# Глава 3 Модель программирования ABAP RESTful

*В этой главе мы познакомим вас с моделью программирования ABAP RESTful, которая доступна на облачной платформе SAP с августа 2018 г. и в локальных установках с SAP S/4HANA 1909 в сентябре 2019 г. Модель программирования ABAP RESTful является стандартной моделью разработки для новых приложений SAP S/4HANA и отражает базовую архитектуру SAP S/4HANA, описанную в главе 1, и является преемником модели программирования ABAP для SAP Fiori, которую мы рассмотрели в первом издании этой книги.*

В целом, как и его предшественник, модель программирования ABAP RESTful поддерживает разработку оптимизированных для SAP HANA служб OData для приложений SAP Fiori на основе представлений основных служб данных (CDS) и охватывает аналитические, транзакционные и поисковые процессы приложений. При разработке приложений с использованием модели программирования существуют два основных сценария: приложения только для чтения и транзакционные приложения. Для приложений только для чтения требуется только базовая модель данных CDS и аналитика для конкретного приложения или поисковые аннотации. Затем модель данных CDS и ее аннотации предоставляются в виде службы OData с использованием технологии языка описания адаптации службы (SADL) соответственно во время выполнения запроса модели программирования. Транзакционные приложения, в дополнение к приложениям только для чтения, требуют создания бизнес-объекта на основе определения поведения и реализации для обработки операций создания, чтения, обновления и удаления (CRUD), а также дополнительной бизнес-логики, реализованной в ABAP. В следующих разделах мы рассмотрим различные технологии, связанные с моделью программирования RESTful (CDS, SAP Gateway и OData), прежде чем углубиться в саму модель. Мы начнем с его процесса разработки во время проектирования и его стека во время выполнения, а затем перейдем к запросам, бизнес-объектам и предоставлению услуг в деталях. Эта глава предоставит вам теоретические основы модели программирования ABAP RESTful, прежде чем мы начнем разработку приложений SAP Fiori в следующих главах.

## 3.1 Основные службы данных (Core Data Services)

Как упоминалось в главе 1, основные службы данных (CDS) являются основой для всех типов приложений SAP S/4HANA. CDS можно развернуть поверх устаревших или новых таблиц SAP ERP и обеспечить разработку семантически богатых моделей данных, которые способствуют переносу кода в базу данных SAP HANA. CDS разрабатываются в стеке ABAP в Eclipse и поэтому используют стандартное управление жизненным циклом объектов разработки ABAP; например, CDS передаются между системами с использованием стандартной системы изменений и транспортировки ABAP (CTS). Определение представления CDS имеет запись репозитория объектов R3TR DDLS <CDS\_DEFINITION\_VIEW\_NAME>.

Чтобы охватить различные сценарии приложений, представления CDS, определенные с помощью языка определения данных (DDL) CDS, можно расширить с помощью следующих типов аннотаций:

- *Аналитические аннотации (Analytical annotations)*

Чтобы использовать представление CDS в качестве куба данных или запроса в сценариях аналитических приложений через Analytical Engine, представление CDS необходимо аннотировать с помощью аннотаций ***@Analytics***.

- *Аннотации пользовательского интерфейса (UI annotations)*

Представления CDS можно аннотировать с помощью аннотаций пользовательского интерфейса (UI) (***@UI***), чтобы определить, где определенные объекты, поля и данные будут размещены в приложении-шаблоне элементов SAP Fiori, что значительно сокращает требуемый внешний код JavaScript SAPUI5. Аннотации пользовательского интерфейса можно переместить в файл расширения метаданных с типом репозитория объектов R3TR DDLX, чтобы не загромождать основное представление CDS множеством аннотаций пользовательского интерфейса.

- *Поисковые аннотации (Search annotations)*

Представления CDS можно настроить для сценариев поиска с помощью аннотаций ***@Search***, например, в качестве модели корпоративного поиска (ESH) для поиска на панели запуска SAP Fiori или для поиска в приложении SAP Fiori путем определения области и нечеткого текстового поиска SAP HANA.

Чтобы включить транзакционную обработку для сущностей CDS, необходимо смоделировать структуру бизнес-объекта с использованием специальных типов ассоциаций, которые мы рассмотрим в разделе о модели программирования ABAP RESTful. Ранее в среде обработки бизнес-объектов (BOPF) бизнес-объект также можно было создавать с помощью аннотаций. На рис. 3.1 показано значение технологий CDS для разработки приложений в рамках SAP S/4HANA. Определения данных CDS являются основой для создания моделей поиска ESH, моделирования бизнес-объектов и предоставления услуг OData. Моделирование бизнес-объектов и реализация его соответствующего поведения с использованием определения и реализации поведения являются частью модели программирования ABAP RESTful, как и подготовка OData, для которой требуется определение службы и привязка службы. Мы рассмотрим модель программирования ABAP RESTful более подробно позже, в разделе 3.4. Вся разработка происходит в Eclipse, за исключением разработки пользовательского интерфейса приложения SAP Fiori, которая выполняется в SAP Web IDE.

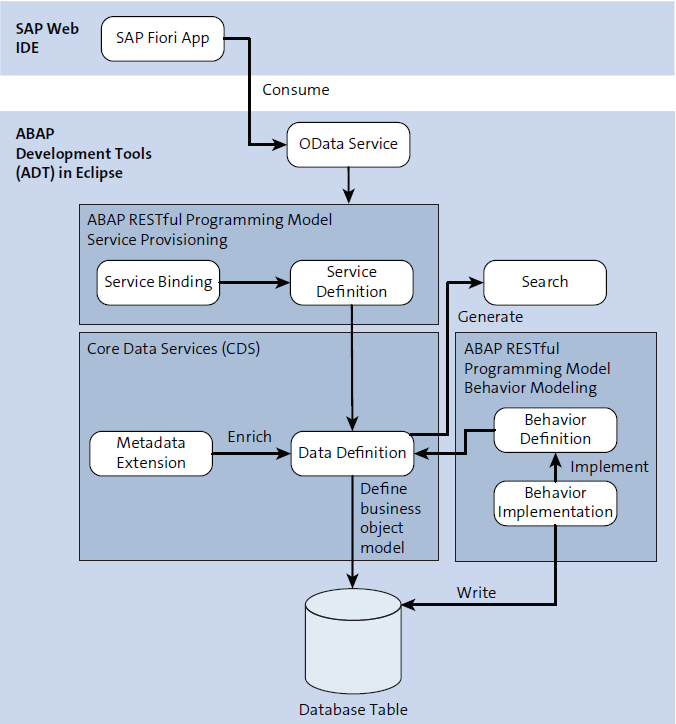


Рис. 3.1 CDS как основа для разработки приложений в SAP S/4HANA с использованием модели программирования ABAP RESTful

## 3.2 SAP-шлюз (SAP Gateway)

Шлюз SAP играет решающую роль в обеспечении простого в использовании доступа без использования ABAP к бизнес-данным, хранящимся в серверных системах SAP NetWeaver. Доступ к бизнес-данным предоставляется через службы OData на основе REST с использованием HTTP в качестве базового протокола передачи данных. Начиная с SAP NetWeaver версии 7.40, программный компонент ***SAP\_GWFND*** устанавливается вместе со стандартом SAP NetWeaver и включает в себя полный набор функций, необходимых для включения хаба и серверной части.

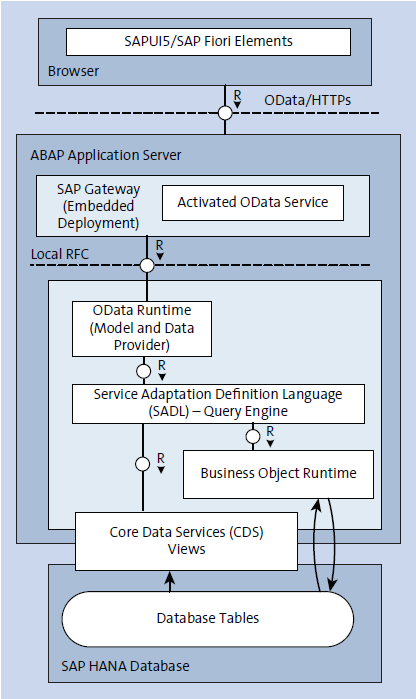


Рис. 3.2 Общий обзор типичной настройки системы SAP S/4HANA с шлюзом SAP, предоставляемым как встроенное развертывание

В общем, с архитектурной точки зрения существует два подхода к развертыванию: встроенное развертывание или развертывание в концентраторе. Развертывание концентратора может быть дополнительно разделено на разработку в концентраторе или разработку в серверной системе. В настоящее время рекомендуется использовать встроенное развертывание. SAP больше не рекомендует использовать дополнительную интерфейсную систему SAP Gateway, как это было рекомендовано до SAP S/4HANA 1809.

На рис. 3.2 показана установка системы SAP S/4HANA. Служба OData, развернутая на встроенном шлюзе SAP, и серверная часть обмениваются данными, используя соединение локального удаленного вызова функции (RFC) для пересылки запросов. Функциональный модуль с поддержкой RFC, передающий данные от внешнего интерфейса к внутреннему, называется ***/IWBEP/FM\_MGW\_HANDLE\_REQUEST***. Приложение SAP Fiori, работающее в браузере, отправляет HTTP-запросы OData GET, POST, DELETE или PUT в систему SAP Gateway, которая предоставляет доступ ко всем зарегистрированным и активированным службам OData. Шлюз SAP передает входящие запросы OData на серверную часть через локальное соединение RFC. Затем среда выполнения OData в серверной части делегирует фактический выбор данных на уровень инфраструктуры SADL. Если запрос является запросом GET только для чтения, SADL делегирует запрос своему механизму запросов, который генерирует оператор SQL SELECT для выбора данных запрошенного объекта OData из таблиц базы данных через соответствующее представление CDS. Для доступа на запись, например, POST для создания или PUT для обновления, запрос будет делегирован среде выполнения транзакционных бизнес-объектов модели программирования ABAP RESTful, которая будет обрабатывать обновления базы данных в соответствии с определенным и реализованным поведением бизнес-объектов.

## 3.3 OData

Как вы уже знаете из главы 1, вы можете напрямую выбирать данные из представления CDS, используя Open SQL в ABAP, точно так же, как вы могли бы выбирать данные из обычной таблицы базы данных. Однако для приложений SAP Fiori требуется предоставление данных в виде службы OData с использованием HTTP в качестве протокола передачи данных. OData — это протокол данных на основе REST, используемый для передачи бизнес-данных, а также метаданных между серверной системой ABAP и клиентскими приложениями через шлюз SAP. В SAP S/4HANA клиентскими приложениями служб OData обычно являются приложения SAP Fiori SAPUI5, работающие в локальных браузерах устройств конечных пользователей, таких как настольные ПК или планшеты. Вместе со шлюзом SAP OData обеспечивает доступ к внутренним бизнес-данным SAP простым для понимания и четко определенным способом, используя HTTP в качестве протокола передачи данных. Служба OData организует данные в виде сущностей, которые имеют набор свойств, связанных между собой ассоциациями. Эти элементы напоминают элементы моделей данных CDS, поэтому модели данных CDS идеально подходят для использования в качестве сервисов OData.

Вы можете изучить структуру службы OData, просмотрев документ службы и документ метаданных службы. Документ службы содержит список сущностей или ресурсов, к которым можно получить доступ с помощью этой службы, и которые можно запросить через путь: */sap/opu/odata/sap/<имя\_службы\_OData>/*. Кроме того, вы можете узнать, разрешает ли служба операции CRUD над сущностями, просмотрев атрибуты: ***sap:creable***, ***sap:updatable*** и ***sap:deletable*** тега ***<app:collection>***, который также содержит относительную ссылку на набор сущностей через атрибут ***href***, как показано на рис. 3.3.



Рис. 3.3 Сервисный документ простого документа покупки и сервиса OData элемента документа покупки

Документ метаданных службы намного более подробен, чем документ службы, и показывает все метаданные службы. Вы можете запросить документ метаданных службы, используя параметр ***$metadata***: */sap/opu/odata/sap/<OData\_service\_name>/$metadata*. В этом документе отображаются все сущности службы, включая их свойства и связи.

На рис. 3.4 показано, как будет выглядеть документ метаданных службы для простого сценария, который мы представили в главе 1. Вспомним, что представление CDS документа покупки (Z\_PurchasingDocumentDDL) связано с представлением элементов документа покупки (Z\_PurchasingDocumentItemDDL). Оба представления теперь отображаются как типы сущностей OData в документе метаданных службы OData, включая информацию о связи между двумя сущностями и их мощностях. Какие операции поддерживает сущность, указывают атрибуты *sap:creable*, *sap:updatable* и *sap:deletable* элемента ***EntitySet***. Обратите внимание, что в следующем примере служба вообще не поддерживает доступ для записи, поскольку мы не включили транзакционную среду выполнения для нашей модели CDS путем моделирования бизнес-объекта и его поведения в модели программирования ABAP RESTful. Таким образом, служба поддерживает доступ для чтения только через HTTP-запросы сущности GET. Как упоминалось ранее, файл метаданных службы OData можно запросить, используя путь $metadata: /sap/opu/odata/sap/<имя\_службы\_OData>/$metadata.



Рис. 3.4 XML-документ метаданных службы простого сценария CDS из главы 1, представленный как служба OData

OData использует HTTP-команды REST POST, GET, PUT и DELETE для операций CRUD над сущностями. Кроме того, OData определяет простой, но мощный язык запросов для ограничения набора результатов, предоставляемого шлюзом SAP. В таблице 3.1 перечислены наиболее распространенные параметры запросов OData.

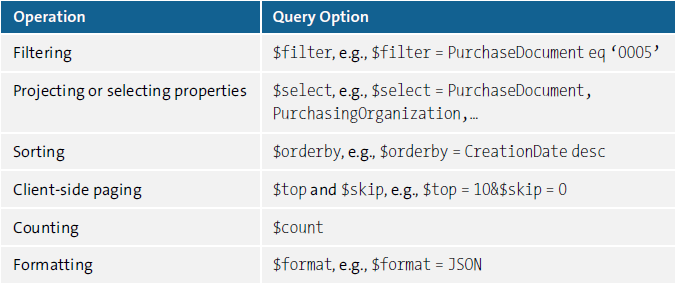


Таблица 3.1 Наиболее важные параметры запроса OData

Открытый доступ к серверным бизнес-данным через OData и SAP Gateway позволяет использовать современные не-ABAP-технологии пользовательского интерфейса для отображения бизнес-данных и взаимодействия с ними. Таким образом, пользовательский интерфейс приложений, разработанных с использованием модели программирования ABAP RESTful, полностью основан на платформе SAPUI5, которая реализует принципы проектирования SAP Fiori. Часть пользовательского интерфейса приложений обычно реализуется с помощью облачной SAP Web IDE и основана либо на приложениях элементов SAP Fiori, либо на приложениях свободного стиля. Приложения элементов SAP Fiori, например, список отчетов и шаблоны страниц объектов или планы этажей, адаптируют свои макеты на основе метаданных службы OData и аннотаций пользовательского интерфейса, определенных в представлениях CDS или их соответствующих расширениях метаданных. Таким образом, шаблоны элементов SAP Fiori значительно сокращают необходимый внешний код SAPUI5 JavaScript до минимума и значительно повышают производительность разработчиков, по-прежнему обеспечивая гибкость с использованием предопределенных точек расширения во внешнем интерфейсе.

Кроме того, прозрачные для разработчика приложения, элементы SAP Fiori реализуют необходимую обработку запросов CRUD для приложений только для чтения и транзакционных приложений, в зависимости от того, какие операции бизнес-объектов были определены и реализованы в определении поведения и реализации с использованием модели программирования ABAP RESTful. Приложения Freestyle, в отличие от элементов SAP Fiori, предоставляют разработчику внешнего интерфейса полную гибкость в отношении дизайна и логики пользовательского интерфейса, но требуют больших усилий на этапе разработки. Макет пользовательского интерфейса и его элементы управления должны быть объявлены разработчиком вручную, и должна быть реализована необходимая логика JavaScript SAPUI5. Риск при разработке приложений свободного стиля заключается в нарушении принципов проектирования SAP Fiori. Следовательно, мы рекомендуем использовать приложения свободного стиля только в том случае, если требуемый дизайн пользовательского интерфейса невозможно реализовать с помощью существующего плана этажа элементов SAP Fiori.

## 3.4 Модель программирования ABAP RESTful

Теперь давайте подробно рассмотрим модель программирования ABAP RESTful и познакомимся с ее различными компонентами. Как последняя модель программирования SAP для ABAP, разработка была сосредоточена на облаке. Основной целью модели программирования ABAP RESTful является разработка сервисов RESTful OData, которые могут использоваться приложениями SAP Fiori или любыми клиентами OData.

В отличие от предыдущей модели программирования ABAP для SAP Fiori, модель программирования ABAP RESTful была интегрирована непосредственно в ядро ​​языка ABAP за счет введения новых переносимых типов объектов разработки ABAP для ***определений поведения***, ***определений служб*** и ***привязок служб***, а также нового ***языка манипулирования сущностями*** (***EML***). В результате модель программирования может быть предложена в облаке как только основная среда выполнения языка ABAP без каких-либо дополнительных библиотек и инструментов (кроме шлюза SAP и среды оркестрации SADL, направляющей запросы либо на выборку данных, либо на рантайм бизнес-объекта).

В предыдущей модели программирования ABAP для SAP Fiori это было невозможно, поскольку она также полагалась на BOPF для обработки транзакций, который является структурой ABAP, а не частью языка ABAP. Кроме того, теперь вы можете разрабатывать сервисы OData end-to-end (E2E) в одной инструментальной среде (сейчас это Eclipse). Доступ к SAP GUI больше не требуется в процессе разработки, что также было требованием для облачной разработки и было раздражающим аспектом предыдущей модели программирования.

### 3.4.1 Введение

Модель программирования ABAP RESTful — это довольно самоуверенная структура, которая проведет вас через все шаги, необходимые для разработки службы RESTful OData. Классические приложения ABAP Web Dynpro, разработанные для SAP GUI, обладают более высокой степенью свободы, чем разработка с использованием модели программирования ABAP RESTful. Однако в результате приложения, разработанные с использованием модели программирования ABAP RESTful, более стандартизированы и, следовательно, их легче поддерживать в долгосрочной перспективе. В этом подразделе мы сначала рассмотрим процесс разработки, к которому привязан каждый разработчик при разработке службы OData с помощью модели программирования ABAP RESTful. Кроме того, мы рассмотрим компоненты, участвующие в обработке запросов OData из запросов пользовательского интерфейса SAP Fiori (или любых других клиентов OData) во время выполнения. Как уже упоминалось, E2E-разработка служб модели программирования ABAP RESTful полностью основана на Eclipse и требует установки Eclipse с установленным подключаемым модулем инструментов разработки ABAP.

*Примечание*

*Ознакомьтесь с главой 4 этой книги и разделом ABAP на веб-сайте инструментов разработки SAP для получения дополнительной информации об установке инструментов разработки ABAP:* [*https://tools.hana.ondemand.com/#abap*](https://tools.hana.ondemand.com/#abap)*.*

#### Поток разработки

На рис. 3.5 показан основной поток разработки при разработке сервисов RESTful OData с использованием модели программирования ABAP RESTful. Пройдемся по потоку, начиная снизу.

#### Моделирование данных и поведение

Обычной отправной точкой приложения модели программирования ABAP RESTful является модель данных, основанная на классических таблицах базы данных, хранящихся в словаре ABAP и созданных в базе данных SAP HANA. Независимо от типа приложения, только для чтения или транзакционного, следующий уровень формируется представлениями CDS. Представления CDS обеспечивают семантическое представление базовых структур таблиц базы данных и их взаимосвязей. Модели данных только для чтения называются запросами (***Queries***) в модели программирования ABAP RESTful, тогда как модели транзакционных данных называются бизнес-объектами (***Business Objects***). Чтобы превратить модель данных CDS в бизнес-объект, модель CDS должна определить композиционную иерархию. В модели программирования ABAP RESTful определение композиционной иерархии было полностью интегрировано в ядро ​​языка ABAP (заменяя использование аннотаций в BOPF). Чтобы добавить фактическое поведение к бизнес-объекту (как операции CRUD, так и пользовательские действия), требуется определение поведения с соответствующей реализацией поведения. Определение поведения (***behavior definition***) — это новый объект разработки ABAP, который определяет, какие действия поддерживает каждый узел бизнес-объекта, используя так называемый язык определения поведения (***BDL***), напоминающий язык определения данных (***DDL***) CDS. Фактическая реализация операций делегируется реализации поведения (***behavior implementation***), которая представляет собой стандартный глобальный класс ABAP (также называемый ***пулом поведения***), который просто служит контейнером и пуст, пока фактическая логика поведения реализуется в ***локальных классах обработчиков***.

#### Предоставление бизнес-услуг

Следуя процессу разработки модели программирования ABAP RESTful, на данный момент мы разработали модель данных CDS, и если в нашем приложении требуется транзакционная обработка данных, мы создадим бизнес-объект с определением поведения и одним или несколькими глобальными классами ABAP (пулы поведения), фактически реализующими поведение (операции CRUD и т. д.).

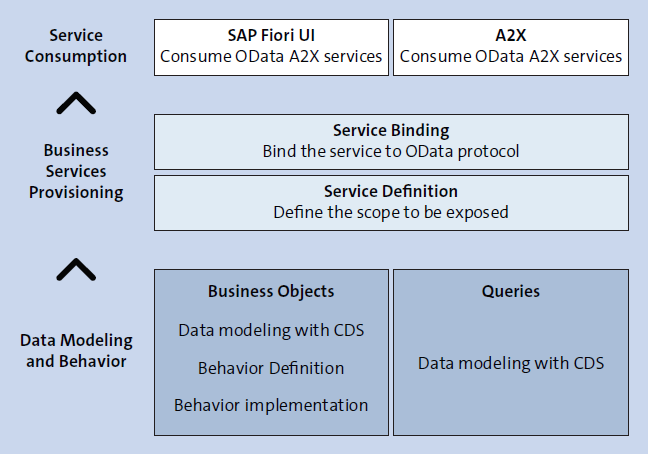


Рисунок 3.5 Модель программирования ABAP RESTful: поток разработки

Пока наша модель данных используется только в стеке ABAP. Чтобы наша модель данных могла использоваться приложениями SAP Fiori или любыми другими клиентами OData, мы должны предоставить ее как службу OData. Раньше вам приходилось выполнять громоздкий процесс, включающий создание проекта построителя сервисов шлюза SAP (транзакция SEGW), сопоставление объектов CDS с объектами OData и развертывание сервиса на шлюзе SAP с использованием транзакции /IWFND/. MAINT\_SERVICE. В модели программирования ABAP RESTful определение того, какие представления CDS должны предоставляться как объекты OData, также было интегрировано в язык ABAP в форме определения службы. В определении службы вы можете декларативным образом, используя синтаксис, подобный CDS, определить, какие сущности CDS будут предоставляться с помощью нашей службы OData без необходимости в дополнительных транзакциях или инструментах SAP GUI. С помощью нового объекта репозитория ABAP, называемого привязкой службы, вы, наконец, можете фактически раскрыть модель приложения, привязав определение службы к определенному протоколу связи клиент-сервер, такому как OData. На этом этапе определения модели данных SADL создает сопоставление сущностей CDS с ***моделью данных сущностей OData*** (***EDM***).

Кроме того, типы данных полей представления CDS сопоставляются с примитивными типами данных EDM; например, встроенный тип DATS словаря ABAP сопоставляется с типом данных ***EDM.DateTime***. Более того, если обработка транзакций включена для сущностей CDS, действия бизнес-объектов будут отображаться как импорт функции OData в документе метаданных службы. В таблице 3.2 представлен обзор преобразований концепций CDS в концепции OData, выполняемых SADL.

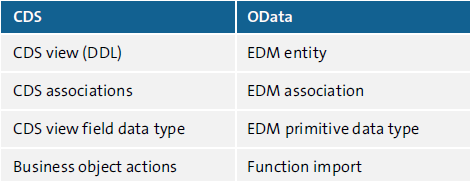


Таблица 3.2. Сопоставление концепций модели данных CDS и концепций OData EDM

Тип привязки службы в настоящее время поддерживает два сценария: *служба пользовательского интерфейса* или *веб-API*. ***Служба пользовательского интерфейса*** — это служба OData, которая позволяет добавлять в службу элементы пользовательского интерфейса SAP Fiori или другие клиенты пользовательского интерфейса, а также включать аннотации пользовательского интерфейса, а ***веб-API*** — это чистая служба OData, используемая для непредвиденных расширений программного обеспечения SAP.

#### Потребление услуг

Теперь служба OData опубликована и может, в зависимости от типа привязки, использоваться либо пользовательскими интерфейсами элементов SAP Fiori (которые будут отображать свой пользовательский интерфейс в соответствии с предоставленными аннотациями пользовательского интерфейса в метаданных службы OData), либо любым типом клиента OData. Например, свободные приложения OpenUI5, которые не используют аннотации пользовательского интерфейса и полностью полагаются на данные и метаданные службы.

#### Время выполнения

Теперь давайте посмотрим, как модель программирования ABAP RESTful обрабатывает клиентские запросы OData во время выполнения. После выполнения всех шагов ранее описанного процесса разработки служба развертывается на шлюзе SAP, запускается и работает. Затем вступает во владение среда выполнения модели программирования ABAP RESTful. На рис. 3.6 показана среда выполнения модели программирования ABAP RESTful и ее различные компоненты. Входящие HTTP-запросы OData сначала проходят через стандартную среду выполнения OData шлюза SAP (provisioning), которая перенаправляет запросы в структуру оркестровки (другими словами, в SADL). SADL позаботится о преобразовании запросов только для чтения (query) в операторы SQL, а также о передаче транзакционных запросов в реализацию поведения бизнес-объекта (CRUD). Бизнес-объекты, реализованные с помощью модели программирования ABAP RESTful, можно использовать не только через протокол OData, но и программно в ABAP с использованием синтаксиса Entity Manipulation Language (EML).

Мы не будем вдаваться в подробности об EML в этой книге, поскольку основное внимание в книге уделяется использованию модели программирования ABAP RESTful для создания сервисов OData, используемых в приложениях SAP Fiori, но вы можете обратиться к справочному порталу SAP для получения более подробной информации.

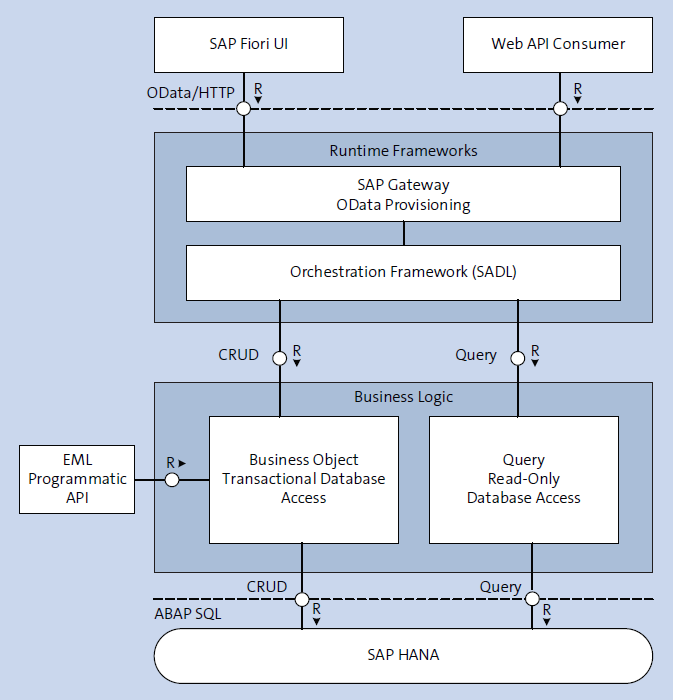


Рисунок 3.6 Компоненты среды выполнения модели программирования ABAP RESTful

### 3.4.2 Запрос (Query)

В модели программирования ABAP RESTful запросы предоставляют доступ только для чтения к моделям данных CDS. Таким образом, эти нетранзакционные аналоги бизнес-объектов основаны на моделях данных CDS без определения и реализации поведения. Запросы служат связующим интерфейсом между запросами службы OData, исходящими из приложений отчетов SAP Fiori или аналитических отчетов, и моделями данных CDS. В общем, мы можем различать управляемые и неуправляемые запросы, которые мы рассмотрим более подробно далее.

#### Управляемые запросы

По умолчанию для запроса используется управляемая реализация. В этом случае платформа запросов (SADL) заботится о преобразовании входящего запроса OData в оператор Open SQL SELECT (или, возможно, также собственный SQL, если требуются определенные собственные функции SAP HANA, такие как текстовый поиск). Среда выполнения OData шлюза SAP делегирует запросы OData общему механизму запросов SADL, который генерирует операторы SQL SELECT для представлений CDS на основе параметров запроса ($select, $top, $filter и т. д.) и запрошенного объекта OData.

Входящий запрос HTTP OData GET для стандартного объекта базового представления заказа на покупку может выглядеть так, как показано в листинге 3.1.

GET I\_PurchaseOrderEntity?$select=

PurchaseOrder,PurchaseOrderType,PurchasingOrganization,PurchasingGroup,Purchas

eOrderDate&$top=10&$orderby=PurchaseOrderDateasc&$filter=

( PurchaseOrderDategedatetime'2018-01-

01T00:00:00'andPurchaseOrderDateledatetime'2018-12-31T00:00:00')

Листинг 3.1. Запрос OData GET для объекта OData I\_PurchaseOrderEntity

Платформа запросов SADL затем прозрачно преобразует этот запрос в инструкцию SQL SELECT в представлении I\_PurchaseOrder CDS, как показано в листинге 3.2.

SELECT

PurchaseOrder,

PurchaseOrderType,

PurchasingOrganization,

PurchasingGroup,

PurchaseOrderDate

FROM

I\_PurchaseOrder

WHERE

PurchaseOrderDate GE '20180601'

AND PurchaseOrderDate LE '20180601'

ORDERBY

PurchaseOrderDate ASCENDING

UP TO 10 ROWS

Листинг 3.2. Сгенерированный SQL SELECT для запроса OData GET

При сравнении запроса OData и сгенерированного оператора SQL обратите внимание на то, как легко запросы OData могут быть переданы в реляционную базу данных. В конце концов, OData уже похож на SQL, и все параметры OData могут быть сопоставлены с определенной частью инструкции SQL SELECT. Это свойство делает OData идеальным кандидатом для предоставления доступа к бизнес-сущностям через HTTP, поскольку простое преобразование HTTP-запросов в операторы SQL SELECT способствует использованию парадигмы SAP HANA Code-to-Data и предотвращает ненужную обработку данных на сервере ABAP. В таблице 3.3 показаны сопоставления между параметрами OData и частями запроса SQL. Эти возможности для преобразования стандартных параметров запроса OData в SQL общедоступны и не требуют явного моделирования.

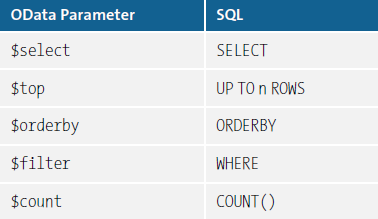


Табл. 3.3 Сопоставление параметров OData и соответствующих частей оператора SQL SELECT

Другие возможности, такие как поиск, агрегирование, справка по значениям или подготовка текста, должны быть явно смоделированы в представлениях CDS с использованием аннотаций и/или ассоциативных отношений.

#### Неуправляемые запросы (Unmanaged Queries)

Если данные должны быть представлены в виде службы OData, которую нельзя прочитать непосредственно из представления CDS и его базовой таблицы базы данных, у вас есть возможность определить и реализовать то, что называется неуправляемым запросом. На рис. 3.7 показана неуправляемая среда выполнения запросов (справа) в отличие от управляемой среды выполнения запросов (слева). Вместо того, чтобы передавать выбор данных через представление CDS в таблицу базы данных, неуправляемый запрос опирается на пользовательский объект CDS, который делегирует выбор данных классу реализации запроса ABAP. Класс реализации запроса ABAP должен реализовать интерфейс ***IF\_A4C\_RAP\_QUERY\_PROVIDER*** и реализовать любой требуемый выбор данных, например, вызвать внешнюю службу OData или какой-либо устаревший код ABAP, который нельзя преобразовать в модель данных CDS.

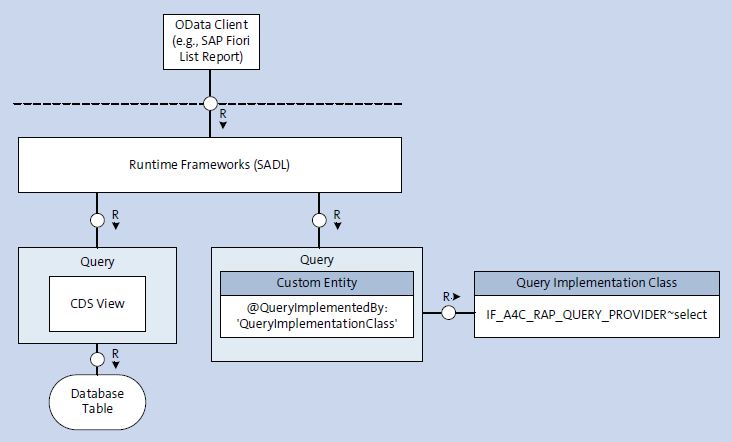
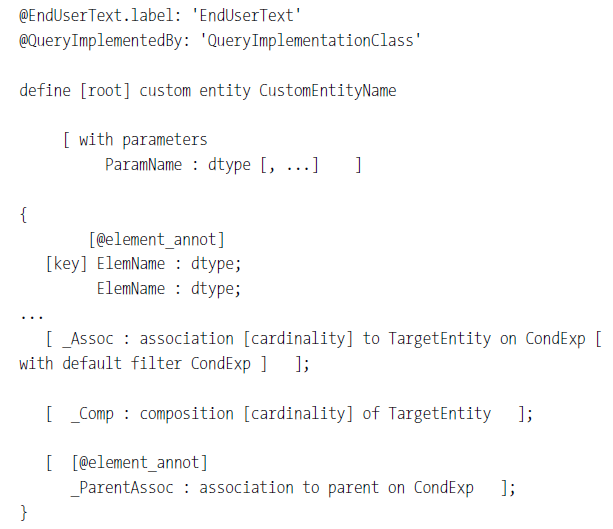


Рис. 3.7 Управляемая и неуправляемая среда выполнения запросов

В листинге 3.3 показан синтаксис для определения пользовательского объекта CDS. Синтаксис должен предоставлять список типизированных элементов, поскольку тип нельзя вывести из базовой сущности CDS или таблицы базы данных. Обратите внимание, что из-за ключевого слова ROOT и ассоциаций иерархии (COMPOSITION, ASSOCIATION TO PARENT) сущность CDS также может использоваться в бизнес-объекте. Однако имейте в виду, что, поскольку для настраиваемых сущностей не создается артефакт SQL, их нельзя использовать как обычные представления CDS в соединениях SQL или операторах SELECT. Кроме того, все возможности, предоставляемые управляемой средой выполнения из коробки, должны быть явно запрограммированы здесь разработчиком, например, $select, $filter и т. д.



Листинг 3.3. Синтаксис для определения пользовательского объекта CDS

### 3.4.3 Бизнес-объекты

Бизнес-объекты — это транзакционные аналоги запросов. Как и запросы, бизнес-объекты основаны на базовой модели данных CDS, но дополняют модель иерархической древовидной структурой, имеющей один корневой узел с дочерними узлами. Кроме того, определение поведения определяет, какие действия поддерживает бизнес-объект, а пул поведения реализует эти конкретные операции в форме одного или нескольких глобальных классов ABAP, имеющих локальные классы-обработчики для изменения и сохранения бизнес-объектов.

#### Управляемые и неуправляемые бизнес-объекты

Модель программирования ABAP RESTful рассматривает несколько сценариев разработки транзакционных приложений, но на данный момент в основном поддерживается неуправляемый сценарий, который также представляет интерес для локальных клиентов, у которых уже есть существующие модели данных и API для своих приложений. В неуправляемом сценарии вы можете использовать существующие API приложений, например классы ABAP и/или функциональные модули, которые записывают данные во внутренние буферные таблицы, или действие сохранения, которое сохраняет эти буферизованные данные в базе данных. Этот сценарий показан на рисунке 3.8.

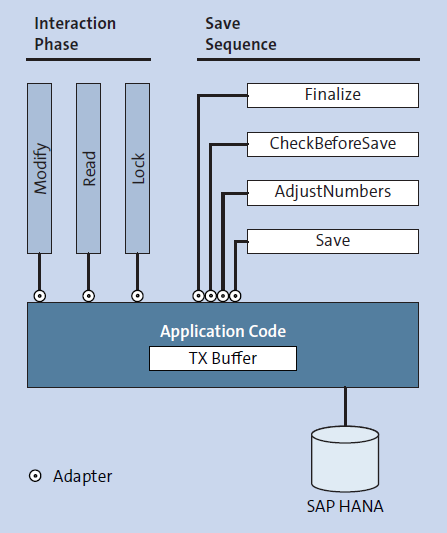


Рисунок 3.8 Модель программирования ABAP RESTful: сценарий неуправляемого приложения

На этапе взаимодействия данные записываются в буферы транзакций (внутренние таблицы ABAP), а в последовательности сохранения данные сохраняются в базе данных. Перед фактическим вызовом метода сохранения вызываются методы Finalize, Check\_before\_save и Adjust\_numbers бизнес-объекта для финализации, проверки и реализации поздней нумерации для бизнес-объектов, если это необходимо. При использовании функции поздней нумерации объекты получают временный ключ только при создании до тех пор, пока в последовательности сохранения не будет вызван метод Adjust\_numbers. Затем метод Adjust\_numbers должен предоставить фактические ключи, которые будут сохранены в базе данных. В управляемом сценарии техническая реализация, например операции CRUD и операция Save, берет на себя инфраструктура бизнес-объектов модели программирования ABAP RESTful.

Разработчики могут добавлять бизнес-логику, используя определения, проверки и действия (что может показаться вам знакомым, если вы разрабатывали приложения с BOPF). В третьем неуправляемом сценарии, представляющем собой гибридный сценарий, основанный на двух предыдущих сценариях, операциями изменения управляет среда выполнения, поскольку в прошлом они были слишком тесно связаны со старой технологией пользовательского интерфейса Web Dynpro (PAI/PBO) и не могут использоваться повторно. Однако доступные функциональные модули задачи обновления можно повторно использовать, в этом случае, в последовательности сохранения для сохранения данных в базе данных. В следующих разделах мы сосредоточимся на неуправляемом сценарии, который наиболее применим для большинства приложений. Он в настоящее время является единственным доступным в SAP S/4HANA 1909.

#### Бизнес-объекты

Как обсуждалось ранее, создание транзакционного бизнес-объекта — это двухэтапный процесс. Сначала вы определите структуру бизнес-объекта, а затем определите и реализуете его поведение в ABAP. Допустим, вы хотите представить свой документ покупки и представления CDS позиции документа покупки как транзакционный бизнес-объект. Вы должны определить документ покупки как корневой узел бизнес-объекта, а товар — как его дочерний узел. Корневой узел имеет особое значение, поскольку он служит представлением бизнес-объекта, обозначенного ключевым словом ROOT. В нашем примере, чтобы определить представление CDS документа покупки в качестве корневого узла бизнес-объекта, необходимо добавить ключевое слово ROOT в определение его представления. Чтобы сделать представление элемента своим дочерним узлом, вы должны добавить специальную ассоциацию COMPOSITION в представление элемента внутри корневого представления документа покупки, как показано в листинге 3.4, и ассоциацию ASSOCIATION TO PARENT, как показано в листинге 3.5, ассоциацию в представлении элемента, указывая на корневой вид документа покупки.

@AbapCatalog.sqlViewName:'CDS\_DB\_VIEW'

@[view\_annotation\_1]

...

@[view\_annotation\_n>]

DEFINE ROOT VIEW RootEntity

[parameter\_list]

AS SELECT FROM data\_source [AS alias]

COMPOSITION [min..max] OF ChildEntity AS \_childEntity

ON condition\_exp

[additional\_composition\_list]

[association\_list]

{

[element\_list]

}

Листинг 3.4. Синтаксис для определения корневой сущности бизнес-объекта

@AbapCatalog.sqlViewName:'CDS\_DB\_VIEW'

[@view\_annotation\_1]

...

[@view\_annotation\_n>]

DEFINE VIEW ChildEntity

[parameter\_list]

AS SELECT FROM data\_source[AS alias]

ASSOCIATION TO PARENT ParentEntity AS \_parentEntity

ON condition\_exp

[additional\_association\_list]

{

[element\_list]

}

Листинг 3.5. Синтаксис для определения дочерней сущности бизнес-объекта

#### Определение поведения

Какие транзакции и бизнес-логика поддерживает бизнес-объект, определяется в одном файле определения поведения. Определение поведения включает характеристику поведения и набор операций для каждой сущности CDS дерева композиции бизнес-объекта.

Определение поведения — это новый переносимый объект репозитория ABAP, который определяется с использованием языка определения поведения (BDL), похожего на CDS DDL. Его синтаксис показан в листинге 3.6.

Определение поведения начинается с определения сценария модели программирования ABAP RESTful. В нашем случае мы сосредоточимся на неуправляемом сценарии, который является наиболее распространенным сценарием.

Затем вы должны определить поведение каждого транзакционного объекта CDS в вашем бизнес-объекте, используя оператор определения поведения для *CDSEntity…{…}* (здесь CDSEntity необходимо заменить фактическим именем представления CDS).

Затем вы должны указать имя глобального класса ABAP, который будет фактически реализовывать указанное поведение вашего узла бизнес-объекта и определить дополнительные возможности, такие как поле, которое будет использоваться в качестве ETag (обычно это последняя измененная метка времени), блокировка или задержка. Нумерация должна быть активной.

Поле ETag включает оптимистичный контроль параллелизма с протоколом HTTP без отслеживания состояния, на котором основан OData. При каждом запросе сервер отправляет значение поля ETag внутри HTTP-заголовка ответа OData, и если клиент хочет записать данные, сначала проверяется значение ETag запрошенного ресурса на наличие изменений (чтобы определить, соответствует ли текущее значение ранее полученному значению). Если это так, клиенту не разрешено обновлять данные, что приведет к перезаписи изменений, сделанных другим клиентом за это время. На эту проблему обычно указывает код ответа HTTP 412 Precondition Failed Header. Свойство главного узла блокировки указывает, что корневой узел бизнес-объекта поддерживает прямую блокировку, тогда как зависящий от блокировки указывает, что состояние блокировки этого узла бизнес-объекта зависит от состояния блокировки его корневого или родительского объекта. Когда активирована поздняя нумерация, вновь созданные экземпляры получают временный ключ $00000001, который необходимо преобразовать в окончательный ключ в последовательности сохранения (в методе Adjust\_numbers) бизнес-объекта, прежде чем экземпляр будет сохранен в базе данных.

Затем между фигурными скобками можно определить элемент управления статическим полем, указать, является ли поле доступным только для чтения или обязательным, указать операции, поддерживаемые узлом бизнес-объекта, и дополнительные действия. Ключевое слово internal указывает, что операцию не следует экспортировать для внешних потребителей. Все ассоциации (композиции), образующие структуру бизнес-объекта, также должны быть объявлены в определении поведения как ассоциации. С помощью необязательного ключевого слова {create;} объекты связанного объекта могут быть созданы через текущий родительский объект.

/\* Header of behavior definition \*/

implementation {unmanaged | managed | abstract};

/\* Definition of entity behavior \*/

define behavior for CDSEntity [aliasAliasName]

/\* Entity properties \*/

[implementation in class ClASS\_NAME unique]

[late numbering]

[etag (field)]

//Only supported for root entities

[lock {master | dependent (

PropertyDependent = PropertyMaster)}]{

/\* Static field control \*/

[field (readonly | mandatory) field1

[,field2, ... ,fieldn];]

/\* Standard operations \*/

[internal] create;

[internal] update;

[internal] delete;

/\* Actions \*/

[static] action ActionName

[parameter {InputParameterEntity | $self)}]

[result [cardinality] {OutputParameterEntity |

$self}];

/\* Associations\*/

association AssociationName

[abbreviation AbbreviationName] {[create;]}

}

Листинг 3.6. Синтаксис для определения транзакционного поведения объекта CDS

*Примечание*

*Определение поведения обычно имеет то же имя, что и корневое представление CDS структуры бизнес-объекта, но с суффиксом \_U для неуправляемого.*

#### Реализация поведения

Как вы видели, внутри определения поведения на класс реализации поведения ссылаются с помощью уникального оператора реализации в классе ClASS\_NAME. Глобальный класс ABAP, на который ссылаются в этом случае, является просто пустым контейнером и в основном пуст, в то время как фактическая логика поведения реализована в нескольких локальных классах. Эти глобальные классы ABAP, указанные в определении поведения, называются пулами поведения. Для одного определения поведения может существовать несколько пулов поведения. Вы можете создать пул поведения для каждого узла бизнес-объекта, что является передовой практикой. В листинге 3.7 показан синтаксис глобального класса пула поведения ABAP. Обратите внимание, что на определение поведения ссылается дополнение FOR BEHAVIOR OF BehaviorDefinition определения глобального класса.

CLASS class\_name DEFINITION PUBLIC

ABSTRACT

FINAL

FOR BEHAVIOR OF BehaviorDefinition.

PUBLIC SECTION.

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS class\_name IMPLEMENTATION.

ENDCLASS.

Листинг 3.7. Синтаксис класса ABAP Global Behavior Pool

Как упоминалось ранее, класс глобального пула поведения пуст, а реальная реализация поведения происходит в локальных классах. На вкладке «Локальные типы» Eclipse должны быть реализованы два так называемых класса-обработчика: один для фазы взаимодействия (изменение операций) и один для последовательности сохранения. Для фазы взаимодействия локальный класс должен быть подклассом ***cl\_abap\_behavior\_handler***, и для последовательности сохранения ***cl\_abap\_behavior\_saver*** локальный класс должен переопределить все свои методы (***finalize, check\_before\_save, adjust\_numbers, save***). Обязательна только реализация метода *save*. Реализация всех остальных методов необязательна. Однако методы *finalize* и *check\_before\_save* могут нарушить последовательность сохранения, если в бизнес-объекте будет обнаружено недопустимое состояние, например, недопустимое значение. Фаза взаимодействия и классы обработчиков последовательности сохранения отражают две части среды выполнения бизнес-объекта, как показано на рис. 3.9. На этапе взаимодействия данные хранятся в транзакционном буфере (одна или несколько внутренних таблиц); в последовательности сохранения данные сохраняются в базе данных. Это разделение также отражает наследие более старых моделей программирования для приложений SAP, например, Web Dynpro. Данные хранились во внутренних буферных таблицах до тех пор, пока не было запущено сохранение, а затем данные сохранялись. Если ваши устаревшие приложения следовали этой парадигме, то модель программирования ABAP RESTful — это просто еще один уровень абстракции поверх ваших существующих API.

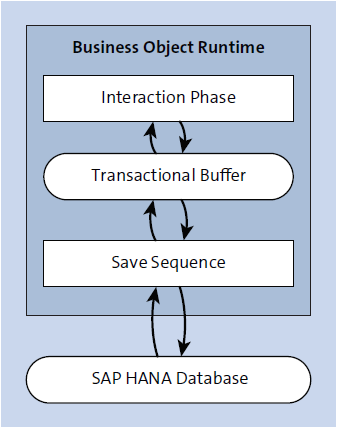


Рис. 3.9 Части среды выполнения бизнес-объекта: фаза взаимодействия и последовательность сохранения

В листинге 3.8 показано определение класса локального обработчика. Этот код содержит методы для модифицирующих операций (CRUD-операций и дополнительных действий); способ обработки блокировки; и метод чтения для чтения сущностей из транзакционных буферов или, если они отсутствуют в буферах, из базы данных через соответствующее представление CDS. Огромная разница между моделью программирования ABAP RESTful по сравнению с BOPF и другими средами ABAP, такими как инфраструктура поставщика услуг (SPI), заключается в том, что параметры методов реализации поведения типизируются текущим объектом CDS. Другие фреймворки полагаются на универсальные параметры, для которых разработчик должен был явно указывать «приведение», если это необходимо. Типизированные параметры в модели программирования ABAP RESTful являются большим преимуществом по сравнению с другими фреймворками и делают программирование в модели программирования ABAP RESTful намного более удобным и безопасным.

CLASS lcl\_handler DEFINITION INHERITING

FROM cl\_abap\_behavior\_handler ABSTRACT FINAL.

PRIVATE SECTION.

METHODS <method\_name> FOR MODIFY

IMPORTING

create\_import\_parameter FOR CREATE entity

update\_import\_parameter FOR UPDATE entity

delete\_import\_parameter FOR DELETE entity

action\_import\_parameter FOR ACTION

entity~action\_name

[RESULT action\_export\_parameter].

METHODS <method\_name> FOR LOCK IMPORTING

lock\_import\_parameter FOR LOCK entity.

METHODS <method\_name> FOR READ IMPORTING

read\_import\_parameter FOR READ entity

[RESULT read\_export\_parameter].

ENDCLASS.

Листинг 3.8. Определение класса локального обработчика

В листинге 3.9 показано определение локального класса сохранения. Методы *finalize* и *check\_before\_save* вызываются первыми и могут прервать последовательность сохранения, установив их параметр FAILED. Метод *adjust\_numbers* вызывается, если в определении поведения активирована поздняя нумерация. Наконец, фактическое сохранение объекта происходит в методе *save*.

CLASS lsc\_saver DEFINITION INHERITING

FROM cl\_abap\_behavior\_saver ABSTRACT FINAL.

PROTECTED SECTION.

METHODS finalize REDEFINITION.

METHODS check\_before\_save REDEFINITION.

METHODS adjust\_numbers REDEFINITION.

METHODS save REDEFINITION.

ENDCLASS.

Листинг 3.9. Определение класса Local Saver

*Примечание*

*Имя пула поведения (глобальный класс ABAP) обычно заканчивается суффиксом* ***\_U*** *для неуправляемого. Класс локального обработчика (lhc) обычно имеет префикс* ***lhc\_****, а класс локального сохранения (lsc) обычно имеет префикс* ***lsc\_****.*

На рис. 3.10 показаны отношения между сущностями бизнес-объектов CDS и определением и реализацией поведения.

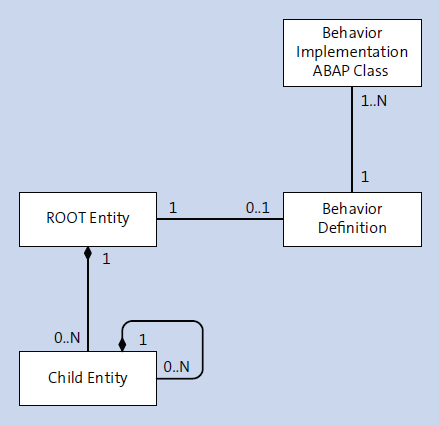


Рис. 3.10 Связь между сущностями бизнес-объектов CDS и определением и реализацией поведения

Может существовать только одна сущность корневого бизнес-объекта, но с несколькими дочерними сущностями, которые, в свою очередь, также могут иметь свои собственные дочерние сущности. Для всего бизнес-объекта или корневой сущности CDS может существовать только одно определение поведения (или ни одного). Реализация определения поведения может быть разделена на несколько классов реализации поведения (по одному для каждого узла бизнес-объекта), но вся реализация также может быть помещена в один класс или пул поведения.

### 3.4.4 Бизнес-услуги

До сих пор все артефакты, которые мы рассматривали, находились исключительно в стеке ABAP и не могли быть использованы каким-либо приложением SAP Fiori. Чтобы модель бизнес-объектов и связанное с ней поведение могли использоваться приложениями SAP Fiori (или любым другим внешним клиентом), она должна быть представлена ​​как служба OData, где вступают в действие новое определение службы и объекты привязки службы.

#### Определение услуги (Service Definition)

Определение службы — это новый переносимый тип объекта репозитория ABAP (R3TR SRVD <SERVICE\_DEFINITION\_NAME>), и, как и определение поведения, его синтаксис напоминает синтаксис CDS DDL, как показано в листинге 3.10. Содержимое определения службы — это, по сути, список сущностей CDS, которые будут предоставляться с помощью службы OData. Сущности, перечисленные в фигурных скобках определения службы, будут прозрачно сопоставлены с сущностями OData во время разработки модели программирования ABAP RESTful. Определение службы еще не зависит от протокола, что означает, что определение не требует определенной версии или протокола OData.

@EndUserText.label: 'text'

@<Annotation\_1>

...

@<Annotation\_n>

DEFINE SERVICE service\_definition\_name

{

EXPOSE cds\_entity\_1 [AS alias\_1];

EXPOSE cds\_entity\_2 [AS alias\_2];

EXPOSE ...

EXPOSE cds\_entity\_m [AS alias\_m];

}

Листинг 3.10. Синтаксис определения службы

*Примечание*

*Определение службы обычно именуется в соответствии с корневым узлом бизнес-объекта с суффиксом* ***\_U*** *для неуправляемого.*

#### Привязка службы

Затем привязка службы привязывает ранее определенное определение службы к конкретному протоколу связи клиента и развертывает службу OData на шлюзе SAP. Как и определение службы, привязка службы — это новый переносимый объект репозитория ABAP: R3TR SRVB <SERVICE\_BINDING\_NAME>. Преимущество разделения определения службы и привязки службы заключается в том, что определение службы можно повторно использовать и привязать к нескольким различным привязкам службы, например, к различным протоколам связи с клиентом. Например, у вас может быть одно определение службы и привязки служб для OData версии 2, еще одно для OData версии 4 и еще одно для пользовательского интерфейса элементов SAP Fiori без фактического изменения базовой реализации. Таким образом, самым важным параметром привязки службы является тип привязки. В настоящее время поддерживаются две основные группы: служба пользовательского интерфейса или веб-API. Кроме того, можно настроить версию OData.

*Примечание*

*Привязка службы обычно называется в соответствии с корневым узлом бизнес-объекта с суффиксом* ***\_U*** *для неуправляемого и дополнительным суффиксом для версии OData, например,* ***\_V2****.*

На рис. 3.11 показана взаимосвязь между моделью данных CDS, определением службы и привязкой службы. Для модели данных CDS, состоящей как минимум из одного представления CDS, может существовать несколько определений службы. Другими словами, вы можете создать множество различных сервисов, каждый из которых предоставляет разные части (сущности) одной и той же модели данных CDS.

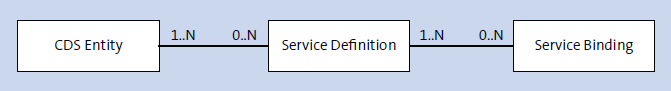


Рисунок 3.11 Взаимосвязь между моделью данных CDS, определением службы и привязкой службы

Определения служб, в свою очередь, также могут предоставляться с использованием различных протоколов связи с клиентами посредством различных привязок служб. Для привязки службы требуется хотя бы одно определение службы. Как упоминалось ранее, вы можете привязать одно и то же определение службы к OData версии 2, к OData версии 4 и к службе пользовательского интерфейса.

### 3.4.5 Потребление услуг

Как упоминалось в предыдущем разделе, служба OData может быть представлена ​​как служба пользовательского интерфейса, которую может использовать приложение SAP Fiori, или как веб-API, который может использоваться любым типом клиента OData.

#### Пользовательский интерфейс SAP Fiori

Если служба OData представлена ​​как пользовательский интерфейс SAP Fiori, дополнительные аннотации пользовательского интерфейса, добавленные в базовую модель данных CDS, будут отображаться как аннотации пользовательского интерфейса OData в метаданных службы. Эти аннотации пользовательского интерфейса, в свою очередь, будут использоваться приложениями элементов SAP Fiori, которые затем настроят свой пользовательский интерфейс в соответствии с аннотациями пользовательского интерфейса. Модель программирования ABAP RESTful также предлагает предварительный просмотр пользовательского интерфейса элементов SAP Fiori. Однако этот процесс не заменяет фактическую разработку пользовательского интерфейса в SAP Web IDE; он просто моделирует соответствующий пользовательский интерфейс элементов SAP Fiori для быстрого тестирования. Вы по-прежнему должны поставить проект пользовательского интерфейса под надлежащий контроль версий в Git и отдельно развернуть его в системе.

#### Веб-API

Если потребитель службы OData не является приложением элементов SAP Fiori, которое будет использовать аннотации пользовательского интерфейса, служба должна быть представлена ​​как веб-API без каких-либо специфических аннотаций пользовательского интерфейса в файле метаданных службы. После этого служба обычно используется, например, приложениями SAP OpenUI5 свободного стиля или любым другим клиентом OData (HTTP). На рис. 3.12 показаны два варианта потребления услуг.

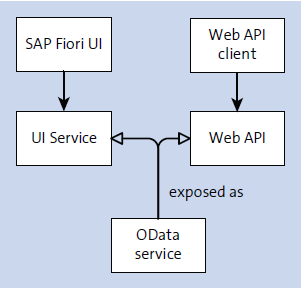


Рис. 3.12 Типы привязки службы пользовательского интерфейса и веб-API и их потребители

### 3.5 Резюме

В этой главе мы представили обзор модели программирования ABAP RESTful и связанных с ней технологий. Мы начали с CDS, которые находятся в центре разработки всех приложений для SAP S/4HANA. CDS являются основой для всех типов приложений и могут быть настроены для различных типов приложений с использованием пользовательских интерфейсов, аналитики или поисковых аннотаций для конкретных доменов. Чтобы превратить модель данных CDS в транзакционный бизнес-объект, требуются новые типы ассоциаций CDS для построения композиционной иерархии. Затем мы рассмотрели SAP Gateway и OData как технологии, обеспечивающие простой сетевой доступ к бизнес-данным, хранящимся в серверных системах SAP NetWeaver. Затем мы обратились к модели программирования ABAP RESTful и изучили, как она поддерживает предоставление основанного на модели доступа для чтения и транзакций к объектам CDS через OData. Модель программирования ABAP RESTful дополняет модель данных CDS новыми переносимыми объектами репозитория ABAP для моделирования поведения (определение поведения и его реализация) и предоставления услуг (определение услуги и привязка услуги), а также представляет собой полностью основанный на Eclipse опыт разработки E2E. Модель программирования ABAP RESTful в настоящее время является современной моделью программирования для разработки приложений для платформы SAP S/4HANA.