In diesem Kapitel lernen Sie Besonderheiten nativer SAP-HANA-Anwendungen und die SAP HANA XS Engine kennen, mit der Sie native Anwendungen erstellen und auf der SAP-HANA-Plattform ausführen lassen können.

2 Struktur einer nativen SAP-HANA-Anwendung

In SAP HANA ist neben der eigentlichen Datenbank unter anderem auch ein Applikationsserver integriert, aus diesem Grund wird SAP HANA nicht als Datenbank, sondern als Plattform bezeichnet. In diesem Kapitel erlernen Sie die wichtigsten Kenntnisse, die Sie für die Erstellung nativer Anwendungen mit der SAP-HANA-Plattform benötigen. Dazu zeigen wir Ihnen, wie Sie eine einfache native Anwendung mithilfe des SAP HANA Studios erstellen. Am Ende dieses Kapitels erklären wir Ihnen noch einige Besonderheiten, die bei Entwicklung nativer Anwendungen mit dem Trial-Account der SAP HANA Cloud Platform (SAP HCP) zu berücksichtigen sind. Sollten Sie Zugriff auf ein On-Premise-System bzw. über einen kostenpflichtigen Account der SAP HCP haben, können Sie natürlich auch diesen Zugang benutzen.

Checkliste

Um die Beispiele aus diesem Kapitel ausprobieren zu können, benötigen Sie:

- Zugang zu einer SAP-HANA-Plattform
- SAP HANA Studio

In Abschnitt 1.5.2, »Benutzerkonto für die SAP HANA Cloud Platform anlegen«, haben wir Ihnen gezeigt, wie Sie sich für die SAP-HANA-Trial-Instanz registrieren, um einen entsprechenden Account für die Entwicklung nativer Anwendungen zu erhalten. Sowohl für das erste einfache Beispiel als auch für die meisten Beispiele in den folgenden Kapiteln ist dieser Entwickler-Account völlig ausreichend.

Des Weiteren haben wir in Abschnitt 1.6, »SAP HANA Studio«, beschrieben, wie die Entwicklungsumgebung für die SAP-HANA-Plattform aufgesetzt wird. Sollten Sie noch nicht über eine funktionierende Entwicklungsumgebung verfügen, sollten Sie diese Schritte nun nachholen.

2.1 Architektur und Programmiermodell der SAP-HANA-Plattform

Die SAP HANA Extended Application Services Engine (XS Engine) bildet die Grundlage für die Entwicklung nativer Anwendungen und besteht aus mehreren Kernkomponenten. Die wichtigsten sind ein voll funktionsfähiger Applikationsserver und ein integrierter Webserver. Neben der SAP HANA XS Engine sind in der SAP-HANA-Plattform noch weitere Komponenten für die Entwicklung von Anwendungen integriert, wie die SAP HANA Web-based Development Workbench und die Werkzeuge zur Administration der XS Engine.

XS Engine

Die Komponenten der XS Engine werden bei der Installation mit installiert, d.h., es wird keine zusätzliche Hardware benötigt. SAP hat sich bewusst dafür entschieden, diese Komponenten in die SAP-HANA-Plattform zu integrieren. Aufgrund dieser Entscheidung kann mit der XS Engine wesentlich performanter auf die Daten innerhalb der SAP-HANA-Datenbank zugegriffen werden, da hierzu keine zusätzlichen Protokolle wie Java Database Connectivity (JDBC) oder Open Database Connectivity (ODBC) benötigt werden.

Mit der XS Engine werden im Wesentlichen zwei Grundprinzipien verfolgt:

- einfache Entwicklung SAP-HANA-basierter Anwendungen sowie deren Einsatz bei gleichzeitiger Minimierung von Architekturschichten
- Entwicklung performanter Anwendungen, die eng mit der SAP-HANA-Datenbank verknüpft sind und somit die Leistungsfähigkeit der SAP-HANA-Datenbank vollständig ausschöpfen

Mit der XS Engine haben Sie die Möglichkeit, webbasierte Anwendungen direkt auf der SAP-HANA-Plattform laufen zu lassen, ohne

einen zusätzlichen Server zu benötigen. Hierdurch kann die Komplexität die Systemlandschaften deutlich reduziert werden.

Programmiermodell

Das Programmiermodell nativer Anwendungen, die mithilfe der XS Engine innerhalb der SAP-HANA-Datenbank ausgeführt werden, lässt sich in die drei folgenden Bereiche unterteilen:

- **Technologien für das grafische User Interface**
  Das grafische User Interface (GUI) wird für native Anwendungen in der Regel an einen Browser oder ein mobiles Endgerät delegiert. Die Erstellung des GUI und die damit verbundene clientseitige Präsentationslogik erfolgt auf Basis von HTML5 und clientseitigem JavaScript. Mit SAPUI5 wird durch die SAP-HANA-Plattform eine HTML5-basierte Frontend-Technologie bereitgestellt. Selbstverständlich können auch andere Clients verwendet werden; jedoch sollte dann die Anbindung an SAP HANA über REST-Schnittstellen bzw. über OData erfolgen.
- **Technologien für die Datenverarbeitung**
  Die wesentlichen Anteile der Applikationslogik, die Geschäftslogik, Berechnungen oder datenintensive Operationen sollten mithilfe von SQL, SQLScript oder der Calculation Engine (CE) implementiert werden. Auf diese Weise wird die vollständige Anwendungsverarbeitung innerhalb der SAP-HANA-Datenbank ausgeführt. Somit wird die Leistungsfähigkeit der SAP-HANA-Plattform vollständig ausgenutzt.
- **Technologie für den Kontrollfluss**
  Der größte Teil einer Anwendung ist mit diesem Programmiermodell bereits erledigt. Es bleiben nur noch Aufgaben im Bereich der Ablauflogik übrig. Diese Aufgaben werden vom Applikationsserver und vom Webserver der XS Engine übernommen. Sie bilden somit das Bindeglied zwischen der Benutzerschnittstelle und der Verarbeitung innerhalb der Datenbank. Zu diesem Zweck stehen die folgenden Technologien zur Verfügung:
  - serverseitiges JavaScript
  - OData (Open Data Protocol)
  - XMLA (eXtensible Markup Language for Analysis)

Abbildung 2.1 zeigt die grundlegende Architektur einer nativen SAP-HANA-Anwendung.

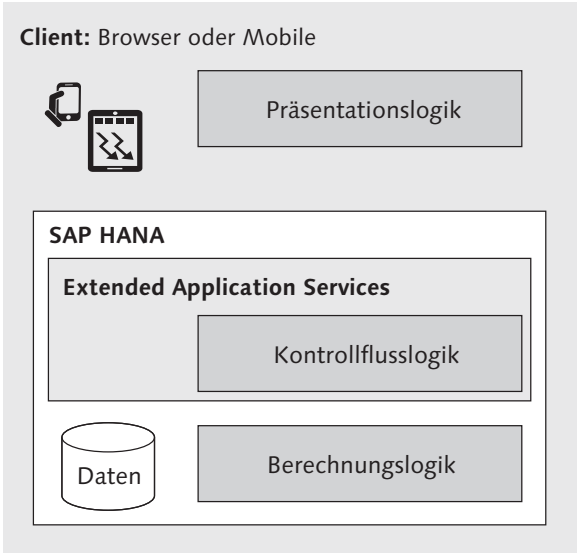

Abbildung 2.1 Architektur einer nativen SAP-HANA-Anwendung

Repository

Zusätzlich beinhaltet die SAP-HANA-Plattform ein eigenes Repository für das Content Lifecycle Management. In diesem Repository werden alle Arten von Entwicklungsobjekten gespeichert. Der Umfang von Anwendungen, die mithilfe der XS Engine erstellt werden können, erstreckt sich von kleinen, leichtgewichtigen Webanwendungen bis hin zu komplexen Unternehmensanwendungen.

2.2 Ihre erste Anwendung erstellen

Nachdem Sie die wichtigsten Grundlagen der XS Engine kennengelernt haben, zeigen wir Ihnen nun, wie Sie eine erste einfache Anwendung erstellen und ausführen. Im Rahmen dieses ersten kleinen Entwicklungsbeispiels erstellen wir mithilfe von serverseitigem JavaScript eine native Anwendung, die den Text »Hello World!« im Webbrowser anzeigt.

SAP HANA Studio

In den folgenden Kapiteln werden wir Ihnen dann zeigen, wie Sie umfangreichere Anwendungen auf Basis der XS Engine erstellen können. Die Entwicklung des ersten Beispiels erläutern wir Ihnen anhand des SAP HANA Studios. Das SAP HANA Studio ist besonders

für den ersten Einstieg in die Programmierung von nativen Anwendungen geeignet, da Wizards und Templates für alle benötigten Objekte bereitgestellt werden.

Web-based Development Workbench

Alternativ können Sie Ihre erste native Anwendung natürlich auch mit der *SAP HANA Web-based Development Workbench* erstellen. Diese Entwicklungsumgebung steht Ihnen sowohl in einer On-Premise- als auch einer Cloud-Umgebung zur Verfügung. In einer On-Premise-Umgebung kann die SAP HANA Web-based Development Workbench über die URL *http://<host:port>/sap/hana/ide* gestartet werden. Um die Web-based Development Workbench in einer Cloud-Umgebung zu starten, finden Sie im SAP HCP Cockpit einen entsprechenden Link unter dem Navigationseintrag HANA XS Applications (siehe Abbildung 2.2).

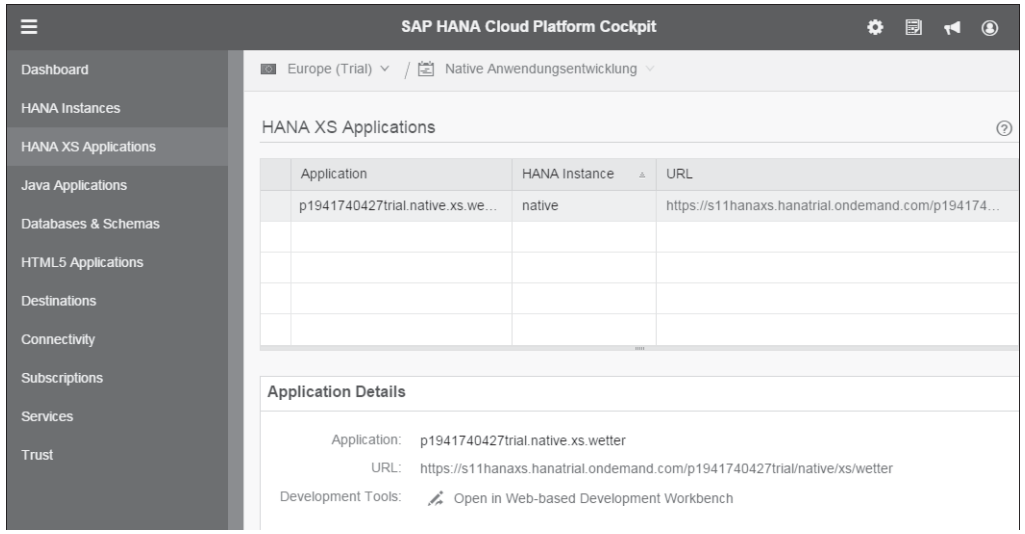

Abbildung 2.2 Start der SAP HANA Web-based Development Workbench

Schritte zur Anwendungsentwicklung

Für die Erstellung unserer ersten Anwendung mit dem SAP HANA Studio müssen Sie die folgenden vier Schritte durchführen:

1. Repository Workspace erstellen
2. ein neues Projekt anlegen und teilen
3. den serverseitigen Code mit JavaScript erstellen
4. Daten darstellen

2.2.1 Repository Workspace erstellen

Bei der Entwicklung einer nativen Anwendung mit dem SAP HANA Studio werden alle erstellten Entwicklungsobjekte im sogenannten *Repository Workspace* abgelegt. Dieser Workspace stellt das Bindeglied zwischen dem lokalen Dateisystem und dem zentralen SAP HANA Repository dar. Mithilfe dieses Repositorys können weitere Entwickler bereits erstellte Entwicklungsobjekte in ihr lokales Dateisystem laden. Auf diese Art und Weise können mehrere Entwickler gemeinsam eine native SAP-HANA-Anwendung programmieren.

Sie erstellen einen neuen Repository Workspace im SAP HANA Studio in der Perspektive SAP HANA Development. Voraussetzung ist hier, dass Sie bereits eine Verbindung zu einem SAP-HANA-System eingerichtet haben.

View »Repositories«

In der Perspektive SAP HANA Development wird der View Repositories bereitgestellt. Wählen Sie dort im Menü File • New • Repository Workspace aus, und geben Sie folgende Informationen ein, um einen neuen Repository Workspace zu erstellen (siehe Abbildung 2.3):

- SAP HANA Systems
  In dieser Auswahlbox wählen Sie das SAP-HANA-System aus, mit dem die Synchronisation des Workspace erfolgen soll.
- Workspace Name
  Der Workspace-Name ist beliebig. Wir haben für unsere erste native SAP-HANA-Anwendung den Namen »DevWS« gewählt. Der von Ihnen gewählte Name des Workspace wird durch den Wizard auch in das Feld Workspace Location integriert.
- Workspace Root
  Der Workspace Root entspricht dem Verzeichnis, in dem die lokalen Objekte des Repositorys gespeichert werden. Dieses Verzeichnis kann an einer beliebigen Stelle auf der Festplatte liegen. In unserem Beispiel verwenden wir anstelle des Standardverzeichnisses das Verzeichnis *C:\SAPHana\DevWS*.

Nachdem Sie alle Informationen eingegeben haben, beenden Sie den Dialog. In der Ansicht Repositories sehen Sie den neu angelegten Workspace (siehe Abbildung 2.4).

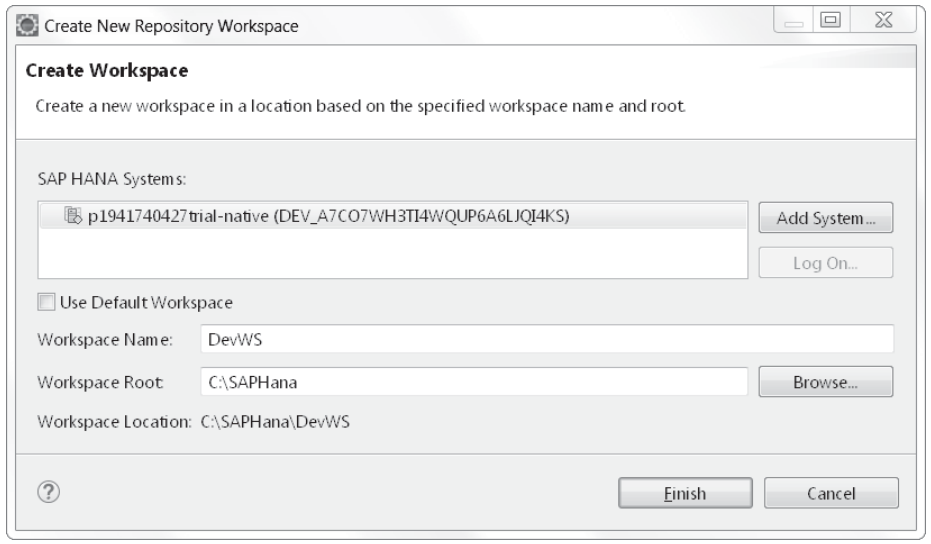

Abbildung 2.3 Neuen Repository Workspace erstellen

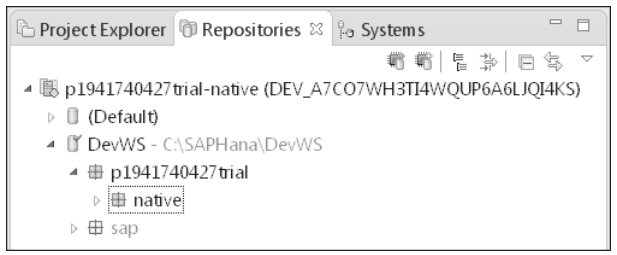

Abbildung 2.4 Neues Repository

2.2.2 Neues XS-Projekt anlegen

Neues XS-Projekt

Ähnlich wie bei Java-Projekten muss für die Entwicklung einer nativen SAP-HANA-Anwendung ein entsprechendes XS-Projekt erstellt werden. Dieses XS-Projekt enthält alle notwendigen Bibliotheken, Ordner und Dateien, die für die weitere Entwicklung notwendig sind.

Um ein neues XS-Projekt anzulegen, wählen Sie im Menü File • New • Projekt. Daraufhin öffnet das SAP HANA Studio den Dialog zur Auswahl des Projekttyps (siehe Abbildung 2.5). Der Wizard für die Anlage eines XS-Projekts befindet sich unter SAP HANA • Application Development. Alternativ können Sie diesen Wizard auch über das

Kontextmenü NEW des Views PROJECT EXPLORER in der Perspektive SAP HANA DEVELOPMENT öffnen.

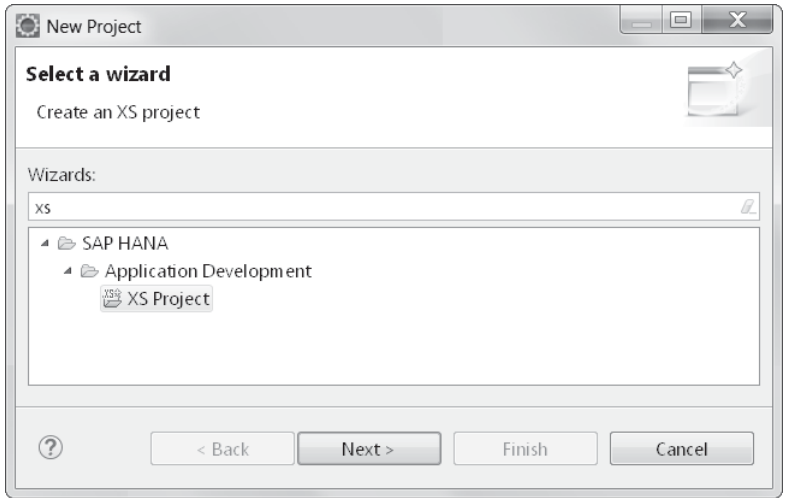

Abbildung 2.5 Auswahl des Projekttyps »XS Project«

Wizard zur Erstellung eines neuen Projekts

Im Wizard zur Anlage eines neuen Projekts erfassen Sie nun folgende Informationen (siehe Abbildung 2.6):

- PROJECT NAME
  In diesem Feld spezifizieren Sie den Namen des Projekts. Da alle Projektnamen im Arbeitsbereich des SAP HANA Studios eindeutig sein müssen, empfehlen wir, bei der Vergabe des Projektnamens den vollständigen Paketnamen zu verwenden.
- LOCATION
  In diesem Feld definieren Sie das lokale Verzeichnis, in dem die Projektdateien gespeichert werden. Hierzu schlägt das SAP HANA Studio bereits ein Verzeichnis vor. Wenn Sie dieses Verzeichnis nicht übernehmen wollen, können Sie es überschreiben. Zu diesem Zweck müssen Sie das Häkchen in der Checkbox SHARE PROJECT IN SAP REPOSITORY entfernen und das Projekt zu einem späteren Zeitpunkt dem Repository hinzufügen.
- WORKING SETS
  Mit dieser Option können Sie das Projekt optional einem bestehenden oder neuen Working Set hinzufügen. Mit einem Working Set können Sie ähnliche Projekte zur Anzeige oder zur Durchführung von Operationen gruppieren. Eine solche Aktion kann z.B.

die Suche nach bestimmten Inhalten von Dateien aus einem Working Set sein.

Für unser erstes Projekt sollten Sie wie in Abbildung 2.6 »xs.example.hello« als Projektnamen eingeben.

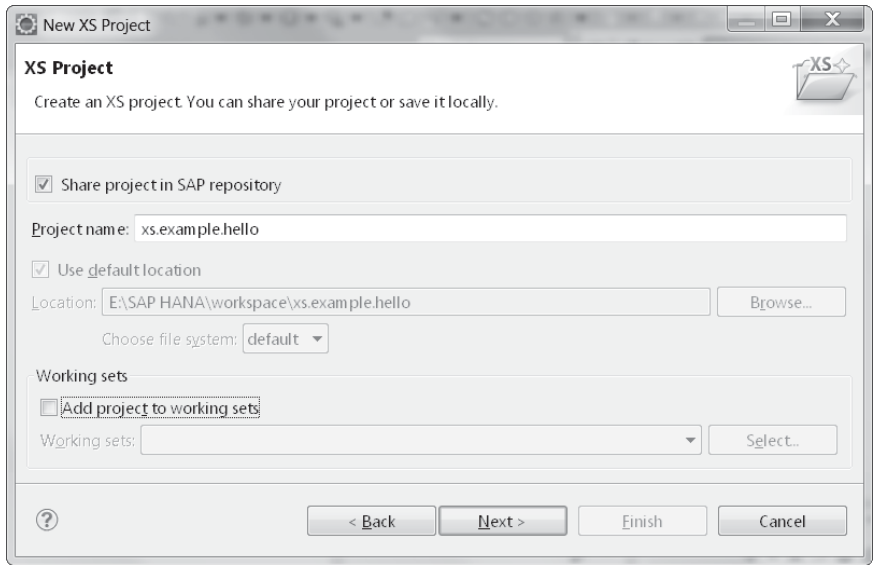

Abbildung 2.6 Neues XS-Projekt anlegen

Nachdem Sie den Wizard mit FINISH geschlossen haben, sollte das neue Projekt, wie in Abbildung 2.7 zu sehen, im View PROJECT EXPLORER sichtbar sein.

Abbildung 2.7 Darstellung des neuen Projekts

Anwendungsprojekt teilen

Durch das Teilen des Projekts werden alle Inhalte des neuen Projekts für weitere Mitglieder des Entwicklungsteams sichtbar, sofern diese ebenfalls eine Verbindung zu demselben Repository eingerichtet haben. Auf diese Art und Weise lassen sich verschiedene Versionen der Entwicklungsobjekte verwalten und zwischen den Mitgliedern des Entwicklungsteams synchronisieren.

Wenn Sie bei der Anlage des Projekts das Häkchen bei der Option SHARE PROJECT IN SAP REPOSITORY gesetzt haben, wird das Projekt bei der initialen Erstellung automatisch dem SAP HANA Repository hinzugefügt. Andernfalls müssen Sie das Projekt manuell hinzufügen.

Teilen eines Projekts

Um das Projekt manuell dem Repository hinzuzufügen, wechseln Sie auf den View PROJECT und wählen das zu teilende Projekt aus. Im Kontextmenü wählen Sie TEAM • SHARE PROJEKT, um den Wizard zum Teilen des Projekts zu starten. Als Repository-Typ wählen Sie SAP HANA RESPOSITORY aus (siehe Abbildung 2.8).

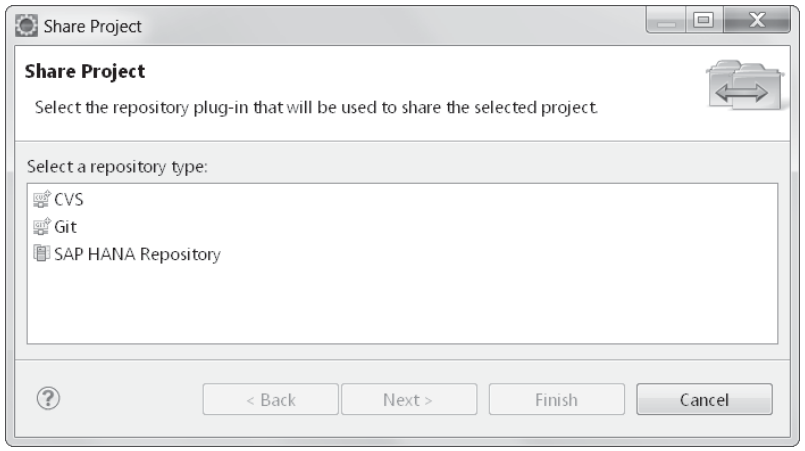

Abbildung 2.8 Auswahl des SAP HANA Repositorys

Auswahl des Repository Workspace

Über den Butten NEXT gelangen Sie zur nächsten Seite des Wizards. Auf dieser Seite müssen Sie nun den Repository Workspace auswählen, in dem das Projekt gespeichert werden soll. Wenn Sie nur einen Repository Workspace erstellt haben, wird dieser im Wizard automatisch vorausgewählt. Sollten Sie jedoch mehrere Repository Workspaces erstellt haben, müssen Sie einen bestehenden Repository Workspace auswählen, in dem das Projekt hinzugefügt werden soll.

Über die Option ADD PROJECT FOLDER AS SUBPACKAGE können Sie definieren, dass der Name des Projekts als Name für das Paket im Repository angelegt wird. Möchten Sie den Namen des Pakets im Repository ändern, entfernen Sie das Häkchen in dieser Checkbox und geben den Namen für das Paket in das Textfeld REPOSITORY PACKAGE ein (siehe Abbildung 2.9).

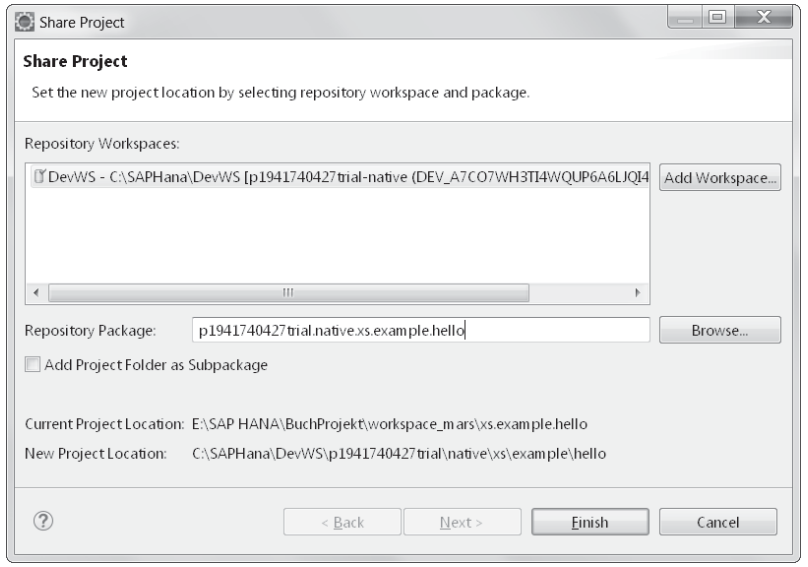

Abbildung 2.9 Projekt teilen

Ablage des Projekts mit dem SAP-HANA-Trial-Account [«]

Wenn Sie – wie vorgeschlagen – für Ihr neues Projekt den SAP-HANA-Trial-Account verwenden, müssen Sie darauf achten, dass das neue Projekt unter Ihrem Account im Repository *<Account>.<SAP-HANA-Instanz>* angelegt wird. In unserem Fall lautet der Name des Accounts `p1941740427trial`. Der Name der SAP-HANA-Instanz lautet in unserem Fall `native`.

Commit der Projektdateien

Nachdem das Projekt dem Workspace Repository hinzugefügt wurde, müssen Sie nun die einzelnen Dateien dem Repository hinzufügen. Das Hinzufügen der Dateien verläuft ähnlich wie der Check-in von Dateien in ein CSV-, SVN- oder Git-Repository. Um die Dateien dem Workspace Repository hinzuzufügen, wählen Sie die entsprechende Datei aus und öffnen das Kontextmenü über die rechte Maustaste. Im Bereich TEAM wählen Sie die Operation COMMIT aus, um die Datei dem Repository hinzuzufügen.

2.2.3 Serverseitiges JavaScript

Wir haben nun ein Projekt aufgesetzt und können mit der Programmierung einer ersten kleinen Anwendung beginnen. Wie Sie es aus

anderen Programmierhandbüchern kennen, verwenden wir auch hier das beliebte Beispiel »Hello World!«.

Konzept der ersten Anwendung

Diese erste Beispielanwendung programmieren wir mithilfe der XS Engine und serverseitigem JavaScript. Unsere JavaScript-Datei wird innerhalb der XS Engine ausgeführt und soll eine HTML-Seite erzeugen, auf der der Text »Hello World!« ausgegeben wird.

Hierfür müssen wir die folgenden Dateien erstellen:

- *MyFirstSapHanaApp.xsjs*
  Diese Datei enthält den serverseitigen JavaScript-Code.
- *.xsapp*
  Diese Datei markiert den Startpunkt der Pakethierarchie für die Anwendung, auf deren Inhalt über das HTTP-Protokoll zugegriffen werden kann.
- *.xsaccess*
  Mithilfe dieser Datei können Sie die Zugriffsrechte auf die jeweiligen Anwendungspakete für jede XS-Anwendung festlegen, die von Ihnen entwickelt und bereitgestellt wird.

Datei MyFirstSapHanaApp.xsjs anlegen

Neue Datei anlegen

Um eine neue Datei anzulegen, wählen Sie im Menü FILE • NEW. Daraufhin öffnet das SAP HANA Studio den Dialog zur Auswahl des entsprechenden Wizards. Für die Erstellung einer XS-JavaScript-Datei stellt das SAP HANA Studio einen Wizard bereit, der unter dem Gliederungspunkt SAP HANA • APPLICATION DEVELOPMENT zu finden ist. In Abbildung 2.10 sind die verfügbaren Wizards zur Erstellung von nativen Anwendungen dargestellt.

[+] Datei automatisch dem Repository hinzufügen

Wenn Sie das SAP HANA Studio verwenden, um neue Dateien zu erstellen, wird die neue Datei durch den Wizard automatisch dem SAP HANA Repository hinzugefügt und automatisch zum Bearbeiten geöffnet.

XS-JavaScript-Datei erstellen

Im nächsten Schritt müssen Sie die Namen der zu erstellenden XS-JavaScript-Datei und das Hauptverzeichnis angeben, in dem die neue Datei erstellt werden soll. In unserem Beispiel ist das richtige Hauptverzeichnis bereits ausgewählt, und Sie müssen nur noch den Namen der XS-JavaScript-Datei festlegen.

Wie bereits erwähnt, soll der Name für unsere XS-JavaScript-Datei *MyFirstSapHanaApp.xsjs* lauten. Geben Sie ihn in das Feld FILE NAME ein, wie in Abbildung 2.11 zu sehen.

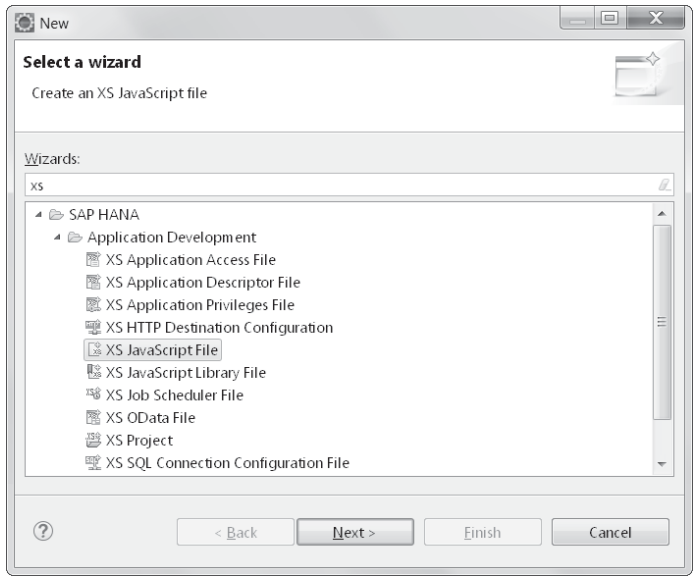

Abbildung 2.10 Auswahl des Wizards zur Erstellung einer neuen XS-JavaScript-Datei

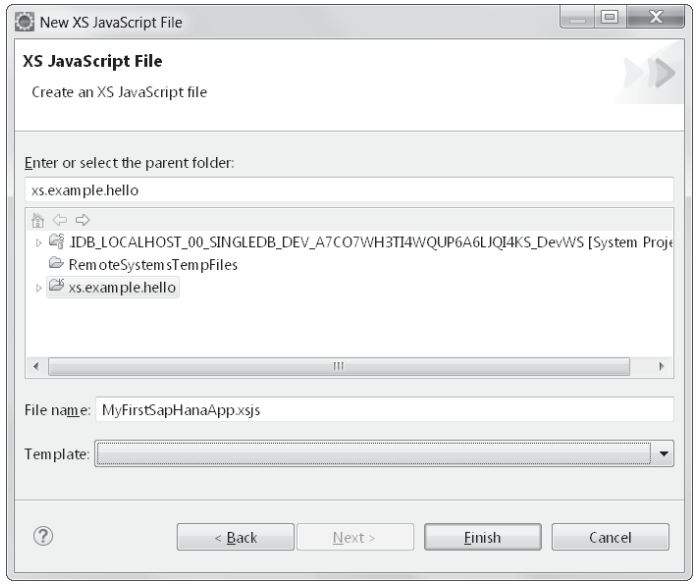

Abbildung 2.11 XS-JavaScript-Datei erstellen

Template auswählen

Bei der Erstellung einer XS-JavaScript-Datei können Sie bereits ein Template auswählen, um ein entsprechendes Grundgerüst für die Datei zu erzeugen. Für unser Beispiel ist dies nicht notwendig.

[+]

Template »Basic« verwenden

Für ein initiale Skriptstruktur stellt das SAP HANA Studio das Template Basic bereit. Mithilfe dieses Templates kann die Struktur für eine einfache Request-Response-Behandlung erstellt werden.

Der erste native SAP-HANA-Code

Nachdem der Wizard die Datei erzeugt hat, öffnet das SAP HANA Studio die Datei *MyFirstSapHanaApp.xsjs* im Editor. In dieser Datei geben Sie nun die folgenden zwei Zeilen ein:

```
$.response.contentType = "text/html";
$.response.setBody( "Hello World!");
```

Dies ist der vollständige Inhalt Ihrer ersten nativen XS-Anwendung. Um die HTML-Seite zu erstellen, verwenden wir die JavaScript-API `response`. Um diese JavaScript-API verwenden zu können, muss das `$`-Zeichen vorangestellt werden.

[»]

Speichern der Datei

Beim Speichern der Datei werden die Änderungen automatisch ins SAP HANA Repository übertragen.

XS-JavaScript-Datei aktivieren

Um unsere erste native SAP-HANA-Anwendung ausführen zu können, müssen Sie die Datei im Repository der SAP-HANA-Plattform aktivieren. Durch die Aktivierung kennzeichnen Sie, dass der Inhalt fertig bearbeitet ist und die Datei für die Ausführung verwendet werden kann.

[»]

Automatisches Aktivieren beim Speichern

Wenn Sie mit der SAP HANA Web-based Development Workbench arbeiten, wird die Datei beim Speichern automatisch im Repository der SAP-HANA-Datenbank aktiviert, sofern die Datei keinerlei Compile-Fehler enthält.

Zum Aktivieren der Datei wählen Sie sie aus, öffnen das Kontextmenü mit der rechten Maustaste und wählen den Eintrag Team • Activate. Alternativ können Sie auch die Tastenkombination Strg + F3 verwenden.

Nach dem erfolgreichen Aktivieren der Datei wechselt das Symbol vor dem Namen der Datei im Project Explorer. Anstelle der grauen Raute wird nun ein gelbes Datenbanksymbol angezeigt (siehe Abbildung 2.12).

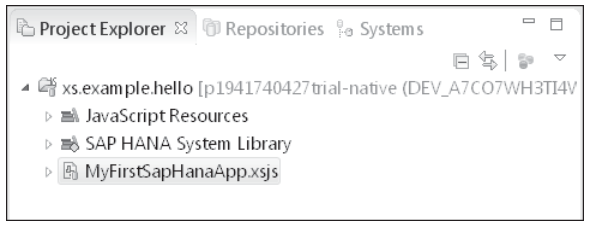

Abbildung 2.12 Darstellung aktivierter Dateien

Application Descriptors erstellen

Application Descriptors

Jede native SAP-HANA-Anwendung, die mithilfe der XS Engine ausgeführt werden soll, benötigt zwei Application Descriptors. Diese Application Descriptors müssen im Root-Paket der Anwendung vorhanden sein. Die Dateinamen der beiden Application Descriptors lauten:

- *.xsapp*
- *.xsaccess*

Die *.xsapp*-Datei ist eine leere Datei und liegt im Root-Verzeichnis der Anwendung. Diese Datei markiert den Startpunkt der Anwendung, von dem aus die Inhalte an den Client übertragen werden.

Die *.xsaccess*-Datei ist für die Sicherheit der nativen SAP-HANA-Anwendung notwendig. In dieser Datei wird geregelt, wer mit welchen Rechten auf die native SAP-HANA-Anwendung zugreifen darf.

Application Descriptors automatisch anlegen [+]

Wenn Sie im Wizard zum Anlegen eines neuen Projekts schon das zuständige Repository auswählen, können Sie auf der dritten Seite des Wizards auswählen, ob diese beiden Application Descriptors gleich mit angelegt werden sollen.

Sollten diese beiden Datei noch nicht in Ihrem Workspace vorhanden sein, müssen Sie diese vor dem ersten Ausführen der nativen Anwendung erstellen und aktivieren. Bei der Erstellung der Applica-

tion Descriptors unterstützt Sie das SAP HANA Studio durch die Wizards XS Application Access File und XS Application Descriptor File.

[»] **Fehlender Application Descriptor**

Fehlen ein oder beide Application Descriptors, wird beim Ausführen der Anwendung im Browser der Fehler 404 angezeigt.

.xsapp-Datei anlegen

Markieren Sie im SAP HANA Studio Ihr Projekt. Über New • Other im Kontextmenü oder mit der Tastenkombination Strg + N öffnen Sie den Dialog zur Auswahl des Wizards. Um die *.xsapp*-Datei zu erstellen, müssen Sie den Wizard XS Application Descriptor File auswählen. Im zweiten Schritt müssen Sie lediglich das Root-Verzeichnis auswählen, in dem der Application Descriptor gespeichert werden soll (siehe Abbildung 2.13). Da diese Datei keinen weiteren Inhalt enthält, müssen Sie sie nach der Erstellung nicht weiter bearbeiten.

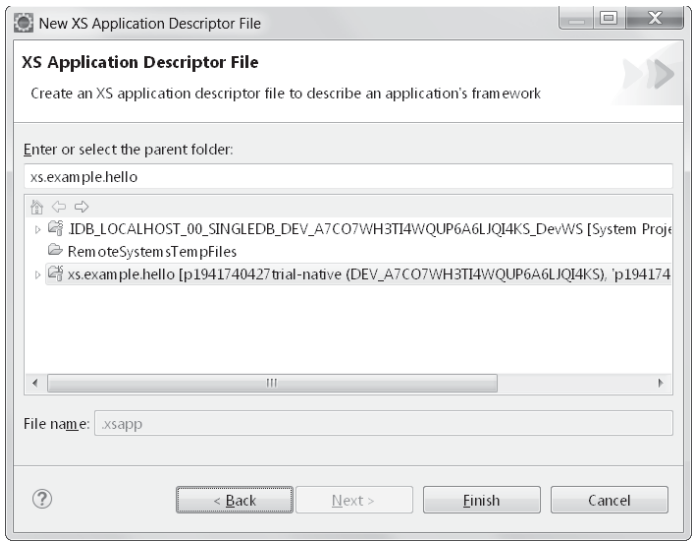

Abbildung 2.13 Wizard zum Erzeugen der .xsapp-Datei

.xsaccess-Datei anlegen

Um die *.xsaccess*-Datei zu erstellen, verfahren Sie ähnlich. Hierbei werden Sie durch den Wizard XS Application Access File unterstützt. Wählen Sie hier ebenfalls das Root-Verzeichnis aus, in dem diese Datei gespeichert werden soll. Das SAP HANA Studio stellt das

Template Basic bereit, über das Sie die Datei initial füllen können (siehe Abbildung 2.14).

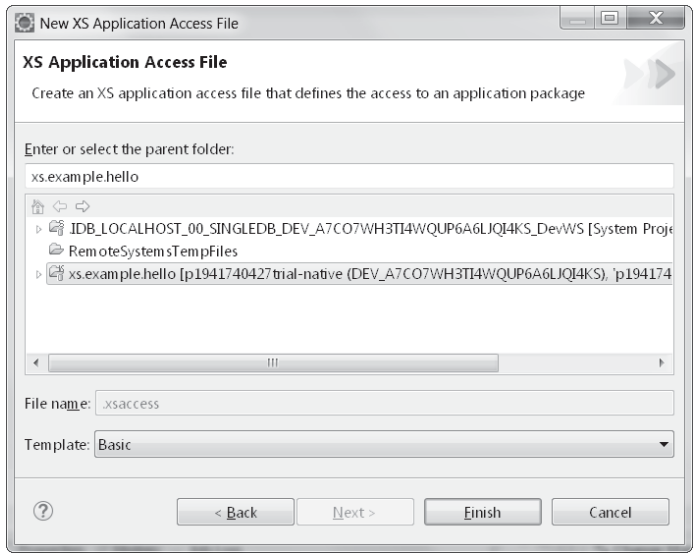

Abbildung 2.14 Wizard zum Erzeugen der .xsaccess-Datei

Für unser erstes kleines Beispiel müssen Sie an dieser Datei keine Änderungen vornehmen. In Listing 2.1 ist der initiale Inhalt des Application Descriptors dargestellt, der auf Basis des Templates Basic erstellt wird.

```
{
     "exposed" : true,
     "authentication" :
            {
               "method": "Form"
            },
     "cache_control" : "must-revalidate",
     "cors" :
            {
             "enabled" : false
            },

     "enable_etags" : false,
     "force_ssl" : false,
     "prevent_xsrf" : true
}
```

Listing 2.1 XS-Application-Access-Datei auf Basis des Templates

Wie auch die XS-JavaScript-Datei müssen die beiden Application Descriptors im Repository aktiviert werden.

SAP-HANA-Anwendung ausführen

Start der Anwendung

Nach dem Aktivieren der Dateien sind alle Voraussetzungen erfüllt, um unsere erste native SAP-HANA-Anwendung zu starten. Wählen Sie dazu im Project Explorer die Datei *MyFirstSapHanaApp.xsjs*. Im Kontextmenü finden Sie unter Run As den Eintrag XS Engine. Klicken Sie darauf, um die Anwendung zu starten. Es öffnet sich ein Browser, und der Text »Hello World!« erscheint innerhalb einer HTML-Seite (siehe Abbildung 2.15).

Abbildung 2.15 Ergebnis der Ausführung der ersten nativen Anwendung

2.3 Anwendungssicherheit

Im Rahmen der Entwicklung einer Anwendung müssen Sie sich Gedanken über die Berechtigungen der Anwender machen. Mit dem Berechtigungskonzept der XS Engine haben Sie die Möglichkeit, den Zugriff auf die jeweiligen Anwendungspakete, die von Ihnen entwickelt und bereitgestellt werden, zu definieren. Die von Ihnen festgelegten Berechtigungen werden in der *.xsaccess*-Datei definiert.

Berechtigungen auf Paketebene

Diese Datei hat keinen Dateinamen, sondern besteht nur aus der Dateiendung *.xsaccess*. Der Inhalt dieser Datei wird im JSON-Format definiert. Die Berechtigungen, die Sie innerhalb dieser Datei festlegen, wirken sich auf das aktuelle Paket sowie auf alle untergeordneten Pakete in der Pakethierarchie aus. Wollen Sie innerhalb Ihrer Anwendung unterschiedliche Berechtigungskonzepte realisieren, müssen Sie für die entsprechenden Unterpakete jeweils eine eigene *.xsaccess*-Datei erstellen.

Mehrere .xsaccess-Dateien [«]

Die Definitionen der Berechtigungen in einem Unterverzeichnis überschreiben die Einstellung der Berechtigungen der übergeordneten *.xsaccess*-Datei. Diese Definitionen werden dann an die Unterpakete in der Pakethierarchie vererbt. Alle Definitionen, die nicht von einer untergeordneten *.xsaccess*-Datei überschrieben werden, werden von der übergeordneten *.xsaccess*-Datei übernommen.

Innerhalb dieser Datei können Sie festlegen, wer über die entsprechenden Berechtigungen für die Komponenten verfügt, um auf den Inhalt der Pakete zuzugreifen, die durch die XS Engine bereitgestellt werden. Des Weiteren können Sie über diese Datei festlegen, wie auf den Inhalt zugegriffen werden kann. So können Sie z. B. festlegen, ob sich ein Benutzer der nativen Anwendung authentifizieren muss, welche Daten gelesen oder welche Daten verändert werden können.

Konfiguration der .xsaccess-Datei

Im Rahmen des Berechtigungskonzepts können folgende Konfigurationen vorgenommen werden:

- **Zugriffsberechtigungen**
  Für die Spezifikation der Authentifizierungsverfahren verwenden Sie in der Datei *.xsaccess* das Schlüsselwort `authentication`. Von der XS Engine ist sowohl eine formularbasierte als auch eine einfache Authentifizierung möglich.
- **Anonyme SQL-Verbindungen**
  Mithilfe des Schlüsselwortes `anonymous_connection` wird eine Datei mit der Endung *.xssqlcc* referenziert, die für den Zugriff auf die Datenbank notwendig ist, sofern der angemeldete Benutzer nicht die Berechtigung zum Zugriff auf die Datenbank besitzt.
- **Voreingestellte SQL-Verbindungen**
  Mit dem Schlüsselwort `default_connection` wird ebenfalls eine Datei mit der Endung *.xssqlcc* referenziert. Ist dieses Schlüsselwort gesetzt, werden alle Zugriffe auf die Datenbank mit den in der *.xssqlcc*-Datei spezifizierten Datenbankbenutzern durchgeführt.
- **Berechtigungen**
  Durch die Verwendung des Schlüsselwortes `authorization` kann der Zugriff eines Benutzers auf bestimmte Pakete der nativen SAP-HANA-Anwendung begrenzt werden.

- **Cache Control**
  Mit dem Schlüsselwort `cache_control` ist es möglich, den *Cache-Control-Header* für statische Webinhalte, die durch den XS-Webserver bereitgestellt werden, zu überschreiben. Mit den sogenannten Cache-Control-Richtlinien (z. B. `public`, `private`, `no-store`) können Sie das Verhalten der Browser und den Proxy-Cache steuern.
- **Verbindungssicherheit**
  Wird der Parameter `force_ssl` im Application Descriptor der nativen Anwendung auf `true` gesetzt, erzwingen Sie damit, dass der Zugriff auf die native Anwendung nur mittels SSL/HTTPS erfolgen kann. Wird ein Request nicht über SSL/HTTPS durchgeführt, wird dieser mit dem HTTP-Fehlercode 403 zurückgewiesen.
- **Cross-Origin Requests**
  Um HTTP-Requests auf Webseiten einer anderen Domäne zu erlauben, wird der Konfigurationsparameter `cors` verwendet. Mit `{"enabled":true}` wird das *Cross-Origin Resource Sharing* (CORS) aktiviert, so dass HTTP-Requests an eine weitere Domäne erlaubt sind. Innerhalb des Konfigurationsparameters `cors` können noch weitere Einstellungen vorgenommen werden. So können Sie die Aufrufe externer Webseiten durch weitere Parameter einschränken.
- **Cross-Site Request Forgery (CSRF/XSRF)**
  Um die native Anwendung vor Cross-Site Request Forgery zu schützen, können Sie in der Datei das Schlüsselwort `prevent_xsrf` verwenden. Bei Cross-Site Request Forgery (meist mit CSRF oder XSRF abgekürzt) versucht ein Angreifer den Request so zu manipulieren, dass im Hintergrund die gewünschte Aktion ausgeführt wird. Weitere Informationen zur Sicherheit in nativen SAP-HANA-Anwendungen finden Sie in Abschnitt 5.10, »Sicherheit von nativen Anwendungen«.
- **Standard-Index-Datei**
  Wird auf das Paket einer nativen Anwendung ohne Angabe einer Datei in der URI zugegriffen, versucht die XS Engine immer, die Datei *index.html* zu laden. Soll statt dieser Datei eine andere HTML-Datei geladen werden, definieren Sie die zu verwendende HTML-Datei über das Schlüsselwort `default_file`.

- **Datenfreigabe**
  Mit dem Schlüsselwort `exposed` wird definiert, welche Inhalte einer nativen Anwendung freigegeben werden. Um die Anwendung optimal zu schützen, sollten Sie nur Inhalte freigeben, die für die Anwendung zwingend benötigt werden.
- **Entity Tags**
  Mit dem Schlüsselwort `enabled_etags` können Sie die Bildung von sogenannten *Entity Tags* (ETags) für statische Webinhalte gestatten oder verhindern. Mithilfe von Entity Tags kann die Performance verbessert werden, da gleiche Daten vom Server nicht erneut gesendet werden müssen, sofern diese nicht verändert wurden.
- **MIME-Mapping**
  Für das Mapping verschiedener Dateiendungen auf einen MIME-Typ kann das Schlüsselwort `mime_mapping` verwendet werden. So werden z. B. mit der folgenden Anweisung alle Dateien mit der Endung *.jpg* auf den MIME-Typ `image/jpeg` gemappt.

  ```
  "mime_mapping": [ {"extension":"jpg",
  "mimetype":"image/jpeg"} ]
  ```

- **URL-Rewrite-Regeln**
  Mit der Konfiguration von URL-Überschreibungsregeln (*Rewrite-Regeln*) können Details über interne URLs vor externen Benutzern, Clients und Suchmaschinen verborgen werden. Hierzu ist in der Datei *.xsaccess* eine entsprechende Konfiguration notwendig, die mit dem Schlüsselwort `rewrite_rules` eingeleitet wird.
- **X Frame Options**
  Mit dem Konfigurationsparameter `headers` in der Datei *.xsaccess* kann die Unterstützung für X-Frame-Options aktiviert werden. Mit dieser Option kann der Server im Header-Response mitteilen, ob der übermittelte Inhalt innerhalb eines Frames dargestellt werden darf.

Weitere Informationen zur der Anwendungssicherheit können Sie dem »SAP HANA Developer Guide« und dem »SAP HANA Security Guide« entnehmen, den Sie unter der URL *http://help.sap.com/hana/sap_hana_developer_guide_en.pdf* bzw. *http://help.sap.com/hana/SAP_HANA_Security_Guide_en.pdf* finden.

2.4 Verwaltung von Anwendungsobjekten

Alle Bausteine, die zu einer nativen SAP-HANA-Anwendung gehören, werden als *Entwicklungsobjekte* bezeichnet und haben jeweils eine eigene Dateierweiterung. So steht z.B. die Dateierweiterung *.hdbtable* für *Designtime Table Definition*, die Erweiterung *.hdbview* für *Designtime SQL-View Definition* und die Erweiterung *.hdbrole* für *Designtime Role Definition*.

Entwicklungsobjekte einer nativen Anwendung

Wenn Sie eine neue native SAP-HANA-Anwendung erstellen, werden Sie die unterschiedlichsten Entwicklungsobjekte verwenden. Für die Strukturierung einer Anwendung bietet sich die Aufteilung in verschiedene Projekte und Pakete an. Weitere Entwicklungsobjekte wie Schemata, die Definitionen von Tabellen, Views oder Analytical bzw. Calculation Views helfen Ihnen als Entwickler, die Daten zu strukturieren.

Prozeduren und serverseitiges JavaScript bilden die wesentlichen Entwicklungsobjekte einer nativen SAP-HANA-Anwendung zur Entwicklungszeit (Designtime). Diese beiden Entwicklungsobjekte haben jeweils eine fest definierte Dateierweiterung: *.hdbprocedure* bzw. *.xsjs*.

Dateierweiterung

Die Verwendung der richtigen Dateierweiterung für die Entwicklungsobjekte spielt bei der Entwicklung einer nativen SAP-HANA-Anwendung eine bedeutende Rolle. Wie wir bereits erläutert haben, müssen die Entwicklungsobjekte im Repository aktiviert werden. Anhand der Dateierweiterung wird entschieden, welches interne Plug-in für die Aktivierung einer Datei im Repository aufgerufen werden soll. Im Rahmen der Aktivierung eines Entwicklungsobjekts wird dessen Inhalt vom entsprechenden Plug-in gelesen, interpretiert und die entsprechenden Laufzeitobjekte werden erzeugt.

Die Dateierweiterungen werden aber auch noch in weiteren Kontexten benutzt. So wird z.B. im SAP HANA Studio anhand der Dateierweiterung ein kontextsensitives Menü für jedes Objekt eingeblendet bzw. die entsprechenden Icons angezeigt. In Tabelle 2.1 sind alle Objekte aufgelistet, die im Rahmen der Entwicklung einer nativen SAP-HANA-Anwendung verwendet werden können.

Dateierweiterung	Entwicklungsobjekt	Beschreibung
.aflpmml	Prozedur	Dieser Dateityp wird vom *Application Function Modeler* verwendet, um Details zu Funktionen zu speichern, die Funktionen aus der Predictive Analysis Library (PAL) oder Business Function Library (BFL) verwenden.
.analyticview	Analytic View	Dieser Dateityp enthält die Definition eines Analytic Views und kann in einer OData-Service-Definition referenziert werden.
.attributeview	Attribute View	Dieser Dateityp enthält die Definition eines Attribute Views und kann in einer OData-Service-Definition referenziert werden.
.calculationview	Calucation View	Dieser Dateityp enthält die Definition eines Calculation Views und kann in einer OData-Service-Definition referenziert werden.
.hdbdd	CDS-Dokument	Dieser Dateityp enthält die Definition eines CDS-konformen Data-Persistence-Objekts, das mithilfe von DDL angelegt wird.
.hdbprocedure	Prozedur	Dieser Dateityp ersetzt den Dateityp *.procedure* und enthält die Definition einer Datenbankfunktion zur Ausführung von komplexen und datenintensiven Operationen, die nicht mit Standard-SQL ausgeführt werden können.
.hdbrole	Rolle	Mithilfe dieses Dateityps können die Rollen eines SAP-HANA-Benutzers definiert werden.
.hdbscalarfunction	benutzerdefinierte, skalare Funktion	Dateien von diesem Typ speichern die Designtime-Definition von skalaren, benutzerdefinierten Funktionen (UDF). Diese skalaren Funktionen können innerhalb von `SELECT`- und `WHERE`-Klauseln verwendet werden.

Tabelle 2.1 Übersicht der Entwicklungsobjekte einer nativen Anwendung

Dateierweiterung	Entwicklungs-objekt	Beschreibung
.hdbschema	Schema	Dieser Dateityp definiert das Schema, nach dem die Datenbank-objekte einer nativen Anwendung zugeordnet werden.
.hdbsequence	Sequenzen	Dieser Dateityp enthält die Definition einer Sequenz.
.hdbstructure	Tabellentypen	Dieser Dateityp enthält die Definition von Tabellentypen unter Verwendung der HDBTable-Syntax (siehe Abschnitt 3.5, »Datenmodell mit HDBTable definieren«).
.hdbtable	Tabelle	Dieser Dateityp enthält die Definition einer Tabelle unter Verwendung der HDBTable-Syntax.
.hdbtablefunction	benutzerdefinierte Tabellenfunktionen	In Dateien dieses Typs können benutzerdefinierte Funktionen (UDF) für eine Tabelle definiert werden. Diese Funktionen können in der `FROM`-Klausel einer SQL-Anweisung verwendet werden.
.hdbtextbundle	Ressource Bundle	Dieser Dateityp beinhaltet Übersetzungen von UI-Texten, die in SAPUI5-Anwendungen verwendet werden.
.hdbti	Import-definition von Tabellen	Dieser Dateityp beinhaltet die Konfiguration zum Importieren externer Dateien aus CSV-Dateien.
.hdbview	SQL-View	Dieser Dateityp beinhaltet die Definition von Views unter Verwendung der HDBTable-Syntax.
.proceduretemplate	Template für eine Prozedur	In Dateien dieses Typs können Designtime-Objekte vordefiniert werden. Diese Dateien beinhalten ein Skript mit vordefinierten Platzhaltern für Objekte wie Tabellen, Views und Columns.

Tabelle 2.1 Übersicht der Entwicklungsobjekte einer nativen Anwendung (Forts.)

Dateierweiterung	Entwicklungs-objekt	Beschreibung
.project	Projekt	Projektdatei einer nativen SAP-HANA-Anwendung, die mit dem SAP HANA Studio entwickelt wird. Diese Datei wird ebenfalls im Repository von SAP HANA gespeichert.
.searchruleset	Search Rule Set	In dieser Datei wird ein Satz von Regeln definiert, der im Rahmen einer Fuzzy-Suche verwendet wird.
.xsaccess	Application-Access-Datei	In dieser Datei werden die Zugriffsberechtigungen für eine native SAP-HANA-Anwendung konfiguriert.
.xsapp	Application Descriptor	Der Application Descriptor markiert das Root-Verzeichnis einer nativen SAP-HANA-Anwendung.
.xsappsite	Application Site	In dieser Datei wird die Definition einer SAP Fiori Launchpad Site gespeichert.
.xshttpdest	HTTP-Destination	In dieser Datei werden die Details für die Verbindung zu einem externen HTTP-Ziel über HTTP (oder HTTPS) definiert.
.xsjob	Scheduled XS Job	Diese Datei beinhaltet die JSON-konforme Definition von Aufgaben, die in regelmäßigen Zeitintervallen ausgeführt werden sollen.
.xsjs	serverseitiger JavaScript-Code	Dateien von diesem Typ beinhalten den JavaScript-Code, der durch die XS Engine ausgeführt werden kann. Dieser JavaScript-Code kann über eine URL aufgerufen werden.
.xsjslib	serverseitige JavaScript-Bibliothek	Serverseitige JavaScript-Bibliotheken enthalten wiederverwendbare Funktionen, die in serverseitigen JavaScript-Dateien (*.xsjs*) verwendet werden können. Serverseitige JavaScript-Bibliotheken können nicht über eine URL ausgeführt werden.

Tabelle 2.1 Übersicht der Entwicklungsobjekte einer nativen Anwendung (Forts.)

Dateierweiterung	Entwicklungs-objekt	Beschreibung
.xsoauthappconfig	OAuth-Appli-kation-Konfigu-rationsdatei	Diese Datei beinhaltet die High-Level-Definition einer Anwendung, die einen Service verwendet, der durch das OAuth-Protokoll abge-sichert ist.
.xsoauthclientflavor	OAuth-Client-Konfigurations-datei	Diese Datei beinhaltet detailliertere Informationen über eine Anwen-dung, die OAuth zur Authentifizie-rung verwendet, um sich mit einer externen HTTP-Anwendung zu ver-binden.
.xsodata	OData Descriptor	Dateien dieses Typs beinhalten die Definition eines OData-Services zur Designtime.
.xsprivileges	Application Privilege	Diese Datei definiert die Privileges, die einer XS-Anwendung zugewie-sen werden.
.xssecurestore	Application Secure Store	Diese Designtime-Datei erstellt einen anwendungsspezifischen Sicherheitsspeicher. Dieser Spei-cher wird von Anwendungen ver-wendet, um Daten als Key-Value-Paar sicher zu speichern.
.xssqlcc	SQL-Verbin-dungskonfigu-ration	In dieser Datei wird die Verbindung einer nativen SAP-HANA-Anwen-dung zur SAP-HANA-Datenbank konfiguriert.
.xswidget	Widget	Diese Datei definiert eine eigen-ständige SAP-HANA-Anwendung, um diese in eine Anwendungsseite zu integrieren.
.xsxmla	XMLA Descriptor	In Dateien dieses Typs werden XMLA-Services zur Designtime definiert, um analytische Daten aus SAP HANA externen Anwendungen zur Verfügung zu stellen.

Tabelle 2.1 Übersicht der Entwicklungsobjekte einer nativen Anwendung (Forts.)

2.5 Spezielle Objekte der SAP HANA Cloud Platform

In Abschnitt 2.2, »Ihre erste Anwendung erstellen«, haben wir den Trial-Account der SAP HCP zur Erstellung unserer ersten nativen SAP-HANA-Anwendung verwendet. Diese Plattform können Sie auch für die Erstellung weiterer nativer Anwendungen verwenden. Auch in den weiteren Abschnitten dieses Buches werden wir versuchen, alle wesentlichen Beispiele mit dem Entwickler-Account der SAP HCP zu erstellen. In bestimmten Situationen müssen wir jedoch auf eine On-Premise-Installation ausweichen, da es aufgrund von eingeschränkten Berechtigungen mit dem Entwickler-Account der SAP HCP gewisse Einschränkungen gibt.

Datenbankschemata

Beim Anlegen einer SAP-HANA-Instanz in der SAP HCP werden zwei neue Datenbankschemata angelegt und Ihrem Account zugewiesen. Ein Datenbankschema beginnt mit dem Präfix `DEV_`, das andere mit dem Präfix `NEO_`:

- Das Datenbankschema mit dem Präfix `NEO_` ist das Datenbankschema für die Entwicklung. In diesem Schema sollten Sie alle Datenbankobjekte wie Tabellen, Views, Prozeduren usw. anlegen, die für eine Anwendung benötigt werden. Dieses Schema bezeichnen wir im Folgenden als *Anwendungsschema*.
- Das Datenbankschema mit dem Präfix `DEV_` ist Ihr Schema. Dieses bezeichnen wir im Folgenden als *Entwicklungsschema*. In dem `DEV_`-Schema sollten Sie keine Datenbankobjekte anlegen, die für eine Anwendung benötigt werden.

Berechtigungen

Die Benutzerkennung des Datenbankbenutzers beginnt ebenfalls mit dem Präfix `DEV_`. Damit mehrere Benutzer auf der SAP HCP gleichzeitig und unabhängig voneinander Anwendungen entwickeln können, ist es notwendig, die Rechte zu begrenzen.

Systemschemata

Neben diesen beiden neu angelegten Datenbankschemata haben Sie unter anderem noch Zugriff auf die folgenden Systemschemata:

- `_SYS_BIC`
  Dieses Schema enthält alle Column Views der aktivierten Datenbankobjekte wie Tabellen, Views, Sequenzen, Prozeduren etc. Im Rahmen der Aktivierung von Datenbankobjekten werden die Laufzeitobjekte im Schema `_SYS_BIC` erstellt. Column Views wer-

den unter anderem erstellt, wenn mit dem Modeler neue Views erstellt werden.

- **_SYS_REPO**
  Dieses Schema wird im SAP HANA Studio während der Aktivierung der Entwicklungsobjekte verwendet. Alle Objekte, die Sie im SAP HANA Studio erstellen, werden dem Datenbankbenutzer _SYS_REPO zugewiesen.
- **_SYS_BI**
  In diesem Schema werden die Metainformationen der erzeugten Column-Views gespeichert. Dieses Schema beinhaltet unter anderem die Tabellen für erzeugte Variablen, Daten für Zeitangaben (Geschäftsjahr, Gregorianischer Kalender) sowie Schema- und Content-Mapping-Tabellen.

Bei der initialen Erstellung Ihres Accounts und des technischen Datenbankbenutzers werden diesem nur die notwendigen Rechte zugewiesen. Für bestimmte Aufgaben – wie z.B. das Debuggen einer nativen Anwendung – müssen Sie jedoch Ihre Rechte erweitern. Hierzu gibt es im HCP-Datenbankschema eine Reihe von Prozeduren. Mit diesen Prozeduren können Sie folgende Berechtigungen erweitern:

- Berechtigungen für Views
- Berechtigungen für Rollen
- Privileges des Accounts
- Berechtigungen von Rollen für Endbenutzer
- Zugriff auf die Metadaten des Accounts

Prozeduren zur Berechtigungserweiterung

Die wichtigsten Prozeduren stellen wir Ihnen hier kurz vor:

- **Metadaten Ihres Accounts**
  Für die Abfrage Ihrer Metadaten können Sie folgende Views verwenden:
  - **HCP_DEV_METADATA**
    Mit diesem View erhalten Sie das Ihnen zugewiesene Anwendungsschema sowie das entsprechende Paket und die Rollen.

    ```
    SELECT * FROM "HCP"."HCP_DEV_METADATA"
    ```

  - **HCP_ACTIVATED_ROLES**
    Mit diesem View erhalten Sie alle aktiven HDB-Rollen, die Ihrem Datenbankbenutzer zugewiesen wurden. Die Erstellung

und Aktivierung von HDB-Rollen stellen wir Ihnen in Kapitel 3, »Definition des Datenmodells«, noch detaillierter vor.

```
SELECT * FROM "HCP"."HCP_ACTIVATED_ROLES"
```

- **Zuweisung der Rechte für modellierte Views**
  Um auf die aktivierten Views, die im SAP HANA Studio modelliert wurden, auf HDB-Views oder Prozeduren zugreifen zu können, müssen Sie eine der folgenden SQL-Prozeduren aufrufen:
  - **HCP_GRANT_SELECT_ON_ACTIVATED_OBJECT**
    Mit dieser Prozedur können Sie der Rolle des Anwendungsschemas den Zugriff auf einen bestimmten View, HDB-View oder eine SQL-Prozedur erlauben. Diese SQL-Prozedur benötigt zwei Parameter:
    - Name des Pakets, in dem das entsprechend Objekt gespeichert ist
    - Name des Objekts, für das der Zugriff erteilt werden soll

    ```
    CALL "HCP"."HCP_GRANT_SELECT_ON_ACTIVATED_
      OBJECT"(.hana.xs', 'CURRENT_WEATHER')
    ```

  - **HCP_GRANT_SELECT_ON_ACTIVATED_OBJECTS**
    Mit dieser Prozedur können Sie die Berechtigung für den Zugriff auf alle aktivierten Views aus dem Entwicklerpaket an den Benutzer des Anwendungsschemas vergeben:

    ```
    CALL "HCP"."HCP_GRANT_SELECT_ON_ACTIVATED_OBJECTS"
    ```

Einleitung

Werfen Sie einen ersten Blick auf SAP HANA, sehen Sie zunächst eine hardwareoptimierte, relationale In-Memory-Datenbank für SAP-Anwendungen, wie z.B. die SAP Business Suite für Enterprise Resource Planning (ERP). Schon bei einem zweiten Blick fällt jedoch auf, dass SAP HANA mehr als eine Datenbank ist. Insbesondere für Anwendungsentwickler, die bisher noch überhaupt keine Berührungspunkte mit SAP-Anwendungen haben, bietet SAP HANA einige interessante Möglichkeiten zur Implementierung innovativer Anwendungen. Genau um diesen zweiten Blick geht es in diesem Buch. Es richtet sich an alle, die mehr darüber erfahren wollen, welche Möglichkeiten SAP HANA bei der Programmierung eigener Anwendungen bietet.

Warum native Anwendungen für SAP HANA programmieren?

In einem datengetriebenen Wirtschaftsumfeld geht es darum, möglichst effizient und einfach Informationen aus Daten zu gewinnen. Diese Anforderung der *Einfachheit* bezieht sich dabei zum einen auf die notwendige Infrastruktur, die für die Informationsgewinnung benötigt wird. Zum anderen bezieht sie sich auf die Art, wie Sie die Software zur Informationsgewinnung implementieren und wie intuitiv Ihre Kunden die Anwendung nutzen können.

In allen drei Bereichen – Infrastruktur, Implementierung und Nutzung – bietet SAP HANA Vorteile. Dieses Buch stellt Ihnen diese Vorteile bei der Implementierung von Anwendungen vor. Sie ergeben sich insbesondere aus der Kombination einer leistungsfähigen In-Memory-Datenbank mit zahlreichen weiteren Funktionen, die Sie bei der Anwendungsentwicklung benötigen. Angefangen bei einem integrierten Applikationsserver, auf dem Sie Ihre native SAP-HANA-Anwendung installieren, stehen Ihnen viele weitere Möglichkeiten zur Verfügung, um Informationen aus Ihren Daten zu gewinnen. So haben Sie z.B. die Möglichkeit, transaktionale und analytische Anfragen auf Basis desselben Datenbestands zu realisieren.

Operatives Reporting

Durch das *operative Reporting* können Sie mit SAP HANA Anwendungen entwickeln, die Massendaten kontinuierlich verarbeiten, in voller Detailschärfe auswerten und die Ergebnisse in Echtzeit darstellen können. Dies vereinfacht nicht nur die Ihrer Anwendung zugrunde liegende Lösungsarchitektur, sondern erlaubt auch die Umsetzung neuartiger Anwendungen, die im Bereich der Echtzeitauswertung liegen.

Predictive Analysis

Um die Auswertung der Daten nicht nur auf reine Kennzahlen zu beschränken, bietet SAP HANA über 90 Data-Mining- bzw. Predictive-Analysis-Algorithmen, die Sie bei der Anwendungsprogrammierung verwenden können. So gruppieren Sie z.B. Kundendaten mit Cluster-Algorithmen direkt in SAP HANA und müssen dafür keine Kopie der Daten für eine andere Softwarekomponente zur Analyse und Datenaufbereitung herstellen. Dadurch, dass diese Algorithmen integraler Bestandteil von SAP HANA sind, werden sie schnell und effizient auch auf Basis großer Datenmengen ausgeführt. Dies erlaubt Ihnen Flexibilität bei der Nutzung der Algorithmen innerhalb Ihrer Anwendung.

Unstrukturierte Daten

Zur Verarbeitung unstrukturierter Daten, wie z.B. Kommentaren in sozialen Medien, nutzt SAP HANA neben linguistischen Analysen auch Extraktionsverfahren zur Klassifikation von Inhalten, wie z.B. Orten, Organisationen, Unternehmen oder positiven und negativen Äußerungen. Auch Anwendungsfälle im Umfeld von Big Data können Sie erschließen, da SAP HANA Daten aus externen, relationalen Datenbanksystemen und Hadoop transparent einbindet.

Wie ist dieses Buch aufgebaut?

In **Kapitel 1**, »SAP HANA als Entwicklungsplattform«, lernen Sie die Grundlagen der SAP-HANA-Plattform kennen. Neben den wichtigsten Komponenten der Plattform stellen wir Ihnen in diesem Kapitel auch einige Anwendungsszenarien für den Einsatz von SAP HANA vor. Des Weiteren lernen Sie den Unterschied zwischen einer nativen und einer nicht nativen SAP-HANA-Anwendung kennen. Damit Sie die Beispiele, die wir in dieses Buch aufgenommen haben, auch ausprobieren können, zeigen wir Ihnen, wie Sie einen Account für die Trial-Instanz der SAP HANA Cloud Platform erstellen, um kostenlos auf SAP HANA zugreifen zu können, und wie Sie Ihre Entwicklungsumgebung einrichten.

SAP HANA Extended Application Services

Kapitel 2, »Struktur einer nativen SAP-HANA-Anwendung«, erläutert den Aufbau und die Verwendung der SAP HANA Extended Application Services (XS) sowie die Struktur einer nativen SAP-HANA-Anwendung. Mit einem ersten einfachen Beispiel führen wir Sie Schritt für Schritt in die Programmierung einer nativen Anwendung ein.

Datenmodell

In **Kapitel 3**, »Definition des Datenmodells«, lernen Sie die verschiedenen Möglichkeiten zur Definition von Datenmodellen kennen. Dazu stehen Ihnen Core Data Services (CDS) oder HDBTables zur Verfügung. Deren Verwendung und die Unterschiede zwischen den beiden Modellen erklären wir in diesem Kapitel. Des Weiteren erhalten Sie einen Einblick in die verschiedenen Möglichkeiten zur Befüllung von Datenbanktabellen.

Analytische Modelle

Kapitel 4, »Echtzeitauswertung mit Information Views«, zeigt die Erstellung eines analytischen Modells zur direkten Auswertung von transaktionalen Daten. Damit lernen Sie in diesem Kapitel die Möglichkeiten des operativen Reportings kennen. Hierzu stellt SAP HANA verschiedene Arten von Views bereit. Außerdem lernen Sie in diesem Kapitel die Analytic Privileges kennen, ein Berechtigungskonzept für den Zugriff auf diese Views.

Anwendungsentwicklung

In **Kapitel 5**, »Anwendungsentwicklung mit der SAP HANA XS Engine«, zeigen wir Ihnen die Erstellung von nativen SAP-HANA-Anwendungen. Nach einer kurzen Einführung in die serverseitige Programmierung mit JavaScript zeigen wir Ihnen, wie Sie XS-JavaScript-Bibliotheken erstellen und diese in native Anwendungen integrieren. Ein weiterer wesentlicher Bestandteil dieses Kapitels ist die Einführung und Verwendung der von der XS Engine bereitgestellten APIs. Als Beispiel zeigen wir Ihnen, wie Sie online Wetterdaten ermitteln und diese in der SAP-HANA-Datenbank speichern.

XS Data Services

Kapitel 6, »Erweitertes Programmiermodell mit den SAP HANA XS Data Services«, behandelt die XS Data Services (XSDS), die mit SAP HANA SP09 eingeführt wurden. Sie erweitern das Programmiermodell durch wiederverwendbare XS-JavaScript-Bibliotheken.

SAPUI5

In **Kapitel 7**, »Entwicklung von Benutzeroberflächen«, zeigen wir Ihnen, wie Sie mit SAPUI5 grafische Benutzeroberflächen erstellen, die sowohl in einem Browser als auch auf einem mobilen Endgerät ausgeführt werden können. Neben den wichtigsten Konzepten bringen wir Ihnen nahe, wie Sie schrittweise eine SAPUI5-Anwendung erstel-

len. Diese Anwendung wird die Wetterstationen einer ausgewählten Stadt in einer Master-Detail-Ansicht anzeigen. Dieses Beispiel lässt Ihnen Spielraum, diese Anwendung entsprechend Ihren Vorstellungen weiterzuentwickeln.

Datenanalyseszenarien

In **Kapitel 8**, »Verarbeitung räumlicher und unstrukturierter Daten«, lernen Sie weitere Möglichkeiten zur Datenanalyse mit SAP HANA kennen. Mit der Textanalysefunktion können Sie z.B. unstrukturierte Daten wie Dokumente und Tweets nach bestimmten Schlagwörtern durchsuchen. Auch können Sie raumbezogene Daten oder zahlreiche Algorithmen zur prädiktiven Analyse verarbeiten.

SQLScript

Kapitel 9, »SQLScript«, erläutert, wie datenintensive Berechnungen mithilfe von SQLScript in die Datenbank ausgelagert, optimiert und performant verarbeitet werden können. Auch das Debugging von SQLScript-Prozeduren wird hier beschrieben. Diese Berechnungen können dann in native SAP-HANA-Anwendungen integriert werden.

OData-Services

In **Kapitel 10**, »Webbasierter Datenzugriff«, lernen Sie, wie Sie mithilfe des HTTP-Protokolls auf die in der SAP-HANA-Datenbank gespeicherten Daten zugreifen und diese veröffentlichen können. In diesem Kapitel legen wir den Fokus auf die Erstellung und Verwendung von OData-Services. Darüber hinaus stellen wir Ihnen XML for Analytics (XMLA) für die Abfrage von Data Cubes vor.

Testen und Debugging

In **Kapitel 11**, »Native SAP-HANA-Anwendungen debuggen und testen«, zeigen wir Ihnen, wie Sie Ihre nativen Anwendungen mit dem SAP HANA Studio oder mit der SAP HANA Web-based Development Workbench debuggen können. Außerdem stellen wir Ihnen das mit SAP HANA SPS 09 eingeführte Framework XSUnit vor und zeigen Ihnen, wie Sie Tabellen und Views simulieren können.

Application Lifecycle Management

In **Kapitel 12**, »Lebenszyklus einer nativen SAP-HANA-Anwendung verwalten«, erklären wir, wie Sie die entwickelten Objekte zu einem Produkt zusammenstellen und mit den sogenannten Delivery Units verteilen. Hierbei werden Sie von der Anwendung SAP HANA Application Lifecycle Management unterstützt.

XS Engine Advanced

Kapitel 13, »SAP HANA Extended Application Services Advanced und weitere neue Konzepte«, beschreibt Neuerungen aus SAP HANA SPS 11. Mit diesem Release wurde die XS Engine Advanced eingeführt, mit der native Anwendungen in weiteren Programmiersprachen entwickelt werden können. Auch im Bereich der Core Data Services und des Deployments gibt es einige Neuigkeiten, die wir Ihnen hier vorstellen.

Bei der Zusammenstellung der Kapitel haben wir darauf geachtet, alle Aspekte der nativen Anwendungsentwicklung mit der SAP-HANA-Plattform zu berücksichtigen. Dabei haben wir versucht, uns auf das Wesentliche zu beschränken. Nach dem Lesen dieses Buches sollten Sie über alle notwendigen Informationen für die Erstellung von nativen Anwendungen verfügen. Wir hoffen, dass wir bei Ihnen das Interesse für die Erstellung nativer Anwendungen geweckt haben und Sie viel Spaß daran finden werden.

So arbeiten Sie mit dem Buch

Sie können dieses Buch sowohl als Einführung als auch als Nachschlagewerk verwenden. Die Inhalte der einzelnen Kapitel veranschaulichen wir mit praktischen Beispielen. Die Beispiele können Sie entweder auf Ihrem fest installierten SAP-HANA-System oder z.B. in der kostenfreien Developer Edition der SAP HANA Cloud Platform (HCP) umsetzen.

Arbeiten Sie das Buch Kapitel für Kapitel durch, werden Sie nach und nach eine vollständige native SAP-HANA-Anwendung aufbauen. Liegt Ihr Schwerpunkt mehr im Bereich der Datenmodellierung oder Datenanalyse, können Sie die praktischen Beispiele in den Kapiteln 2, 3, 4, 8 und 9 in dieser Reihenfolge durcharbeiten. Liegt Ihr Schwerpunkt in der Anwendungsprogrammierung, können Sie die Kapitel 2, 5, 6, 7, 10 und 11 der Reihenfolge nach durcharbeiten.

Informationskästen

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

- **Checkliste** [▪]
  Kästen mit diesem Icon zeigen Ihnen, welche notwendigen Voraussetzungen (z.B. Tools) Sie in den Kapiteln benötigen.
- **Hinweise** [«]
  In Kästen, die mit diesem Symbol gekennzeichnet sind, finden Sie Informationen zu *weiterführenden Themen* oder wichtigen Inhalten, die Sie sich merken sollten.

[+] ▸ **Tipp**
Kästen mit diesem Icon geben Ihnen Empfehlungen zu Einstellungen oder Tipps aus der Berufspraxis.

[!] ▸ **Achtung**
Kästen mit diesem Icon geben Ihnen besonders wichtige Hinweise zur besprochenen Thematik. Außerdem warnen wir Sie hier vor möglichen Fehlerquellen.

Codebeispiele

Dieses Buch enthält eine Vielzahl an Codebeispielen, um Syntax, Funktionen etc. zu veranschaulichen. Um diese Abschnitte zu kennzeichnen, verwenden wir eine Schriftart, die mit der in vielen integrierten Entwicklungsumgebungen eingesetzten Schriftart vergleichbar ist, um die Lesbarkeit von Code zu verbessern (siehe Listing 1). Wenn neue Syntaxkonzepte eingeführt werden, werden diese Anweisungen durch fett formatierte Listingschrift gekennzeichnet.

```
function getCurrentUser() {
  var body;
  try {
    var conn =
 $.db.getConnection("sap.hana.sqlcon::AdminConn");
    var pStmt = conn.prepareStatement("SELECT CURRENT_
USER FROM dummy");
    var rs = pStmt.executeQuery();
    if (rs.next()) {
      body = rs.getNString(1);
    }
    …
  }
  $.response.status = $.net.http.OK;
  $.response.setBody( body );
}
```

Listing 1 Formatierungsbeispiel für Codesyntax

Download des Beispielcodes

Im Verlauf dieses Buches werden wir eine Beispielanwendung entwickeln, die die wichtigsten Konzepte veranschaulicht. Um es so einfach wie möglich zu machen, diese Arbeitsbeispiele herunterzuladen, haben wir ein öffentliches Repository für den gesamten Code im Buch auf GitHub erstellt. Die URL des Repositorys lautet *https://github.com/nativeDevelopment/XSGeoWeatherApp*. Diese Informationen stehen auch unter *www.sap-press.de/3916* zum Download zur Verfügung, zu finden unter den Materialien zum Buch.

Falls nötig, haben wir auch Readme-Dateien zu den verschiedenen Beispielen zur Verfügung gestellt, um Ihnen zu zeigen, wie Sie sie zur Bereitstellung in Ihrem eigenen Trial-Account der SAP HANA Cloud Platform konfigurieren müssen.

Wir freuen uns auf Ihr persönliches Feedback zum Buch unter *stefan.kuehnlein@web.de* und *hseubert@gmail.com*.

Danksagung von Stefan Kühnlein

Es ist schon ein etwas merkwürdiges Gefühl, wenn man am Ende des Schreibens eines Buches das vergangene Jahr im Geiste an sich vorbeiziehen lässt. Zum einen breitet sich das Gefühl der Zufriedenheit aus, da dieses Buch nun fertig geschrieben ist und wieder Neues in den Mittelpunkt rücken kann. Aber auch das Gefühl der Dankbarkeit steigt in mir auf: Dankbarkeit für eine gutes Gelingen und die viele Unterstützung, die ich in dieser Zeit erfahren habe.

Ein ganz besonderer Dank geht zuerst an meine Frau Diana und an meine beiden Söhne Raphael und Yannick. Da ich dieses Buch in meiner Freizeit geschrieben habe, blieben mir hierzu nur die Abende, das Wochenende und der Urlaub. In dieser Zeit brachte mir meine Familie sehr viel Verständnis und Geduld entgegen. Immer, wenn ich im Büro verschwand, wussten meine Jungs, dass ich wieder am Buch schreibe. An den Abenden, wenn ich sie ins Bett brachte, fragten sie immer nach, wie viele Seiten ich denn schon geschrieben habe und wie viele Seiten ich noch schreiben muss. Vor allem meine Frau Diana hielt mir den Rücken frei, so dass ich mich auf das Schreiben dieses Buches konzentrieren konnte. Diese Unterstützung von meiner Familie hat mich stets angetrieben, dieses Buch zur Vollendung zu bringen.

Danken möchte ich dieser Stelle auch Jochen Wilms und Andreas Becht. In Ihrer Funktion als Manager Business Development bzw. Leiter der Competence Unit Software Development haben Sie das Thema der Entwicklung von Anwendungen mit der SAP-HANA-Plattform bei der OPITZ CONSULTING Deutschland GmbH positioniert. Nach einer anfänglichen Evaluationsphase wurde mir die Leitung des Competence Centers SAP HANA Development übertragen. Dadurch konnte ich mich sehr intensiv mit der Erstellung von Anwendungen beschäftigen, die die SAP-HANA-Plattform nutzen. Nicht

zuletzt gehört ihnen beiden der Dank, denn ohne den nötigen Mut zur Gründung des Competence Centers und dem damit verbundenen Know-how-Aufbau hätte ich dieses Buch nicht schreiben können.

Ein weiterer Dank geht an unsere Lektorin Janina Schweitzer, die die Idee hatte, ein Buch über die native Anwendungsentwicklung mit der SAP-HANA-Plattform zu schreiben. An dieser Stelle möchte ich mich für das entgegengebrachte Vertrauen, die Geduld und für die tatkräftige Unterstützung bei der Erstellung dieses Buches bedanken.

Danksagung von Holger Seubert

Ein Fachbuch über SAP HANA ist nicht einfach zu schreiben, insbesondere, weil sich das Produkt, um das es geht, mit rasanter Geschwindigkeit entwickelt. Beim Schreiben dieses Buches hat mich eine Reihe von Personen unterstützt. Bedanken möchte ich mich an dieser Stelle insbesondere bei meiner Frau Nina und meinen Söhnen Felix und Philipp für ihre Unterstützung, bei Frau Janina Schweitzer, Lektorin bei SAP PRESS, für die unendliche Geduld und bei meinem Co-Autor Stefan für die gute Zusammenarbeit.