse-basierte Editor verwendet.

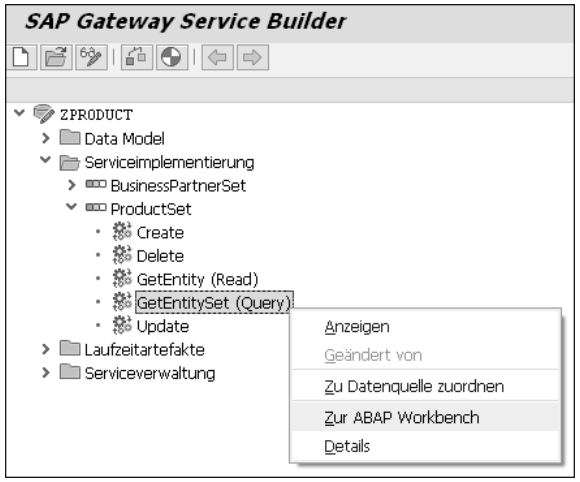


Abbildung 5.15 Serviceimplementierung durch ABAP-Programmierung

Darüber hinaus kann es erforderlich sein, dass noch zusätzliche Methoden in der Daten-Provider-Erweiterungsklasse redefiniert werden müssen, die

nicht entitätsmengenspezifisch sind, wie die zuvor erwähnten CRUD-Q-Methoden (dies ist z. B. der Fall, wenn für den OData-Service die Unterstützung von Deep Insert implementiert werden soll).

Serviceimplementierung durch das Mappen von RFC-/BOR-Schnittstellen

Die Vorgehensweise bei der Implementierung durch das Mapping von RFC-/BOR-Schnittstellen unterscheidet sich grundlegend von der Implementierung durch ABAP-Programmierung.

Der Mapping-Prozess wird dadurch gestartet, dass Sie im Kontextmenü eines CRUD-Q-Vorgangs einer Entitätsmenge im Verzeichnis **Serviceimplementierung** (*Service Implementation*) die Option **Zu Datenquelle zuordnen** (*Map to Datasource*) wählen, wie in Abbildung 5.16 gezeigt.

Das Mapping-Werkzeug des Service Builders ermöglicht es Ihnen dann, Relationen zwischen den Schnittstellenparametern eines RFC-Funktionsbausteins oder BAPIs und den Attributen einer Entitätsmenge festzulegen.

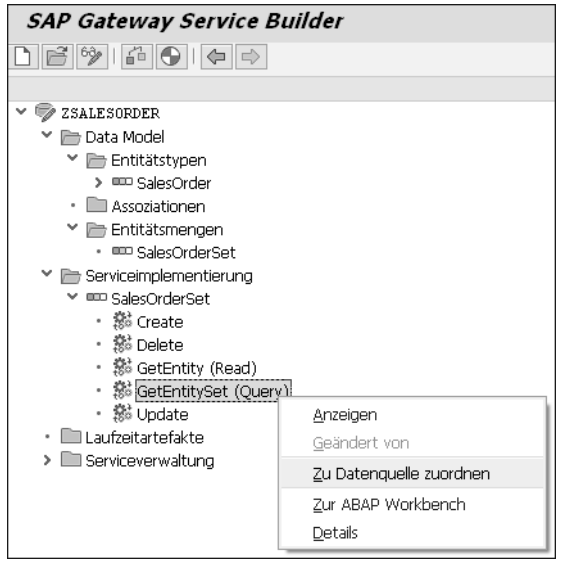


Abbildung 5.16 Mappen des Vorgangs einer Entitätsmenge zu einer Datenquelle

Die Vorgänge CREATE, READ, UPDATE, DELETE und QUERY (CRUD-Q) einer Entitätsmenge müssen dabei separat gemappt werden. Die eigentliche Serviceimplementierung, das heißt der ABAP-Code der Methoden der ABAP-Klasse, die die einzelnen CRUD-Q-Vorgänge implementieren, wird dabei vom Service Builder auf Basis des zuvor beschriebenen Mappings erzeugt.

CRUD-Q

Der Service Builder bietet Ihnen eine besondere Unterstützung, wenn ein Entitätstyp durch den Import der Schnittstelle eines BOR-Objekts oder RFC-

Funktionsbausteins angelegt wurde. In diesem Fall schlägt der Service Builder ein geeignetes Mapping vor, wenn Sie auf **Zuordnung vorschlagen** klicken (siehe ❶ in Abbildung 5.17). Wie in der Abbildung gezeigt, schlägt der Service Builder ein Mapping zwischen der Eigenschaft `SoId` der Entitätsmenge `SalesOrderSet` und des Feldes `SO_ID` des Exportparameters `SOHEADERDATA` des RFC-Funktionsbausteins `BAPI_EPM_SO_GET_LIST` vor ❷.

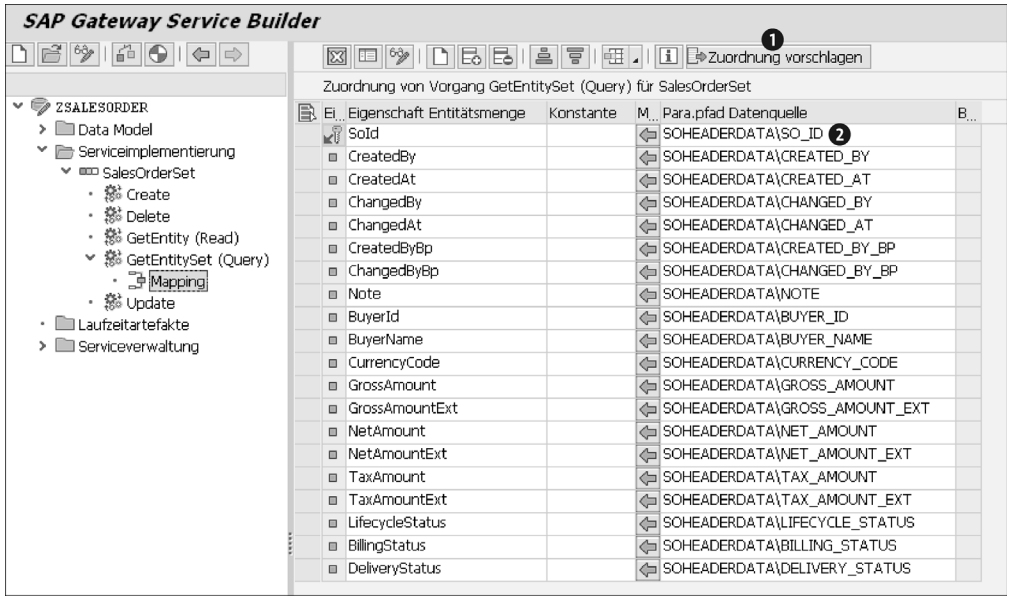


Abbildung 5.17 Mapping-Vorschläge – RFC-/BOR-Generator

Dieser Mapping-Vorschlag konnte automatisch erstellt werden, da der Entitätstyp, auf dem die Entitätsmenge `SalesOrderSet` basiert, durch den Import des Schnittstellenparameters `SOHEADERDATA` erzeugt wurde.

Werden weitere Vorgänge der Entitätsmenge gemappt, prüft der Service Builder die bereits existierenden Mappings und leitet aus diesen Vorschläge für die zusätzlich zu mappenden Vorgänge ab.

Wurde also zunächst der `QUERY`-Vorgang (`GET_ENTITYSET`) einer Entitätsmenge gemappt und möchte man nun noch den `READ`-Vorgang (`GET_ENTITY`) mappen, kann der Service Builder einen Vorschlag für die Attribute erstellen, die bereits in der `GET_ENTITYSET`-Methode gemappt wurden.

Serviceimplementierung durch das Mappen von CDS Views

Die Implementierung eines Service durch das Mappen eines CDS Views unterscheidet sich von der Vorgehensweise beim Mappen einer RFC-/BOR-Schnittstelle.

Um den Assistenten für das Mapping zu starten, müssen Sie den Menüpunkt **Zu Datenquelle zuordnen** aus dem Kontextmenü einer Entitätsmenge im Verzeichnis **Serviceimplementierung** auswählen, anstatt dies wie beim RFC-/BOR-Generator bei den einzelnen Methoden (z. B. `QUERY`) zu tun.

Das Mappen von CDS Views ist die einzige Möglichkeit, CDS Views in SAP NetWeaver 7.40 zu nutzen. Mit dem Release SAP NetWeaver 7.50 hat man die weitaus bequemere Möglichkeit, CDS Views als referenzierte Datenquellen zu nutzen. Beide Möglichkeiten werden wir ausführlich in Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«, behandeln.

Der Mapping-Dialog im Service Builder erlaubt es Ihnen dann, sowohl ein Mapping zwischen den Eigenschaften eines CDS Views und den Eigenschaften einer Entitätsmenge (siehe Abbildung 5.18) als auch ein Mapping zwischen der Navigationseigenschaft einer Entitätsmenge und der Assoziation eines CDS Views zu definieren (siehe Abbildung 5.19).

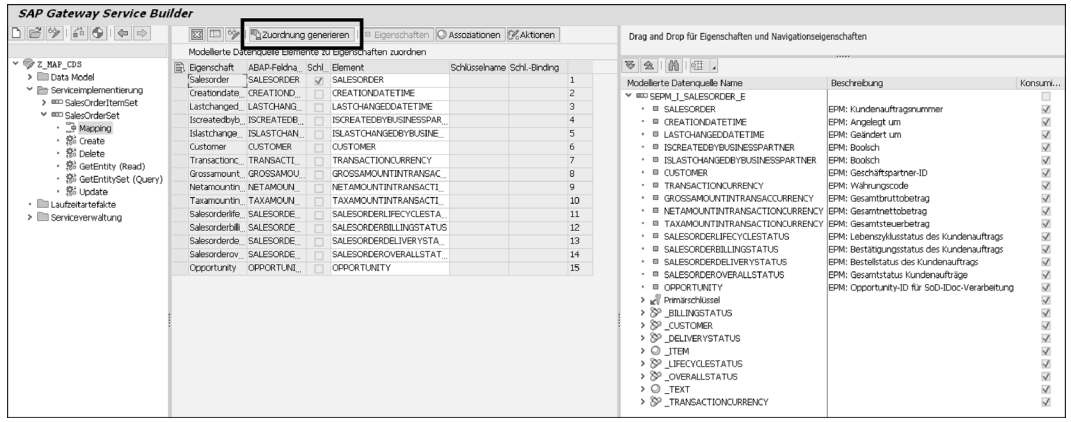


Abbildung 5.18 Mapping eines CDS Views – Eigenschaften

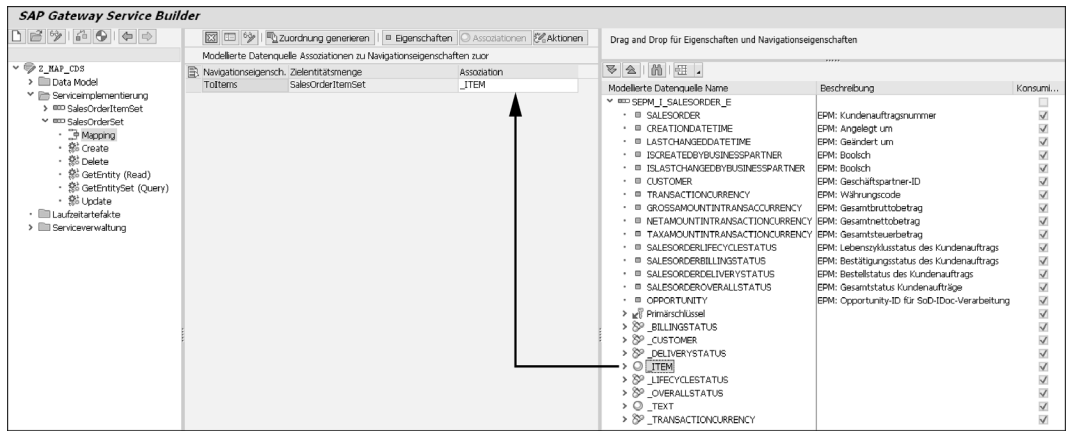


Abbildung 5.19 Mapping eines CDS Views – Assoziation

Hierdurch wird erreicht, dass der lesende Zugriff, das heißt die READ- und die QUERY-Methode einer Entitätsmenge, durch reines Mapping implementiert werden. Die Implementierung der Methoden CREATE, UPDATE und der DELETE (CUD) kann dann immer noch codebasiert erfolgen oder durch das Mappen eines geeigneten RFC-Funktionsbausteins auf die CUD-Methoden.

Serviceimplementierung durch das Mappen von F4-Suchhilfen

Die Vorgehensweise bei der Implementierung durch das Mappen von F4-Suchhilfen ist sogar einfacher als die Mapping-Vorgänge beim RFC-/BOR-Generator und einem CDS View. Der Assistent für den Import einer F4-Suchhilfe bietet nicht nur das Anlegen einer Entitätsmenge an, sondern führt auch direkt das Mapping für die QUERY- und READ-Vorgänge der Entitätsmenge durch (siehe Abbildung 5.20).

Abbildung 5.20 Assistent »Suchhilfe importieren« – automatisches Mapping der READ- und QUERY-Methode

Wie auch im Fall von CDS Views kann die Implementierung von ändernden CUD-Methoden entweder über eine codebasierte Implementierung oder das Mappen von RFC-Funktionsbausteinen erfolgen, die einen Schreibzugriff erlauben.

5.4.4 Serviceverwaltung

Die Phase der Serviceverwaltung besteht hauptsächlich aus den beiden Schritten Serviceregistrierung und Aktivierung im Gateway-Server-System.

Aktivierung und Verwaltung von Services

Die Registrierung und Aktivierung von Services im Gateway-Hub-System führen Sie mit der Transaktion /IWFND/MAINT_SERVICE (Services aktivieren und verwalten) durch. Die Transaktion /IWFND/MAINT_SERVICE wird

auch dafür verwendet, alle aktivierten Services im Gateway-Server zu pflegen. Services müssen gepflegt werden, wenn sie in zusätzlichen Backend-Systemen registriert wurden oder deaktiviert werden sollen, z. B. weil sie nicht mehr benötigt werden. Da der Service Builder der zentrale Einstiegspunkt für die Serviceentwicklung ist, wurde im Service Builder die Funktionalität implementiert, die es Ihnen erlaubt, die Transaktion für die Pflege von Services im Gateway-Hub-System (/IWFND/MAINT_SERVICE) direkt aus dem Service Builder aufzurufen. Der Aufruf der Transaktion funktioniert auch remote, wenn der Service Builder im SAP-Backend aufgerufen wird.

Nun können Sie entweder ein Gateway-System aus der Liste der Gateway-Systeme im Knoten **Serviceverwaltung** auswählen und aus dem Kontextmenü die Option **Registrieren** ❶ oder auf den Button **Bearb.** ❷ klicken (siehe Abbildung 5.21).

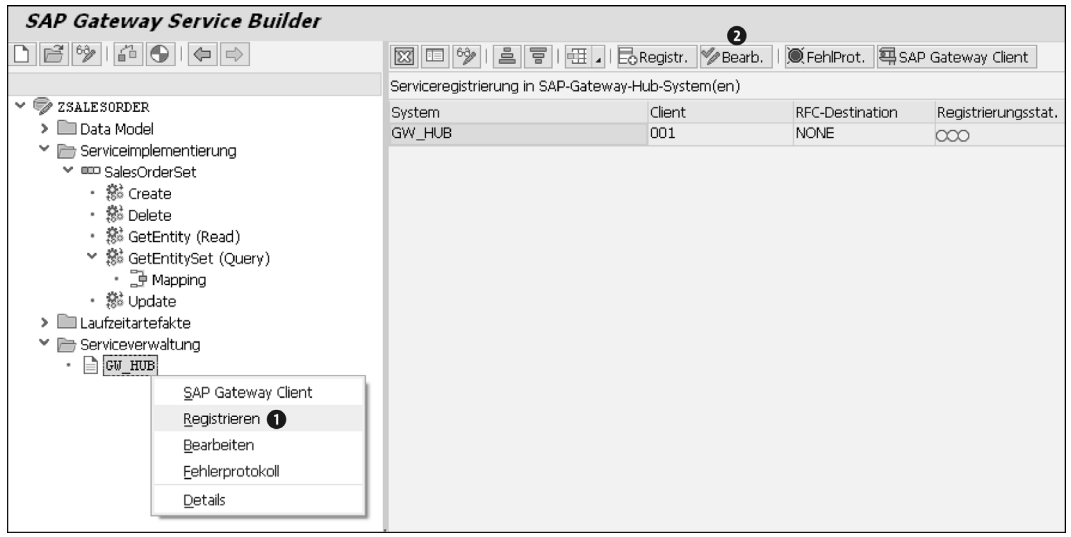


Abbildung 5.21 Aktivieren eines Service im SAP-Gateway-Hub aus dem Service Builder

Generierte Serviceimplementierung

Wie bereits zuvor beschrieben, ist kein separater Schritt für die Serviceimplementierung nötig, wenn ein Service mittels Redefinition oder über eine referenzierte Datenquelle generiert wird. Auch wenn ein CDS View mit der Annotation `OData.publish:true` versehen wird, ist keine Implementierung erforderlich. In den genannten Fällen schließt sich an die Phase der Datenmodellierung direkt die Serviceaktivierung an.

5.4.5 Servicegenerierung mittels Redefinition

Wie in Abschnitt 5.1, »Serviceerstellung – Möglichkeiten«, dargestellt, geht es bei dem Prozess der Redefinition um das Generieren eines Service auf Basis eines existierenden Geschäftsobjekts. Die Generierung erfolgt mithilfe eines Assistenten, bei dem die beiden Phasen der Datenmodelldefinition und der Implementierung zu einer Phase kombiniert werden, die Redefinition genannt wird.

Der so generierte Service muss nun im Gateway-Server aktiviert werden (Phase Serviceverwaltung) und kann dann konsumiert werden. Das Ziel der Redefinition ist es, Services mit geringem Aufwand anzulegen.

Bestehende Geschäftsobjekte

In einem SAP-Backend-System, wie z. B. SAP Customer Relationship Management (SAP CRM), SAP Product Lifecycle Management (SAP PLM) oder SAP Enterprise Asset Management (SAP EAM), gibt es eine Vielzahl verschiedenster Geschäftsobjekte. Obwohl all diese Geschäftsobjekte für unterschiedliche Anwendungsfälle entwickelt wurden, definieren sie doch alle Objekte, Relationen, Aktionen und Suchen, die denen ähneln, die man im OData-Protokoll findet. Daher liegt es nahe, Generatoren zu entwickeln, mit denen man aus den genannten Geschäftsobjekten OData-Services generieren kann.

Erweiterbarkeit

Es ist ebenfalls möglich, einen SAP-Gateway-Service auf Basis eines bereits bestehenden SAP-Gateway-Service mittels Redefinition zu generieren. Dieses Szenario wird verwendet, wenn ein Kunde einen von SAP ausgelieferten OData-Service erweitern möchte, z. B. den OData-Service einer SAP-Fiori-Anwendung. Die Erweiterbarkeit von SAP-Fiori-Anwendungen wird detailliert in Kapitel 11, »Erweiterbarkeit«, beschrieben.

Drittanbieter-OData-Services

Neben der Integration bestehender Geschäftsobjekte ist es auch möglich, OData-Services von Drittherstellern zu integrieren. Dieses Integrationsszenario weist jedoch einige technische Einschränkungen auf.

Redefinitionsassistent

Der Assistent für die Generierung von OData-Services mithilfe der Redefinition (Überdefinition) unterscheidet sich kaum innerhalb der einzelnen Integrationsszenarien. Wählen Sie eine der möglichen Optionen (basierend auf den installierten Add-ons) aus, startet ein Assistent, der Sie durch die folgenden drei Schritte leitet:

1. Auswahl eines Geschäftsobjekts
2. Auswahl der Artefakte der Datenquelle (Datenmodelldefinition)
3. Generierung von Laufzeitartefakten und Serviceregistrierung im Backend (Serviceimplementierung)

Der Assistent startet also mit der Phase der Definition des Datenmodells und führt anschließend automatisch die Schritte durch, die zur Phase der Serviceimplementierung gehören. Nachdem der Service im SAP-Backend-

System auf diese Weise registriert und implementiert wurde, muss er nur noch im SAP-Gateway-Server aktiviert werden.

Die verschiedenen Integrationsszenarien, die in diesem Abschnitt beschrieben werden, basieren teilweise auf den in Tabelle 5.1 aufgeführten Add-ons. Wenn diese Add-ons in einem SAP-Backend-System installiert sind, erscheinen wie in Abbildung 5.22 im Kontextmenü des Service Builders die entsprechenden Menüeinträge.

Name des Add-ons	Integrationsszenario	Remote aufrufbar
IW_GIL	Generic Interaction Layer (GenIL)	
IW_SPI	Service Provider Infrastructure (SPI)	✓
SAP_GWFND und IW_BEP	Analytical Queries	✓
SAP_GWFND oder IW_BEP und IW_FND	OData-Service (External)	✓
SAP_GWFND oder IW_BEP	OData-Service (SAP Gateway)	✓

Tabelle 5.1 Add-ons für die Generierung von Services auf Basis existierender Datenquellen

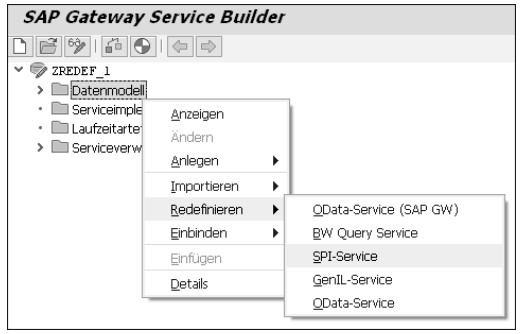


Abbildung 5.22 Kontextmenüoptionen für das Anlegen eines Datenmodells durch Redefinition (Überdefinieren)

Die meisten Integrationsszenarien sind remote aufrufbar. Das bedeutet, dass das zu konsumierende Geschäftsobjekt (z. B. das SPI-Objekt) nicht notwendigerweise auch in dem System vorhanden sein muss, in dem die BEP-Komponente bzw. das spezifische Add-on für das jeweilige Integrationsszenario installiert ist. Aus diesem Grund ist es prinzipiell möglich, die remotefähigen Integrationsszenarien auch im Gateway-Server zu implementieren. In diesem Fall spricht man von einem *Hub-Deployment mit Entwicklung auf dem Hub*.

Add-on IW_SPI und Add-on IW_GIL in SAP S/4HANA

Das Add-on `IW_SPI` wird in SAP S/4HANA nicht mehr unterstützt und kann daher auch deinstalliert werden. Obwohl das Add-on `IW_GIL` noch immer in SAP S/4HANA unterstützt wird, kann es auch deinstalliert werden, wie in den beiden nachfolgend genannten SAP-Hinweisen beschrieben:

- 2319114 – Deinstallation des Add-ons IW_GIL 100 mit der Transaktion SAINT
- 2313222 – Deinstallation des Add-ons IW_SPI 100 mit der Transaktion SAINT

Im Folgenden stellen wir nun die verschiedenen Geschäftsobjekte näher vor, die als Datenquellen dienen können.

Generic Interaction Layer (GenIL)

Bestehende Geschäftslogik integrieren

Die Integration mit dem *Generic Interaction Layer* (GenIL) aus SAP Gateway bietet Ihnen die Möglichkeit, OData-Services auf Basis existierender GenIL-Objekte zu generieren. GenIL fungiert als Schnittstelle für existierende Geschäftslogik und bietet die Option, auf verschiedene Geschäftsobjekte über eine einheitliche Schnittstelle zuzugreifen. Sie ist Teil des *Business Object Layers* (BOL), der aus den zwei folgenden Ebenen besteht:

- **GenIL**
  Die untere der beiden Ebenen ist ein Dispatcher, der GenIL-Komponenten und deren Modelle zur Laufzeit verwaltet und Anfragen von darüberliegenden Komponenten auf diejenigen Komponenten verteilt, die die angefragten Objekte implementieren.
- **BOL**
  Diese zustandsbehaftete Ebene bietet einen performanten Zugriff, indem sie als Puffer für die Benutzeroberfläche dient und so wiederholte Zugriffe auf die Schnittstellen zu den darunterliegenden Geschäftsobjekten vermeidet.

Obwohl der BOL für den SAP CRM Web Client entwickelt wurde, ist der GenIL nicht hierauf beschränkt und auch in anderen Integrationsszenarien einsetzbar. Ein Beispiel hierfür ist das Webservice-Werkzeug, mit dem aus GenIL-Objekten SOAP-Webservices generiert werden können, die den Zugriff auf GenIL-Objekte über das SOAP-Protokoll ermöglichen. Auf ähnliche Weise bietet Ihnen SAP Gateway die Möglichkeit, OData-Services auf Basis von GenIL-Objekten zu generieren (siehe Abbildung 5.23).

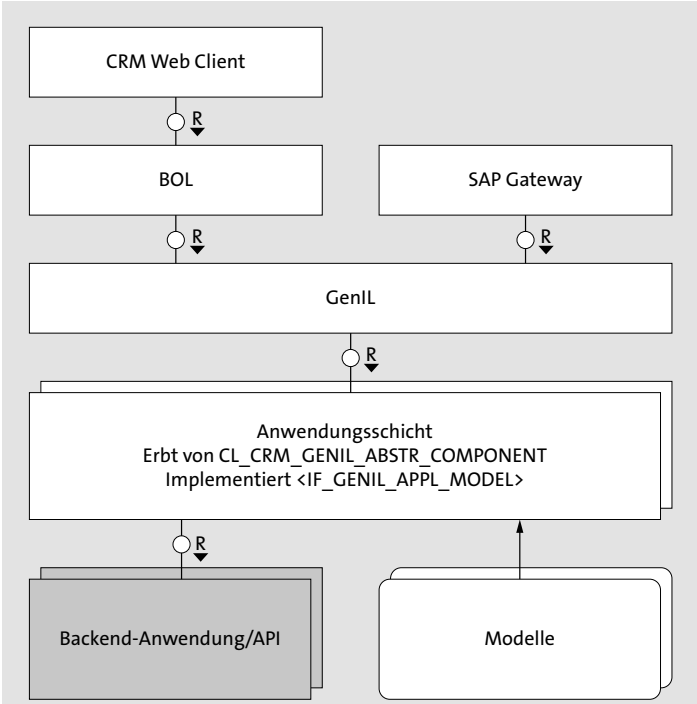


Abbildung 5.23 Integration von GenIL mit SAP Gateway

Die Knoten, Relationen und Queries des GenIL-Modells werden dabei, wie in Abbildung 5.24 gezeigt, auf entsprechende Entitäten eines OData-Modells gemappt.

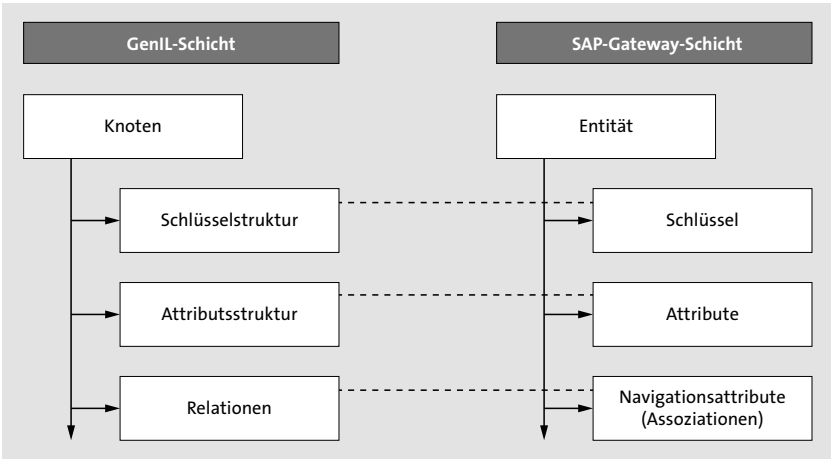


Abbildung 5.24 Mapping zwischen GenIL und einem OData-Modell

Obwohl die Schichten BOL und damit auch GenIL für den SAP CRM Web Client entwickelt wurden, wurde GenIL auch in anderen Business-Suite-Anwendungen wie SAP ERP Financials und SAP ERP Human Capital Management (HCM) eingesetzt. Die Integration mit SAP Gateway ist im Add-on `IW_GIL` enthalten. Dieses muss lokal im SAP-Backend-System (z. B. in SAP CRM) installiert werden und erfordert seinerseits die Installation des Add-ons `IW_BEP`.

Services auf Basis von GenIL-Objekten

Die GenIL-Integration ist nicht remote aufrufbar. Um Services nutzen zu können, die auf Basis von GenIL-Objekten generiert wurden, muss das Add-on `IW_GIL` im SAP-Backend-System installiert sein, das seinerseits das Add-on `IW_BEP` oder ab 7.40 die Softwarekomponente `SAP_GWFND` erfordert.

Service Provider Infrastructure (SPI)

Die Service Provider Infrastructure (SPI) wurde ursprünglich für die Anwendung SAP PLM entwickelt. Bei SPI handelt es sich um ein Framework auf der Anwendungsebene, das verschiedene Konsumenten hat. Das SPI-Framework wurde nicht nur von der Anwendung SAP PLM genutzt, für die es ursprünglich entwickelt wurde, sondern auch von vielen weiteren Anwendungen der SAP Business Suite.

SPI-Objekte sind remote aufrufbar. Aus diesem Grund ist es nicht zwingend erforderlich, dass das Gateway-Add-on `IW_SPI` auf dem SAP-Backend-System installiert wird. Da das Add-on die RFC-Schnittstelle des SPI-Layers aufruft, kann es auch auf dem Gateway-Server installiert werden. Das Add-on `IW_GIL` muss, wie erwähnt, lokal auf dem SAP-Backend-System (z. B. SAP CRM) installiert werden. Die Integration von SPI mit SAP Gateway ermöglicht es Ihnen, SPI Application Building Blocks als OData-Services zu publizieren.

SPI Application Building Blocks als OData-Services publizieren

Weiterführende Informationen zu diesem Thema finden Sie hier:

- SPI-Wiki in der SAP Community: *http://wiki.sdn.sap.com/wiki/display/SPI*
- SAP-Onlinehilfe: *http://s-prs.de/v671713*

Analytische Queries

Analytische Queries sind die wichtigsten Werkzeuge für den Zugriff auf analytische Daten, die sich entweder in Business-Anwendungen, wie z. B. der SAP Business Suite, oder in Data-Warehouse-Systemen befinden, wie z. B. SAP Business Warehouse (BW).

Während analytische Queries in der SAP Business Suite den Zugriff auf konsistente, operationale Daten ermöglichen, bieten analytische Queries im BW-Hub den Zugriff auf konsistente, stark aggregierte Daten aus verschiedenen Unternehmensteilen.

Die Integration von SAP Gateway und SAP BW ermöglicht es Ihnen, die Inhalte von SAP BW über einen OData-Service zu publizieren, der auf Basis von multidimensionalen Ausdrücken (*Multidimensional Expressions*, MDX) oder SAP BW Easy Query entwickelt wurde. Während die Verwendung der MDX-Queries bereits mit BW-Systemen ab Release 7.0 möglich ist, benötigt man für die Integration von BW Easy Query mindestens Release SAP BW 7.30. BW Easy Queries sind jedoch im Vergleich zu MDX-Queries einfacher zu verstehen und zu verwenden.

Analytische Annotationen

Mit SAP S/4HANA wurden CDS Views mit analytischen Annotationen eingeführt. Bei CDS Views mit der Annotation `@Analytics.query:true` handelt es sich um transiente Abfragen, die durch die Analytic Engine verarbeitet werden können, während es sich bei CDS Views mit der analytischen Annotation `@Analytics.dataCategory:#CUBE` um transiente SAP-BW-Provider handelt.

SAP BW Easy Query

SAP BW Easy Query bietet analytische Queries, die bestimmte Voraussetzungen erfüllen. Für jede BW Easy Query wird vom System ein RFC-Funktionsbaustein erzeugt. Dies geschieht automatisch auf Basis der vorhandenen BW-Query-Definition. Auf Basis dieses RFCs kann ein entsprechender OData-Service erzeugt werden.

Um eine analytische Query als SAP BW Easy Query freigeben zu können, müssen Sie die entsprechende Checkbox in den Query-Attributen im BEx Query Designer (siehe Abbildung 5.25) aktivieren.

Danach erfolgt die Generierung eines RFC-Funktionsbausteins. Die generellen Merkmale einer SAP BW Easy Query sind, dass sich Merkmale in den

Zeilen und die Kennzahlen in den Spalten befinden und freie Merkmale nicht auf OData gemappt werden.

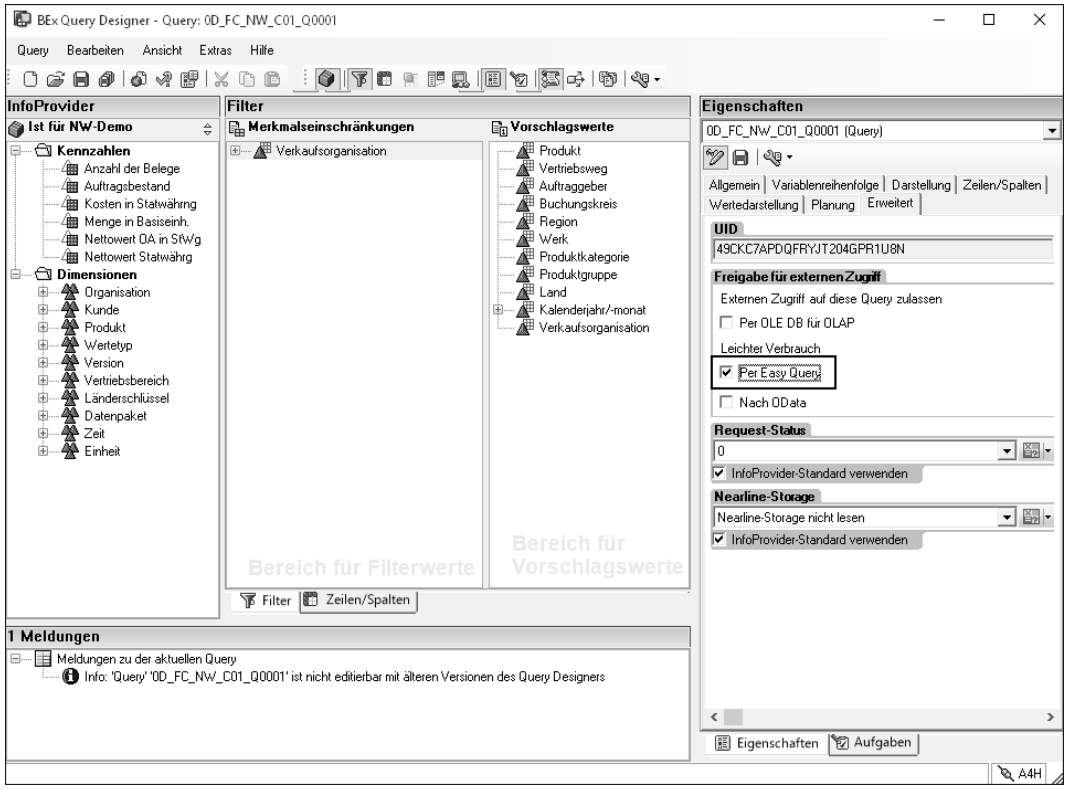


Abbildung 5.25 Definition einer Easy Query im BEx Query Designer

SAP BW Easy Query

Weiterführende Informationen über SAP BW Easy Query können Sie in der SAP-Onlinehilfe finden. Wir empfehlen an dieser Stelle das folgende Dokument: *http://s-prs.de/v671714*

Analytische Annotationen

Dimensionen, Dimensionsattribute und Maße werden als Attribute eines Entitätstyps dargestellt. Entitätstypen, die die Ergebnisse einer MDX-Query oder SAP BW Easy Query darstellen, sind als `sap:semantics=aggregate` gekennzeichnet.

Tabelle 5.2 zeigt, wie BW-Objekte, z. B. Dimensionen, Dimensionsattribute und Maße, in OData dargestellt werden. Die Tabelle zeigt nur die wichtigsten Annotationen.

SAP-BW-Objekte	OData-Repräsentation	SAP-Annotation
Cube vom Typ Query	Entitätstyp	`sap:semantics=aggregate`
Dimension	Eigenschaft	`sap:aggregation-role=dimension`
Dimensions-attribut	Eigenschaft	`sap:attribute-for=<dimension name>`
Maß	Eigenschaft	`sap:aggregation-role=measure`

Tabelle 5.2 Analytische Annotationen

Externer OData-Service

OSCI

Das Integrationsszenario *OData Services Consumption and Integration* (OSCI) zielt darauf ab, beliebige (auch Drittanbieter-) OData-Services aus SAP Gateway zu konsumieren und als Gateway-Services zu publizieren. Mit SP07 von SAP Gateway 2.0 ist diese Funktionalität vollständig in den Service Builder integriert worden.

Die Integration muss auf dem Gateway-Serversystem implementiert werden, wo auch das Add-on `IW_BEP` installiert werden muss. Dies ist notwendig, weil für die Konsumierung von OData-Services die OData-Bibliothek benötigt wird, die sich nur auf dem Gateway-Server befindet. Darüber hinaus ist auch das Add-on `IW_BEP` für die Entwicklung des Service auf dem Gateway-Server erforderlich.

Mit SAP NetWeaver ABAP 7.40 SP02 werden diese beiden Voraussetzungen von jedem ABAP-System erfüllt, da bei diesen Systemen die Softwarekomponente `SAP_GWFND` die entsprechenden Funktionalitäten enthält.

Bitte beachten Sie, dass die OSCI-Integration viele technische Einschränkungen aufweist, die in SAP-Hinweis 1574568 beschrieben sind (siehe auch *http://s-prs.de/v671762*).

OData-Service (SAP Gateway)

Erweiterungskonzept

Der Service Builder erlaubt Ihnen auch die Generierung von Services, die auf OData-Services in SAP Gateway basieren. Dieses Integrationsszenario ist Teil des Erweiterungskonzepts von SAP-Gateway-Services und kann z. B. für die Erweiterung von Services genutzt werden, die von SAP mit SAP Fiori ausgeliefert werden.

Der redefinierte Service kann zum einen auf die Funktionen des Originalservice zugreifen und zum anderen um zusätzliche Attribute oder Entitäts-

mengen erweitert werden. Auf diese Weise ist auch eine Erweiterung von Services möglich, die auf Basis der zuvor beschriebenen Integrationsszenarien entwickelt wurden. Das Vorgehen bei der Erweiterung eines von SAP ausgelieferten Gateway-Service und einer SAPUI5-Anwendung zeigen wir anhand der SAP-Fiori-Referenz-App »Approve Purchase Orders« in einer detaillierten Schritt-für-Schritt-Anleitung in Abschnitt 11.3, »SAP-Fiori-Anwendungen erweitern«.

5.4.6 Servicegenerierung mit referenzierten Datenquellen

Mit der Verfügbarkeit von SAP HANA kam es zu einem Paradigmenwechsel hinsichtlich der Art und Weise, wie Geschäftsanwendungen von SAP auf Basis des ABAP-Programmiermodells für SAP Fiori entwickelt werden. Der Zugriff auf Daten in SAP S/4HANA basiert auf CDS und OData. Dies ist möglich, da CDS nicht nur reine Leseszenarien unterstützt, sondern auch transaktionale, analytische Szenarien und Szenarien, die auf Suchen beruhen.

SAP Fiori Elements

Mithilfe von CDS können Datenmodelle über Annotationen semantisch angereichert und durch *SAP Fiori Elements* (ehemals Smart Templates) konsumiert werden. SAP Fiori Elements sind »schlau«, da sie beispielsweise in der Benutzeroberfläche ein Eingabefeld bereitstellen, wenn eine Eigenschaft als `sap:updatable` gekennzeichnet ist. CDS Views können auch sehr leicht erweitert werden. Mit der Option **Referenzierte Datenquellen** können Sie als ABAP-Entwickler in der Transaktion SEGW dynamische OData-Services generieren (siehe Abbildung 5.26).

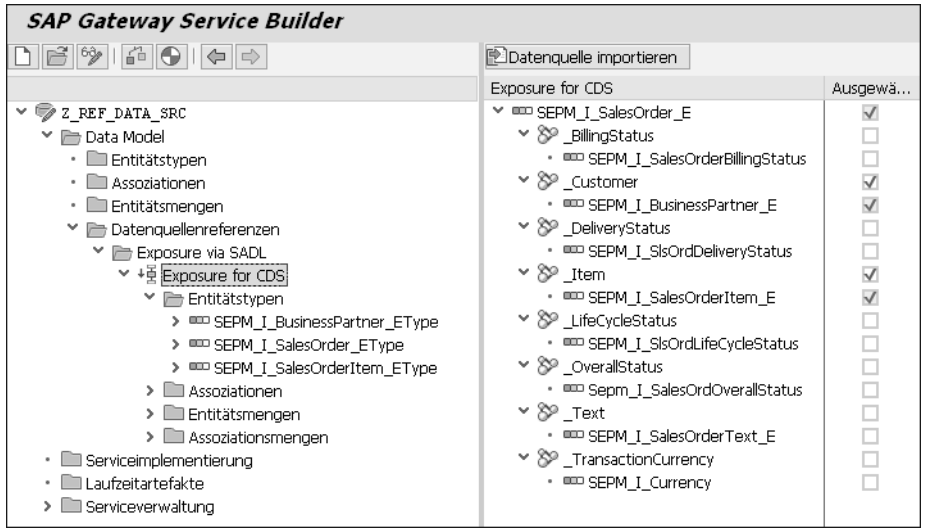


Abbildung 5.26 CDS View als referenzierte Datenquelle in der SEGW

Das bedeutet, dass sich ein OData-Service, der über eine referenzierte Datenquelle generiert wurde, automatisch an die Definition des zugrunde liegenden CDS Views anpasst. Im Service Builder können Sie die Entitäten und Assoziationen auswählen, die Teil des generierten Service sein sollen.

5.4.7 Servicegenerierung mit Eclipse mit der Option »@OData.publish:true«

Ähnlich wie bei den referenzierten Datenquellen im Service Builder können in den ABAP Development Tools in Eclipse aus CDS Views mithilfe der Option `@OData.publish:true` OData-Services generiert werden. Durch das einfache Setzen einer einzelnen Annotation im DDL-Quellcode des CDS Views wird auf Basis des CDS Views ein OData-Service im SAP-Backend-System registriert. Technisch wird eine Modell-Provider-Klasse im SAP-Backend generiert und mit einer generischen Daten-Provider-Klasse als OData-Service registriert. Um den OData-Service zu publizieren, müssen Sie als Entwickler oder der Administrator den im SAP-Backend registrierten Service mithilfe der Transaktion /IWFND/MAINT_SERVICE im SAP-Gateway-Hub aktivieren.

Im Gegensatz zu allen anderen bislang vorgestellten Möglichkeiten, OData-Services zu generieren, wird hier der Service Builder nicht verwendet.

Mit OData-Services, die über die Option `OData.publish:true` publiziert wurden, kann nicht nur ein rein lesender Zugriff unterstützt werden, sondern es ist auch möglich, transaktionale Services zu entwickeln. Dazu können parallel zu dem OData-Service entsprechende BOPF-Objekte generiert werden. Die so generierten OData-Services bieten dann auch die Möglichkeit, schreibend auf SAP-Anwendungsdaten zuzugreifen.

Welche der beiden Möglichkeiten im Einzelfall genutzt werden sollte, um einen OData Service auf Basis eines CDS Views zu generieren, ist in der SAP-Onlinehilfe beschrieben: *http://s-prs.de/v671715*.

Zusammenfassend kann man sagen, dass der Ansatz über referenzierte Datenquellen viele Möglichkeiten bietet, die beim Einsatz der Option `@OData.publish:true` nicht genutzt werden können.

5.5 OData-Channel

Nachdem wir die grundlegenden Konzepte der verschiedenen Möglichkeiten für die Erstellung von Gateway-Services beschrieben haben, stellen wir nun das Konzept der OData-Channel-Programmierung vor. Diese Einfüh-

rung ist essenziell für das Verständnis von Kapitel 6, »Serviceentwicklung«, in dem wir die Entwicklung von OData-Services detailliert beschreiben werden.

Der OData-Channel ermöglicht Ihnen die Entwicklung von Services durch die Definition von Objektmodellen und die Registrierung der entsprechenden Laufzeit, die in der Daten-Provider-Klasse implementiert wurde. Der Vorteil des OData-Channels ist, dass er Ihnen gewisse Freiheiten bei der Entwicklung von Services bietet. So können komplette DDIC-Informationen und lokale Schnittstellen im SAP-Backend genutzt werden, um Gateway-Services zu entwickeln.

Darüber hinaus können `QUERY`-Optionen auch im SAP-Backend-System genutzt werden. Dies bedeutet, dass nur die Daten, die vom OData-Client abgefragt wurden, auch aus dem SAP-Backend-System gelesen und zurück zum Client geschickt werden. Das Ergebnis sind hoch optimierte Services mit einer optimalen Performance, da nur ein Minimum an Daten an den Client gesendet wird.

Vier Komponenten von Gateway-Services

Gateway-Services bestehen aus der Sicht des OData-Programmiermodells aus vier Komponenten:

- der Implementierung einer *Modell-Provider-Klasse* und deren Basisklasse, die die Modelldefinition zur Laufzeit darstellt
- der Implementierung einer *Daten-Provider-Klasse* und deren Basisklasse, die zur Laufzeit aufgerufen wird, um die Datenanfragen auszuführen
- dem *technischen Servicenamen*, der zusammen mit dem *technischen Modellnamen* für die Registrierung des Service im SAP-Backend-System genutzt wird
- dem *technischen Modellnamen*, der zusammen mit dem *technischen Servicenamen* für die Registrierung des Service im SAP-Backend-System genutzt wird

Der technische Servicename und der technische Modellname werden zusammen mit den Klassennamen auf Basis des Projektnamens im Service Builder automatisch vorgeschlagen.

5.5.1 Modell-Provider-Klasse

Die Modell-Provider-Klasse (*Model Provider Class*, MPC) ist eine ABAP-Klasse, die die Definition des Datenmodells zur Laufzeit erzeugt. Aus diesem Grund werden alle Modellinformationen, die man im Projekt definiert hat, in der Modell-Provider-Basisklasse generiert. Die Basisklasse der

Modell-Provider-Klasse muss deshalb immer dann neu generiert werden, wenn Sie die Modelldefinition in Ihrem Projekt ändern.

Die Modell-Provider-Klasse ist wichtig, da alles, was man im Servicemetadatendokument eines Service findet, in der Modell-Provider-Klasse auch durch ABAP-Programmierung implementiert wurde (siehe Abbildung 5.27).

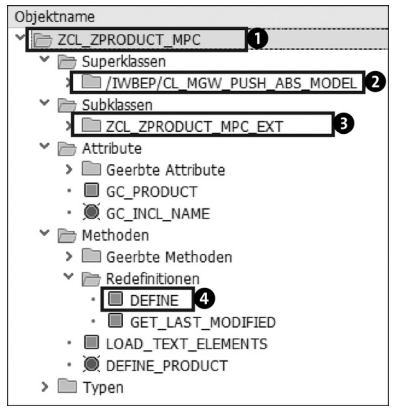


Abbildung 5.27 Modell-Provider-Klasse

Klassen

Genau betrachtet, wird die Modelldefinition in zwei verschiedenen Klassen generiert:

- Der Basisklasse ❶ (mit dem Suffix `_MPC`): Technisch wird die Basisklasse von der folgenden Superklasse ❷ abgeleitet: `/IWBEP/CL_MGW_PUSH_ABS_MODEL`.
- Der Modell-Provider-Erweiterungsklasse ❸ (mit dem Suffix `_MPC_EXT`): Die Modell-Provider-Erweiterungsklasse ist die Klasse, die über den technischen Modellnamen im Backend registriert wird. In der Modell-Provider-Erweiterungsklasse kann man wählen, welche Methoden redefiniert und welche Methoden von der Basisklasse übernommen werden.

In den meisten Fällen besteht für Sie kein Grund, die Modell-Provider-Klasse zu ändern, die durch den Service Builder generiert wurde. Die Ausnahme von dieser Regel ist, wenn Sie einen Gateway-Service bauen möchten, der Features besitzt, die derzeit nicht mit dem Service Builder modelliert werden können. In diesem Fall können Sie Methoden, z. B. die `DEFINE`-Methode ❹, in der Modell-Provider-Erweiterungsklasse redefinieren.

Modell-Provider-Klasse

Normalerweise ist es nicht erforderlich, dass Sie Methoden in der Modell-Provider-Klasse redefinieren, die vom Service Builder auf Basis des OData-Datenmodells generiert wurden. Dennoch schauen wir uns die Methoden

genauer an, die vom Service Builder generiert wurden, um das darunterliegende Framework besser zu verstehen.

Die DEFINE-Methode in der Basisklasse der Modell-Provider-Klasse, die vom Service Builder generiert wird, enthält Aufrufe für die entitätstypspezifischen define_<entity_type>-Methoden. Darüber hinaus enthält die DEFINE-Methode einen Aufruf der define_Association-Methode, die die Assoziationen, Assoziationsmengen, Referential Constraints und Navigationsattribute erzeugt.

Die Methode GET_LAST_MODIFIED ist wichtig für die Kommunikation zwischen dem SAP-Backend-System und dem Gateway-Server, wenn die Modell-Provider-Erweiterungsklasse im Backend geändert wurde. In diesem Fall wird ein Refresh der gecachten Metadaten des Service im Gateway-Server initiiert. Diese Methode sollte nicht manuell geändert werden.

In den entitätstypspezifischen DEFINE-Methoden generiert der Service Builder den ABAP-Code, der die Teile des OData-Modells erzeugt, das die Entitätstypen und die Entitätsmengen generiert, die auf diesem Entitätstyp basieren.

Die Attribute eines Entitätstyps werden durch den folgenden Code erzeugt, und die Attribute, die als Schlüsselfelder markiert wurden, werden im Code als Schlüsselfelder gesetzt (siehe Listing 5.3).

```
lo_property = lo_entity_type->
create_property( iv_property_name = 'ProductID'
iv_abap_fieldname = 'PRODUCT_ID' ).
lo_property->set_is_key( ).
```

Listing 5.3 Erzeugen von Eigenschaften in der Modell-Provider-Klasse

Schlussendlich wird der Entitätstyp an eine DDIC-Struktur gebunden, und eine oder mehrere Entitätsmengen werden angelegt (siehe Listing 5.4). Beachten Sie: Wenn ein Entitätstyp an eine existierende DDIC-Struktur gebunden wird, kann er Konvertierungsroutinen und die Feldbezeichner der Datenelemente aus dem DDIC nutzen. Der mittlere Feldbezeichner mit einer Länge von 15 Zeichen eines Datenelements wird dabei als Default-Wert für die SAP-spezifische Annotation sap:label verwendet.

```
...
lo_entity_type->
bind_structure( iv_structure_name   =
'BAPI_EPM_PRODUCT_HEADER' iv_bind_conversions = 'X' ).
...
lo_entity_set = lo_entity_type->
create_entity_set( 'Products' )
```

Listing 5.4 Anlegen von Entitätsmengen

In der Methode DEFINE_ASSOCIATION finden Sie den generierten Code, der die Assoziationen, Assoziationsmengen, Referential Constraints und Navigationsattribute des OData-Modells definiert.

5.5.2 Daten-Provider-Erweiterungsklasse und -Basisklasse

Die Daten-Provider-Klasse (DPC) ist eine ABAP-Klasse, die alle Methoden bereitstellt, um OData-Aufrufe zu beantworten. Sie wird zur Laufzeit aufgerufen, um diese weiteren Aufrufe auszuführen. Daher sprechen wir auch von der Laufzeitrepräsentation der Serviceimplementierung. Die Daten-Provider-Klasse führt unter anderem die Operationen CREATE, READ, UPDATE, DELETE und QUERY aus.

Daten-Provider-Klasse und Basisklasse

Wie im Fall der Modell-Provider-Klasse findet man die Daten-Provider-Erweiterungsklasse (*Data Provider Extension Class* ❶) mit der Endung _DPC_EXT und eine Basisklasse (*Data Provider Class* ❷) mit der Endung _DPC. Die Daten-Provider-Klasse erbt von der Basisklasse (siehe Abbildung 5.28). Die Daten-Provider-Erweiterungsklasse ist die Klasse, die über den technischen Servicenamen registriert wird, das heißt, sie ist die Klasse, die in einem OData-Service ausgeführt wird.

Entitätsmengentypische Methoden

An dieser Stelle ist es wichtig zu erwähnen, dass es neben den entitätsmengenspezifischen Methoden ❸ auch Methoden gibt, die nicht spezifisch für eine einzelne Entitätsmenge sind.

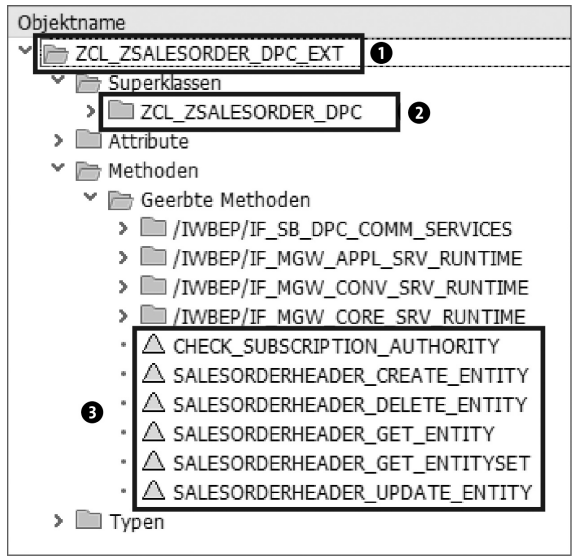


Abbildung 5.28 Interface der Daten-Provider-Klasse

Daten-Provider-Klasse

Für jede Entitätsmenge werden vom Service Builder Methoden angelegt, die vom Framework aufgerufen werden, wenn ein CREATE-, READ-, UPDATE- oder DELETE-Vorgang (CRUD) für die Entitätsmenge aufgerufen wird. Für eine Entitätsmenge mit dem Namen <ENTITYSET> werden die in Tabelle 5.3 aufgeführten Methoden angelegt.

Name der DPC-Methode	HTTP-Verb	Ziel
<ENTITYSET>_CREATE_ENTITY	POST	Entitätsmenge
<ENTITYSET>_DELETE_ENTITY	DELETE	Entität
<ENTITYSET>_GET_ENTITY	GET	Entität
<ENTITYSET>_GET_ENTITYSET	GET	Entitätsmenge
<ENTITYSET>_UPDATE_ENTITY	UPDATE	Entität

Tabelle 5.3 Entitätsmengenspezifische CRUD-Methoden

Darüber hinaus gibt es weitere Methoden, die nicht nur für eine einzelne Entitätsmenge gelten, sondern für alle Entitätsmengen, die sogenannten nicht entitätsmengenspezifischen Methoden.

Beispiele hierfür sind die Methoden, die die Ausführung von $EXPAND-oder Deep-Insert-Ausdrücken übernehmen, oder die Methoden, die aufgerufen werden, wenn ein Funktionsimport aufgerufen wird. Lassen Sie uns daher einen Blick auf diese Beispiele werfen:

- GET_EXPANDED_ENTITY, GET_EXPANDED_ENTITYSET
  Die Ausführung von $expand-Ausdrücken wird vom SAP Gateway Framework out of the box generisch angeboten, wenn Ihr Modell geeignete Navigationsattribute enthält und das Handling der Navigationsattribute implementiert wurde. Es wird jedoch oft Situationen geben, in denen man die Ausführung der $expand-Aufrufe selbst implementieren möchte. Ein Beispiel hierfür sind RFC-Funktionsbausteine wie BAPI_EPM_SO_GET_LIST. Dieser Funktionsbaustein liest zusammen mit den Kopfdaten eines Verkaufsauftrags auch dessen Einzelposten. Wird nun die Entitätsmenge, die die Kopfdaten der Verkaufsaufträge enthält, so aufgerufen, dass aufgrund des $expand-Ausdrucks zugleich auch die Einzelposten des Verkaufsauftrags gelesen werden sollen, würden unnötige Zugriffe auf die Datenbank erfolgen, da die Einzelposten schon beim Lesen der Header-Daten aus der Datenbank gelesen wurden.

- CREATE_DEEP_ENTITY
  Das Gegenstück zum Schlüsselwort $expand ist die Deep-Insert-Anweisung, bei deren Aufruf die CREATE_DEEP_ENTITY-Methode ausgeführt wird. Ein typisches Beispiel hierfür ist der Fall, bei dem zusammen mit einem Verkaufsauftrag zwingend auch mindestens ein Einzelposten angelegt werden muss.
- EXECUTE_ACTION
  Die EXECUTE_ACTION-Methode ist ebenfalls nicht entitätsmengenspezifisch. Diese Methode wird aufgerufen, wenn ein Funktionsimport in einem OData-Service ausgeführt werden soll. Funktionsimporte ermöglichen die Ausführung von lesenden und schreibenden Szenarien.
- Funktionsimporte können in einem Business-Szenario eingesetzt werden, wenn Daten gelesen oder geschrieben werden müssen und diese Funktionen nicht durch den Aufruf von CRUD-Q-Methoden abgebildet werden können.

5.5.3 Technische Überlegungen zur OData-Channel-Entwicklung

Die OData-Channel-Entwicklung kann entweder im SAP-Backend-System oder auf dem Gateway-Server stattfinden, wie in Abbildung 5.29 dargestellt. Wo auch immer Sie entwickeln möchten: Die BEP-Komponente muss in dem entsprechenden System installiert werden, oder Sie müssen ein System verwenden, das auf SAP NetWeaver 7.40 SP2 oder höher basiert.

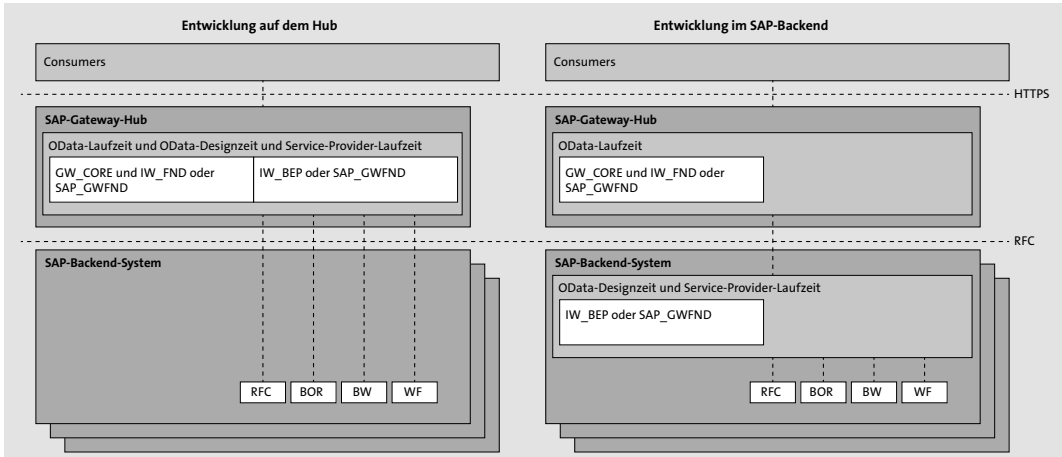


Abbildung 5.29 OData-Channel-Entwicklung auf dem Hub oder im SAP-Backend-System

5.6 Zusammenfassung

Das Anlegen von OData-Services folgt dem bereits vorgestellten Prozess der Erzeugung von Gateway-Services. Dieses Vorgehen wird von SAP empfohlen und durch den Service Builder optimal unterstützt.

In diesem Kapitel haben wir Ihnen zunächst die Entwicklungswerkzeuge und den Entwicklungsprozess vorgestellt, um Ihnen ein Basis-Know-how für die technisch anspruchsvollen Schritt-für-Schritt-Anleitungen zur Serviceentwicklung und Servicegenerierung in Kapitel 6, »Serviceentwicklung«, Kapitel 7, »Servicegenerierung«, und Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«, zu vermitteln. In Kapitel 6 werden Sie die Vorteile des OData-Channel-Programmiermodells, das wir in diesem Kapitel vorgestellt haben, nutzen können.

Einleitung

Es ist gerade einmal 30 Jahre her, dass wir anfingen, im World Wide Web zu surfen. Zu ungefähr der gleichen Zeit begannen wir auch, Mobiltelefone im Alltag zu verwenden. Damals dachten nur wenige Menschen daran, diese beiden Technologien zusammenzubringen. Noch weniger Menschen rechneten damit, dass die Kombination dieser beiden Technologien so populär werden würde, wie sie heute ist, und an »die Cloud« hat schon gar niemand gedacht.

Wenn Sie sich noch an diese Zeit erinnern, werden Sie wissen, dass es Firmen wie Google oder Amazon noch gar nicht gab. Geräte wie iPods, iPhones oder iPads waren noch nicht einmal erfunden. Selbst Nokia – viele Jahre einer der Marktführer in der Mobiltelefonsparte – begann gerade erst, sich in diesem Bereich zu etablieren. Wenn Sie sich also an die letzten 25 Jahre erinnern können (und wir gehen davon aus, dass manche von Ihnen dies können), ist es leicht zu sehen, wie schnell sich der IT-Bereich entwickelt.

Springen wir ins Hier und Jetzt: Heute können wir nicht nur mit unserem PC im Web surfen, wir können dafür auch unser Smartphone, Tablet, Fernsehgerät oder die Spielekonsole nutzen und noch einiges mehr. Aus eben diesem Grund stehen wir allerdings vor der Herausforderung, dass wir heute Geschäftsanwendungen für eine gefühlt endlose Anzahl an Geräten bzw. Kanälen anbieten müssen. Verschärfend kommt hinzu, dass wir dabei nicht nur technologische Entwicklungen berücksichtigen müssen, sondern auch soziale Trends.

In diesem Buch geht es um ein Produkt, das sich dieser Herausforderung für SAP-Anwendungen erfolgreich stellt: *SAP Gateway*. Das Buch nimmt Sie mit auf eine Reise, auf der Sie alles über SAP Gateway lernen. Wenn Sie SAP Gateway noch nicht kennen, sollten Sie diese Reise am besten unternehmen, indem Sie das Buch vom ersten bis zum letzten Kapitel lesen. Wenn Ihnen allerdings schon einige Konzepte und Technologien vertraut sind, können Sie auch andere Reiserouten wählen: Wir haben das Buch bewusst so geschrieben, dass ein Springen zwischen den Kapiteln möglich ist. Um dies noch zu vereinfachen, haben wir das Buch nicht nur in Kapitel strukturiert, sondern zusätzlich auch in mehrere übergreifende Teile.

Teil I: Einstieg

Der erste Teil des Buches besteht aus vier Kapiteln, die sich den Grundlagen von SAP Gateway widmen. Hier sollten Sie Ihre Reise beginnen, wenn Sie

sich über SAP Gateway und verwandte Konzepte wie *OData* informieren möchten.

Einführung in SAP Gateway

Kapitel 1 ist eine grundlegende Einführung in SAP Gateway und erklärt die Motivation, die hinter der Entwicklung des Produkts steht. Das Kapitel schließt mit einer Positionierung des Produkts im Kontext anderer SAP-Produkte.

Einführung in OData

OData ist der Industriestandard, den SAP Gateway nutzt. Diesen Standard schauen wir uns in **Kapitel 2, »Einführung in OData«**, im Detail an.

Architektur und Integration

Kapitel 3, »Architektur und Integration«, führt in die Architektur von SAP Gateway ein und beleuchtet auch die Backend-Konzepte sowie die Integration mit anderen SAP-Schnittstellen.

Deployment-Optionen, Installation und Konfiguration

Kapitel 4, »Deployment-Optionen, Installation und Konfiguration«, schließt den ersten Teil des Buches mit einer Diskussion der Deployment-Optionen für SAP Gateway ab, die heute in echten, produktiven Systemlandschaften zu finden sind.

Teil II: Serviceerstellung

Als erfahrener Reisender, der sich gut mit SAP Gateway und OData auskennt, haben Sie vielleicht den ersten Teil des Buches komplett übersprungen. Andere Leser wiederum haben den ersten Teil des Buches komplett durchgearbeitet. Gleichgültig, was auf Sie zutrifft, in diesem zweiten Teil werden Sie alles lernen, was Sie über die Erstellung von Services in SAP Gateway wissen müssen.

Erstellung von OData-Services mit SAP Gateway

Kapitel 5, »Einführung in die Erstellung von OData-Services mit SAP Gateway«, erklärt die Ende-zu-Ende-Entwicklungswerkzeuge und den Entwicklungszyklus, um SAP-Gateway-Services zu erstellen. Es führt Sie in die zwei hauptsächlichen Methoden der Serviceerstellung ein: Serviceentwicklung und Servicegenerierung. Dieses Kapitel ist die Basis für die weiteren Kapitel in Teil II.

Serviceentwicklung

Serviceentwicklung ist das Thema im gleichnamigen **Kapitel 6**. In diesem Kapitel lernen Sie, wie Sie Services im SAP-Backend mit ABAP entwickeln. Dabei konzentriert sich das Kapitel auf die praxisbezogenen Aspekte bei der Erstellung von OData-Services.

Servicegenerierung

Kapitel 7, »Servicegenerierung«, stellt die zweite Methode der Erstellung von Services vor: die Servicegenerierung. Es erklärt die Generierung von OData-Services im SAP-Backend.

ABAP-Programmiermodell für SAP Fiori

Die Servicegenerierung für SAP Fiori und SAP S/4HANA wird in **Kapitel 8, »ABAP-Programmiermodell für SAP Fiori«**, betrachtet. In diesem Kapitel

lernen Sie, wie Sie diese mit *Core Data Services Views* (CDS Views) und dem *ABAP-Programmiermodell für SAP Fiori* umsetzen können.

Teil III: Anwendungsentwicklung

Wie jede Münze zwei Seiten hat, hat auch SAP Gateway zwei Seiten: *Bereitstellung* (*Provisioning*, Backend-Services und ihre Entwicklung) und *Konsumierung* (*Consumption*, Verwendung und Gebrauch von Backend-Services in Anwendungen). Während sich der zweite Teil des Buches auf die Bereitstellung konzentriert, ist der dritte Teil auf die Konsumierung ausgerichtet. Die verschiedenen Kapitel in diesem Teil zeigen aus unterschiedlichen Perspektiven, wie flexibel SAP Gateway ist und wie Sie OData-Services in verschiedenen Anwendungen konsumieren können.

SAPUI5-Applikationsentwicklung

In **Kapitel 9, »SAPUI5-Applikationsentwicklung«**, geht es um *SAPUI5*-Anwendungsentwicklung und *SAP Fiori*. SAPUI5 ist eine Sammlung von Bibliotheken, die Entwickler nutzen können, um Anwendungen zu bauen, die in einem Browser laufen, der HTML5 unterstützt. Bei SAP Fiori handelt es sich um eine Gruppe von Business-Anwendungen, die SAPUI5 nutzen.

SAP Web IDE

Um Anwendungen bauen zu können, brauchen Sie eine spezifische Entwicklungsumgebung. **Kapitel 10, »SAP Web IDE«**, widmet sich der Entwicklungsumgebung für SAP Fiori: der *SAP Web IDE*.

Erweiterbarkeit

Kapitel 11, »Erweiterbarkeit«, beschäftigt sich erneut mit SAP Fiori, allerdings liegt der Schwerpunkt in diesem Kapitel auf der Ende-zu-Ende-Erweiterbarkeit und wie man sie anwendet. Das Kapitel deckt außerdem ab, wie OData-Dienste erweitert werden können.

Entwicklung mobiler Apps

Eine der am häufigsten gestellten Fragen im Kontext der Anwendungsentwicklung ist die, wie mobile Anwendungen gebaut werden können. **Kapitel 12, »Entwicklung mobiler Apps«**, beantwortet diese Frage und führt durch einige Beispiele der Anwendungsentwicklung für mobile Anwendungen.

Social-Media-Applikationsentwicklung

Kapitel 13, »Social-Media-Applikationsentwicklung«, hat sich den Anwendungen für soziale Medien verschrieben. Hier lernen Sie, wie Sie Anwendungen entwickeln, die die Möglichkeiten der sozialen Medien Facebook, Twitter und Sina Weibo in Kombination mit OData und SAP Gateway nutzen.

Entwicklung von Unternehmensanwendungen

In **Kapitel 14, »Entwicklung von Unternehmensanwendungen«**, schließen wir den dritten Teil des Buches mit Beispielen der Konsumierung von OData-Services mit Anwendungen wie Microsoft Excel oder Microsoft SharePoint ab.

Teil IV: Administration

Lifecycle Management

Der vierte Teil des Buches behandelt die Administration von SAP Gateway. Zusätzlich zum Deployment von SAP Gateway ist es wichtig zu verstehen, wie die Software verteilt, das heißt ausgerollt wird, wie mit Fehlern zu verfahren ist etc. All diese Themen werden in **Kapitel 15, »Lifecycle Management: Qualitätssicherung, Service-Deployment und Operations«**, besprochen.

Sicherheit

Theoretisch sollte das Thema Sicherheit in Kapitel 15 mitbehandelt werden, allerdings ist es so wichtig und umfangreich, dass wir beschlossen haben, ihm ein eigenes Kapitel zu widmen. **Kapitel 16** beschäftigt sich mit der Sicherheit von SAP Gateway, von der Authentifizierung über die Autorisierung bis hin zum Single Sign-on (SSO).

Teil V: Ausblick

Aktuelle und zukünftige Entwicklungen

Im letzten Teil unseres Buches findet sich nur **Kapitel 17, »Aktuelle und zukünftige Entwicklungen«**, das einen Blick auf zukünftige Trends bietet. Wir werfen dabei einen tiefen Blick in die Kristallkugel zu Themen wie dem *Internet der Dinge* (*Internet of Things*) und *Gamification*.

Hilfestellungen

Informationskästen

In hervorgehobenen Informationskästen sind in diesem Buch Inhalte zu finden, die wissenswert und hilfreich sind, aber etwas außerhalb der eigentlichen Erläuterung stehen. Damit Sie die Informationen in den Kästen sofort einordnen können, haben wir die Kästen mit Symbolen gekennzeichnet:

- In Kästen, die mit dem Plussymbol gekennzeichnet sind, finden Sie Informationen zu *weiterführenden Themen*.

- *Hinweise* oder *Tipps* sind durch das Pfeilsymbol kenntlich gemacht.

Materialien zum Buch

Zusätzlich zu allen Informationen in diesem Buch haben wir Textversionen der Programmierbeispiele erstellt. Diese finden Sie auch auf der Webseite zum Buch unter *http://www.sap-press.de/4802* über den Kasten **Materialien zum Buch**.

Auf einen Blick

Inhalt

TEIL II Serviceerstellung