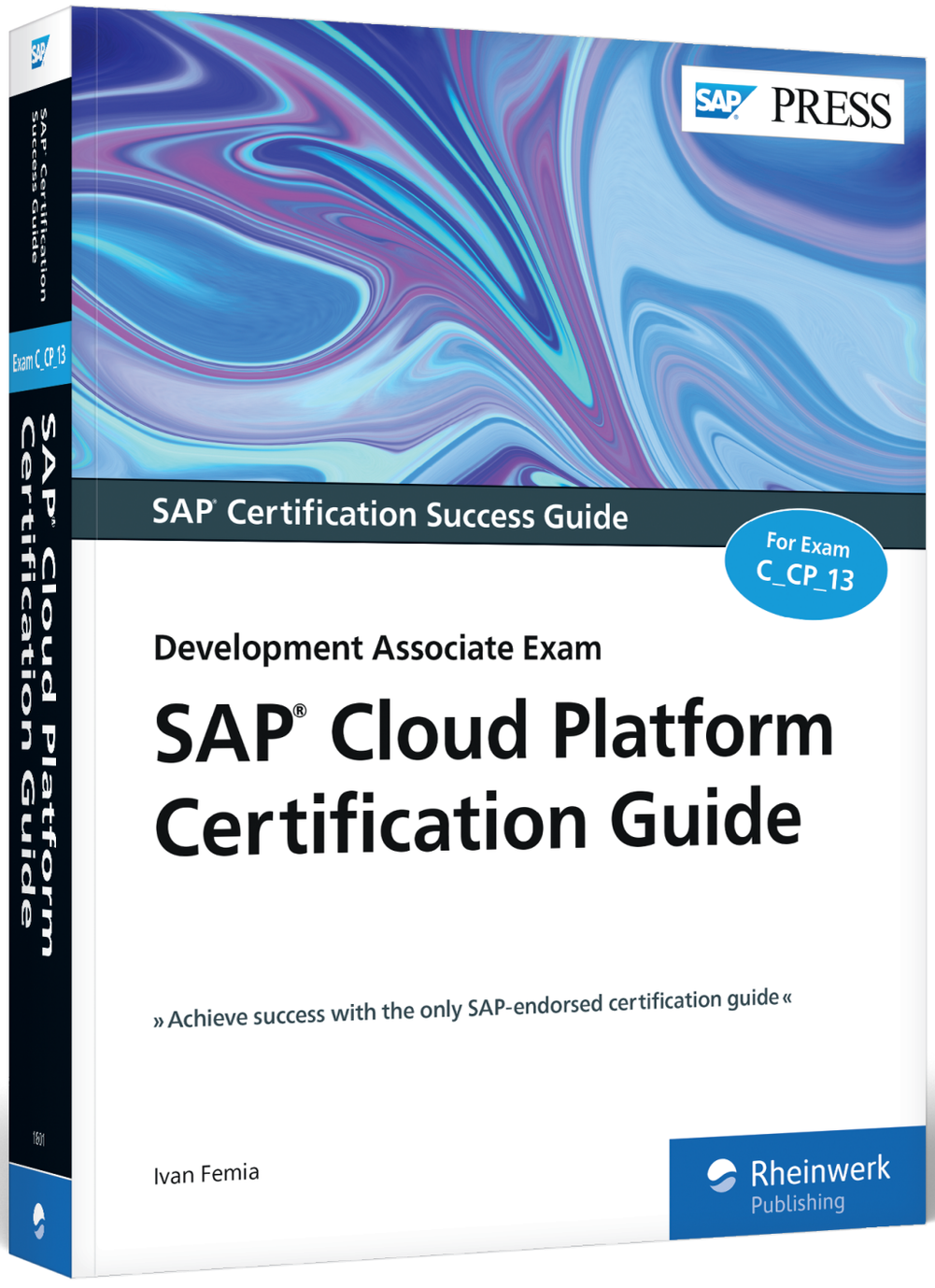


Browse the Book

In this chapter, we'll show you how this book covers SAP HANA development! You'll understand the different engines available in the various SAP HANA versions in the cloud and learn how to implement solutions on top of these services.

Ivan Femia

SAP Cloud Platform Certification Guide: Development Associate Exam

477 Pages, 2019, $79.95
ISBN 978-1-4932-1801-1

www.sap-press.com/4825

Chapter 3
SAP HANA Development

Techniques You'll Master:

- SAP HANA extended application services, classic model (SAP HANA XS) development
- SAP HANA extended application services, advanced model (SAP HANA XS Advanced) development

In this chapter, we'll explore SAP HANA's development capabilities within SAP Cloud Platform. We'll describe how to approach development using the SAP HANA XS engine and SAP Cloud Platform SDKs for both the Neo and Cloud Foundry environments.

In this chapter, we'll explore the tools available in SAP Cloud Platform for managing data storage in the cloud and discuss how to leverage capabilities embedded in SAP HANA to easily expose the data. This chapter is a key chapter because we'll set the stage for future chapters to understand the flexibility and the available capabilities when combining different SAP Cloud Platform services together.

Real-World Scenario

Data has become increasingly important in any corporate scenario, and we are challenged every day to collect, analyze, and display data. A great advantage of SAP Cloud Platform in many scenarios is its ability to easily provide data storage in the cloud that is fast, reliable, and scalable.

SAP HANA is a database as a service (DBaaS). By being an in-memory database in the cloud, it can easily adapt to different use cases.

Let's consider a new Internet of Things (IoT) project: The amount of data collected from sensors and devices cannot be stored in our SAP S/4HANA system, that is, the system of record, but we don't want to transfer data to physical storage on-premise, which could become expensive. In this case, you could create a database in the cloud that can easily and rapidly respond to your business requirements.

In some cases, you may want to extend your transactional data in SAP S/4HANA, for example, a sales and distribution (SD) process, with some information that is not directly relevant to your system of records, for example, weather information that may affect the delivery process. This information is suitable for storage in the cloud as additional information that will enrich your business processes.

Objectives of This Portion of the Test

The purpose of this portion of the certification exam is to test your knowledge of the knowledge of integration services and to confirm your ability to improve business agility in the cloud.

The certification exam expects you to have a good understanding of the following topics:

- SAP HANA development using SAP HANA XS
- SAP HANA development using SAP HANA XS Advanced

Note
The mobile service topic makes up 12% of the total exam.

Key Concepts Refresher

As mentioned earlier, we'll focus on the two different SAP Cloud Platform SDK environments: Neo and Cloud Foundry. We'll implement the same scenario, data storage for simple material master data, in the two SAP Cloud Platform environments. In section "SAP HANA XS Development," we'll use the SAP HANA XS engine in Neo and, in section "SAP HANA XS Advanced Development," the SAP HANA XS Advanced engine in Cloud Foundry. In addition, we'll leverage the capabilities of the SAP HANA XS engine to expose the information in an OData format.

Introduction to SAP HANA Development

SAP HANA extended application services (SAP HANA XS) is the paradigm adopted by SAP to promote SAP HANA development to simplify the architecture needed to create new applications by leveraging SAP HANA's in-memory database and by sponsoring code pushdown to the database, an approach that forms the basis of new developments in SAP applications.

In the past, you would have an application layer that performed all the hard work, querying data from different tables and looping, organizing, and manipulating the data with calculation. This process is expensive because the system must load a huge amount of "raw" data. In the SAP HANA approach, as shown in Figure 3.1, most of the logic is pushed down into the database to leverage the speed of an in-memory database's ability to perform advanced calculations directly where the data resides. The application logic, in this new approach, is leaner and focuses on exposing the data.

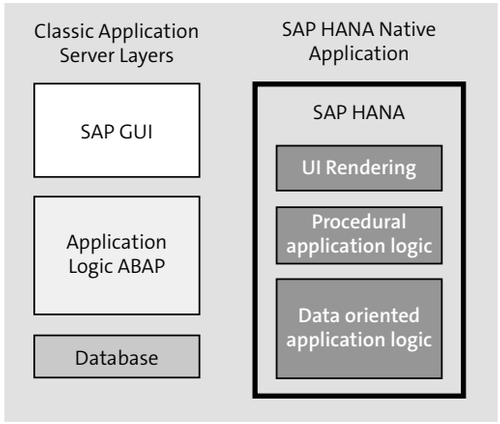


Figure 3.1 Architectural Paradigm Shift

The SAP HANA XS application has been available in SAP HANA for a long time, since SAP HANA support package stacks (SPS) 05, and developers started to successfully develop many of their own applications using the SAP HANA XS engine.

SAP HANA XS is the architecture that was first used in SAP HANA to approach SAP XS development, but limitations or gaps due to the technology adopted existed. These gaps have been overtaken by the new SAP HANA extended application services, advanced model (SAP HANA XS Advanced), starting with SAP HANA SPS 11, which has been developed using the Cloud Foundry concepts of microservices and openness.

The shift from SAP HANA XS to SAP HANA XS Advanced helps us leverage new concepts in Cloud Foundry, such as:

- Different runtimes and programming languages to develop applications, such as JavaScript on Node.js, Java on TomEE, and C++ via FastCGI
- Microservice approach, which allows applications to scale up as needed while being responsive and reliable
- Application compatibility between on-premise and cloud-based SAP HANA systems because both are based on the same architectural model
- Better version control of SAP HANA artifact, now with support for Git repositories

SAP HANA XS is still supported in SAP HANA. You can use this engine with the SAP Cloud Platform SDK for the Neo environment, while SAP HANA XS Advanced is the default engine used with the SAP Cloud Platform SDK for the Cloud Foundry environment.

SAP HANA XS Development

In this section, we'll cover SAP HANA XS development with the SAP Cloud Platform SDK for the Neo environment. We're using the SAP HANA 1.0 also known as XS classic. This service is also available in your trial account; you just need to enable it.

In this section, we'll walk step by step through developing an SAP HANA XS application using the SAP HANA XS, starting from building a database table, exposing data using an OData endpoint, and using the data to perform operations in the database table.

Create a New SAP HANA Instance

As done previously, go into the service list in the SAP Cloud Platform SDK for the Neo environment, and activate the SAP HANA service. Once the service is enabled, create a new instance of the SAP HANA database in SAP Cloud Platform. From the **SAP HANA/SAP ASE** section in the main menu area, select **Databases & Schemas**, which opens the SAP HANA service homepage where you'll see an overview of all the SAP HANA instances currently available.

Since in our case, no instances exist, we'll create a new one:

1. Let's go ahead and create a new instance by clicking the **New** button.
2. A simple form, shown in Figure 3.2, appears requesting you provide a **Database ID** ❶ for the database instance (in our case *mydb*) and a password for the SYSTEM user ❷.
3. Now, you're ready to click the **Create** button ❸.

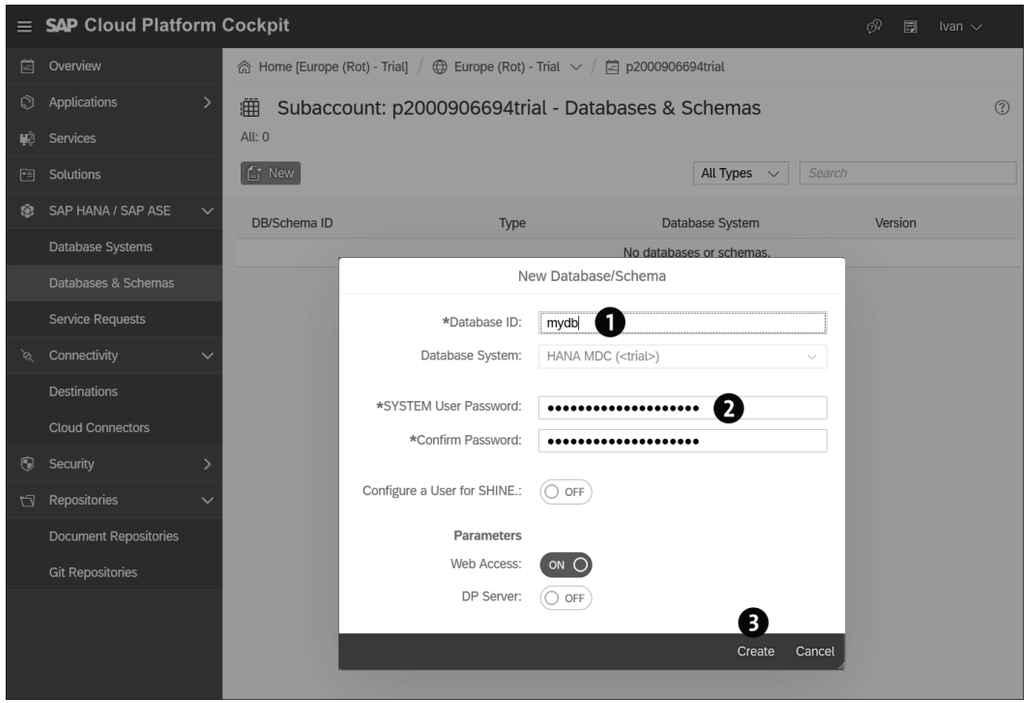


Figure 3.2 Create a New SAP HANA Instance in the SAP Cloud Platform SDK for the Neo Environment

The service starts to create a new instance and provides the step-by-step events that, in less than ten minutes, will deliver us a new SAP HANA instance in the cloud, as shown in Figure 3.3.

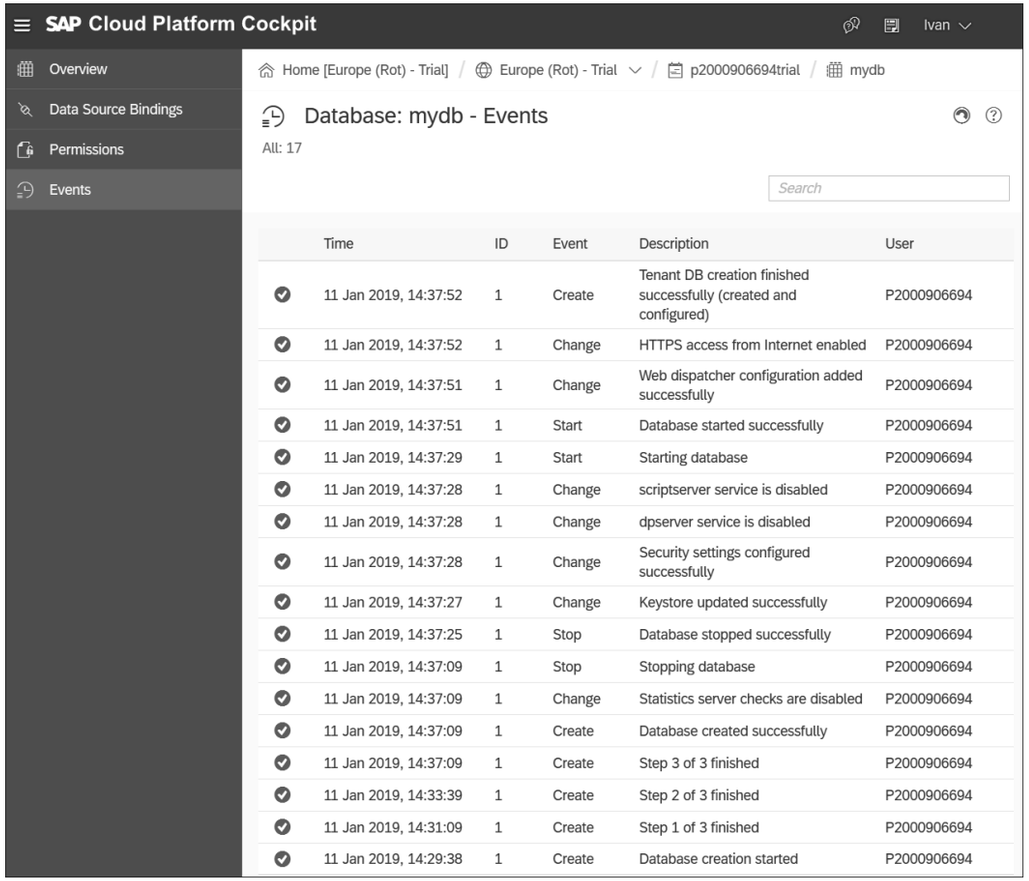


Figure 3.3 Database Creation Event Logs

Explore the SAP HANA Database

When the database instance is ready, click on **mydb** to open the database overview, shown in Figure 3.4. On this page, you'll have access to all the details related to your SAP HANA database: the version currently running, the Java database connectivity (JDBC) URL to connect with a JDBC client, the current configuration, and the service currently enabled or disabled. For our purposes, let's focus on web access and the SAP HANA XS engine.

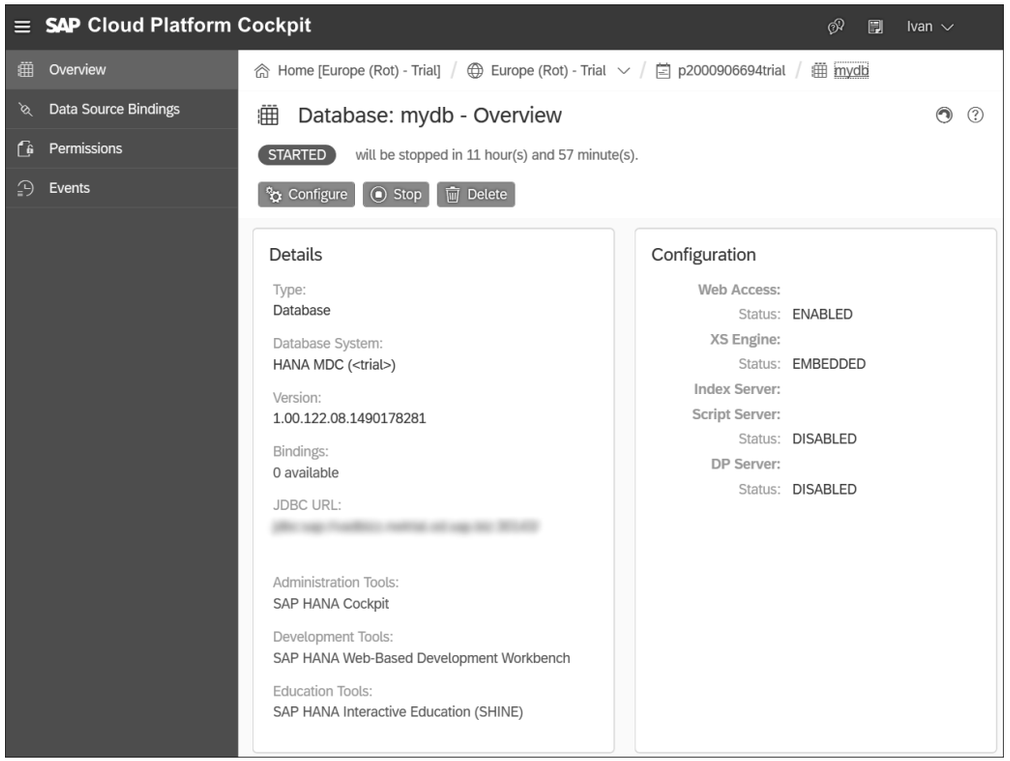


Figure 3.4 SAP HANA Database Overview

From the overview page, you can also access three additional tools:

- *SAP HANA cockpit*, which provides an administrative overview of the SAP HANA instance
- *SAP HANA web-based development workbench*, which enables SAP HANA developers to create tables and SAP HANA XS applications
- *SAP HANA interactive education (SHINE)*, which provides educational content that can be deployed in your SAP HANA instance and can be used to learn from preexisting scenarios.

SAP HANA Interactive Education

SAP HANA Demo Model 1.0 - SHINE is available for free and can be downloaded from the SAP Software Download Center (*https://support.sap.com/en/my-support/software-downloads.html*) or the SAP Store (*https://store.sap.com/sap/cpa/ui/resources/store/html/StoreFront.html*).

To access to the SHINE, you'll need to add two additional roles to your user, `sap.hana.democontent.epm.data.model_access` and `sap.hana.democontent.epm.data.model_admin`, which control application access and admin access, respectively.

The SAP HANA cockpit is reserved for database administrators and displays, as shown in Figure 3.5, all the general information about the runtime of your SAP HANA instance. This powerful tool consists of one page where an administrator can have a quick overview of the health of the SAP HANA database, with drilldown capability to find more technical information by clicking each tile.

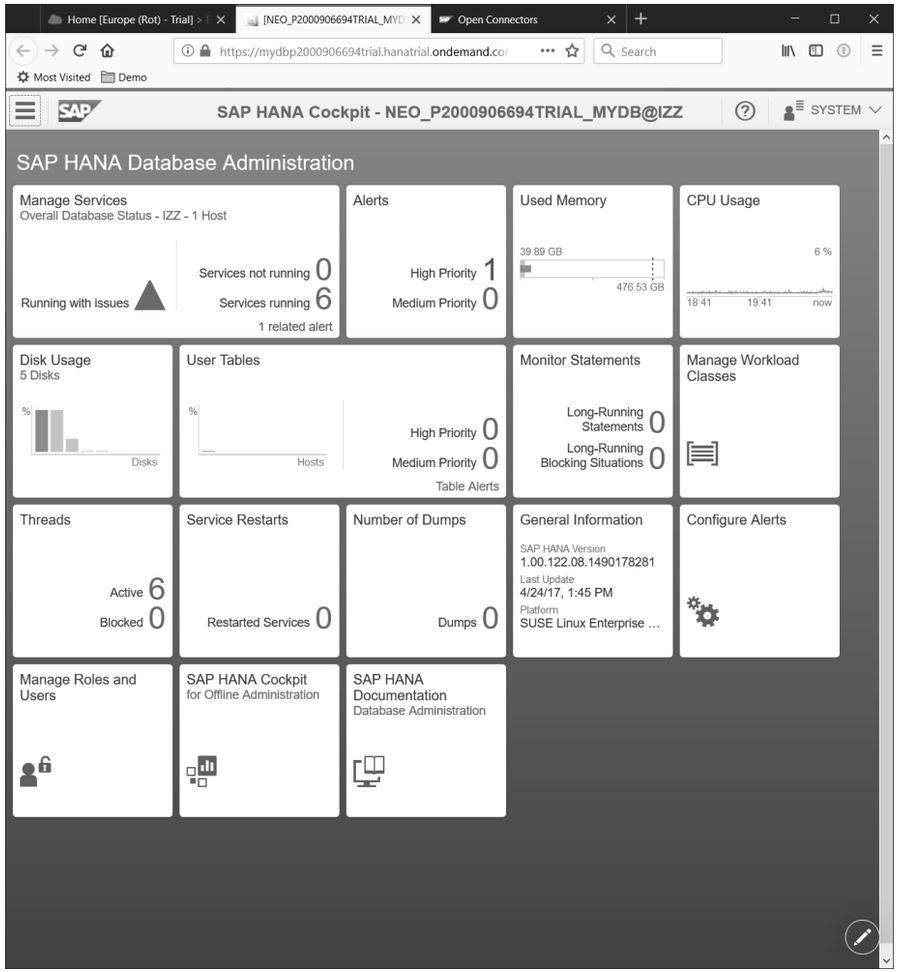


Figure 3.5 SAP HANA Cockpit

The SAP HANA Web-Based Development Workbench, shown in Figure 3.6, is the service designed for SAP HANA developers. This tool provides access to:

- *SAP HANA editor* to create, edit, debug, and manage SAP HANA repository artifacts
- *SAP HANA catalog* to create, edit, debug, and manage SAP HANA SQL artifacts
- *Security* to maintain user access roles to the SAP HANA instance
- *Traces* to maintain tracing settings and to monitor traces of our SAP HANA applications

Security

The security service, even if included in the SAP HANA Web-Based Development Workbench, should be accessible to only a few SAP HANA users, usually administrators that assign roles and groups to users/developers.

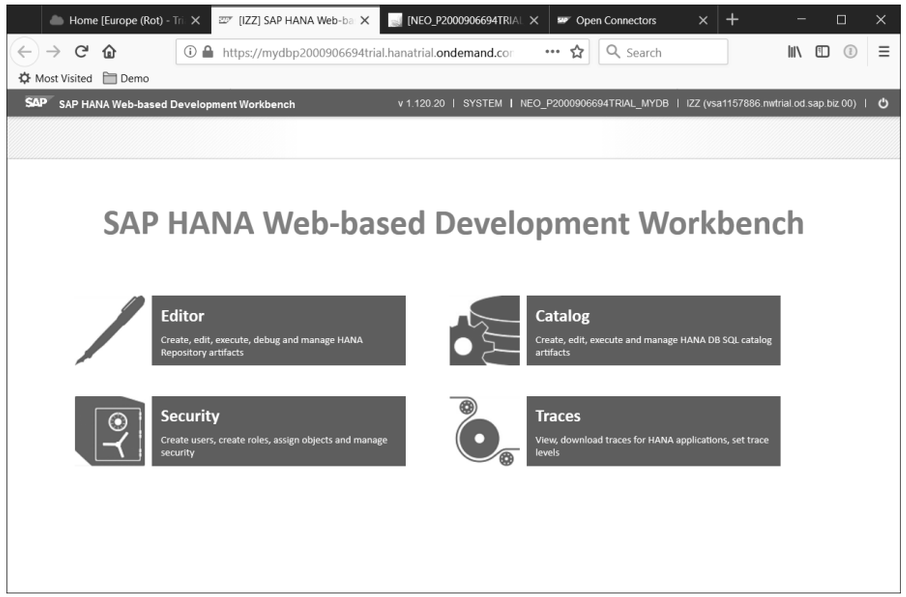


Figure 3.6 SAP HANA Web-Based Development Workbench

Manage Authorization in SAP HANA

The first step is to assign to our SYSTEM user the authorization to access the SAP HANA editor and the SAP HANA catalog:

1. On the main page of the SAP HANA Web-Based Development Workbench, select **Security**, which opens the security administration page where you'll maintain user details and manage user roles.
2. From the left menu, select the **Users** section and look for the SYSTEM user.
3. In the user administration page on the right, select the **Granted Roles** tab and click the **+** button to add an additional role.
4. In the role selection popup window, search for the `sap.hana.xs.ide.roles::Developer` role, as shown in Figure 3.7, and add the role to your user.
5. Save these new settings.

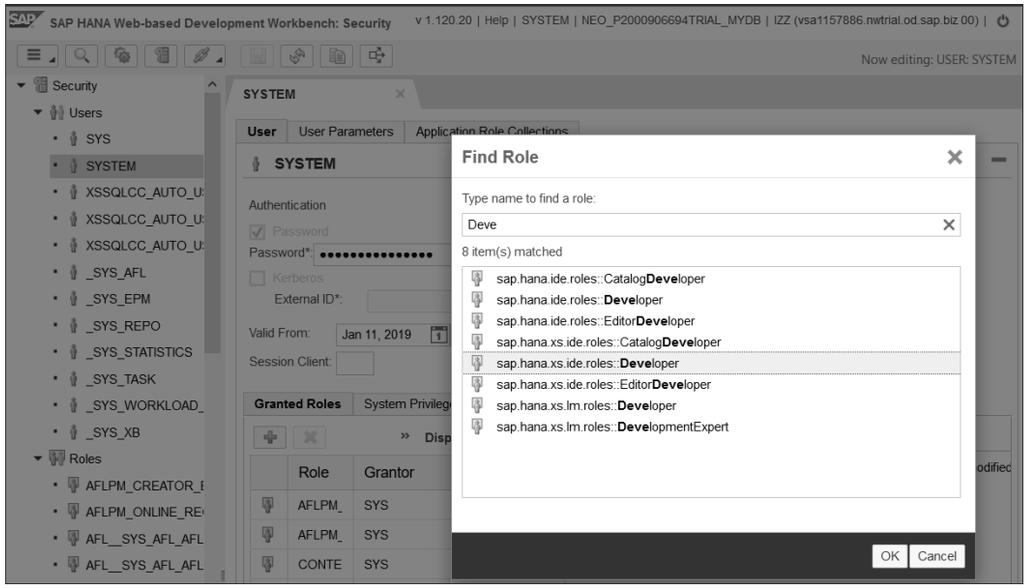


Figure 3.7 Roles Management

The SAP HANA web-based development workbench provides distinct roles specific to each module:

- **SAP HANA editor**
  - `sap.hana.ide.roles::EditorDeveloper`
  - `sap.hana.xs.debugger::Debugger`
- **SAP HANA catalog**
  - `sap.hana.ide.roles::CatalogDeveloper`

- **Security**
  - `sap.hana.ide.roles::SecurityAdmin`
- **Traces**
  - `sap.hana.ide.roles::TraceViewer`

Real-World Tips

In this example, we'll only use the SYSTEM user to perform either administration and developer tasks. In a productive environment, you'll profile different users with different authorizations to act as administrators or as developers.

Create a Database in the Catalog

You have now access to the SAP HANA editor and the catalog. First, create a new SAP HANA table in a new database schema. Open the catalog web development tool from the SAP HANA web-development workbench, which lists the current schemas available in your SAP HANA database. In our case, we'll create a new schema for our developments. Right-click the **Catalog** folder to open the context menu and select **New Schema**, as shown in Figure 3.8, and create a new schema called "MYSCHEMA."

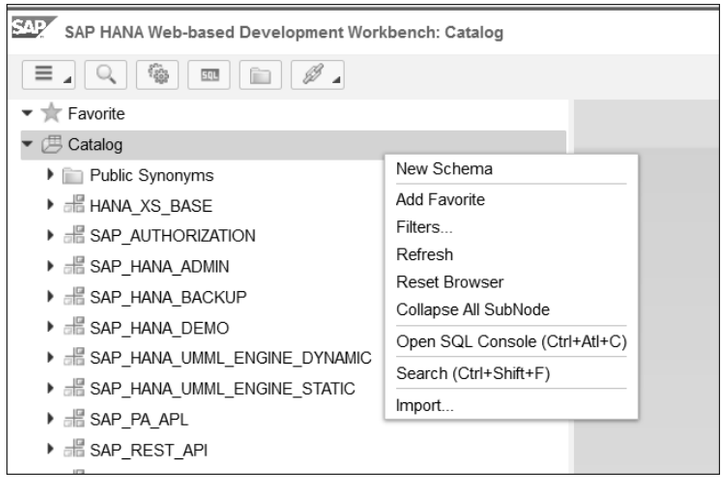


Figure 3.8 Create a New Database Schema

Now that you have your own personal schema, let's look at its structure. Under each schema, different folders group together various SAP HANA SQL artifacts based on their nature, in the following ways:

- *Columnar tables* are database tables that use a columnar structure for storing the data.
- *Functions* enable you to create custom logic or complex queries that are not possible using a graphical view.
- *Indexes* provide the ability to catalogue specific column tables to optimize access to the records they contain.
- *Procedures* are a collection of SQL scripts that can enable complex database logic scripts.
- *Sequences* generate an automatically incremented numeric value based on the definition.
- *Synonyms* are aliases that enable you to refer, for example, to a table that only exists as a catalog object.
- *Tables* are the actual row database tables used to store the data.
- *Triggers* perform actions after specific SQL operations are executed.
- *Views* are virtual representations of collections of one or more tables or views, based on the dynamic results returned in response to an SQL statement.

Columnar and Row Tables

Row tables are the classic tables common in relational databases. These objects store records each as a single row. Columnar databases use a columnar approach to store the data, which allows you to reduce the redundancy of the information.

Which one is better? Both have pros and cons. Row tables are faster for transactional operation, such as inserts, updates, or deletes, but column tables are unrivaled in analytical operations such as data retrieval, aggregations, and calculations. Moreover, columnar tables can compress the data up to 90% more compared to a row table storage with the same structure and data content.

You can select a folder to create the relevant SAP HANA SQL artifact, or you can use the SQL console and SQL statements. We'll choose the latter route: Right-click on **MYSCHEMA** to open the context menu and select **Open SQL Console**, as shown in Figure 3.9.

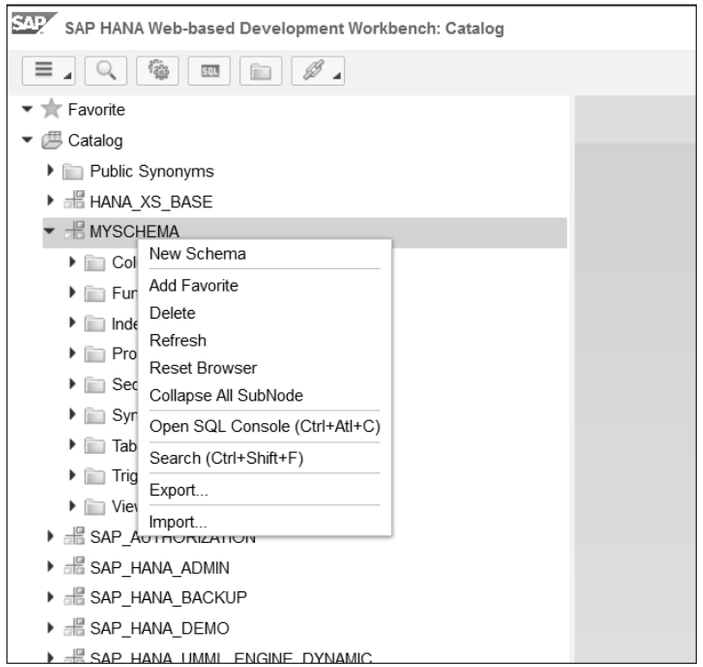


Figure 3.9 Open the SQL Console

In the SQL console, shown in Figure 3.10, you can execute most common SQL statements. Since we want to create a new table, enter the SQL statement shown in Listing 3.1.

```
CREATE COLUMN TABLE "MYSCHEMA"."MATERIAL"(
"MATERIAL" VARCHAR(18) NOT NULL,
"MATERIAL_DESCRIPTION" VARCHAR(40),
"BASE_UNIT" VARCHAR(3),
"MATERIAL_GROUP" VARCHAR(9),
"MATERIAL_GROUP_DESCRIPTION" VARCHAR(20),
"GROSS_WEIGHT" NVARCHAR(17),
"NET_WEIGHT" NVARCHAR(17),
PRIMARY KEY ( "MATERIAL" ) )
UNLOAD PRIORITY 5 AUTO MERGE;
```

Listing 3.1 Create a New Column Table

You should already be familiar with SQL syntax, but, briefly, in Listing 3.1, we're creating a table named `MATERIAL` in the schema `MYSCHEMA` with 7 fields. The field

`MATERIAL` as primary key. A few keywords at the end of the statement—`UNLOAD PRIORITY 5`—expresses the priority used for unloading the table from the SAP HANA memory. The keyword `AUTO MERGE` specifies the modality that SAP HANA database uses to perform the merge process.

Delta Merge

In SAP HANA databases, all write operations are performed on the delta storage table, a separate "copy" of the in-memory database used to store the latest changes to the data's content.

The SAP HANA engine, via a system process called *Mergedog*, checks the changes in this delta table, and based on the configuration of the table, set by default to `AUTO MERGE`, decides when and how to trigger the shift of this new data into the in-memory table.

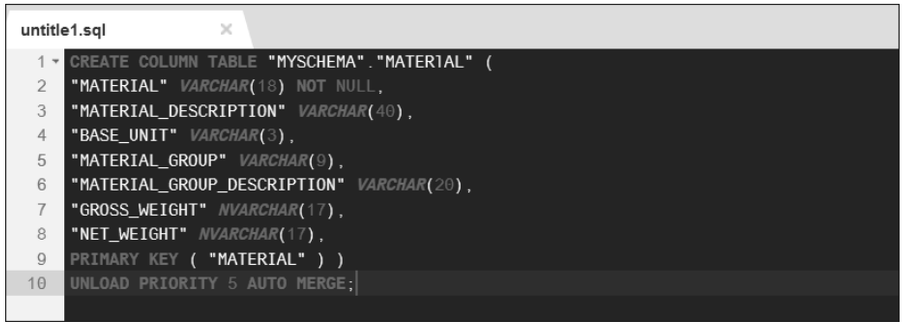


Figure 3.10 SQL Console

Create a New Application

Now, you have a new database table in your SAP HANA Cloud instance, but as you can imagine, this table is completely empty. You can easily create an `INSERT SQL` statement to insert dummy data into the table, but we need an SAP HANA XS application. From the SAP HANA Web-Based Development Workbench, shown in Figure 3.6, select **Editor** to open the SAP HANA editor tools for developing SAP HANA repository objects.

The SAP HANA editor tool looks similar to the SAP HANA catalog tools. As before, you'll create a container, in this case a package, for your development objects called "mypackage." Right-click the catalog to open the context menu and select **New • Package**, as shown in Figure 3.11.

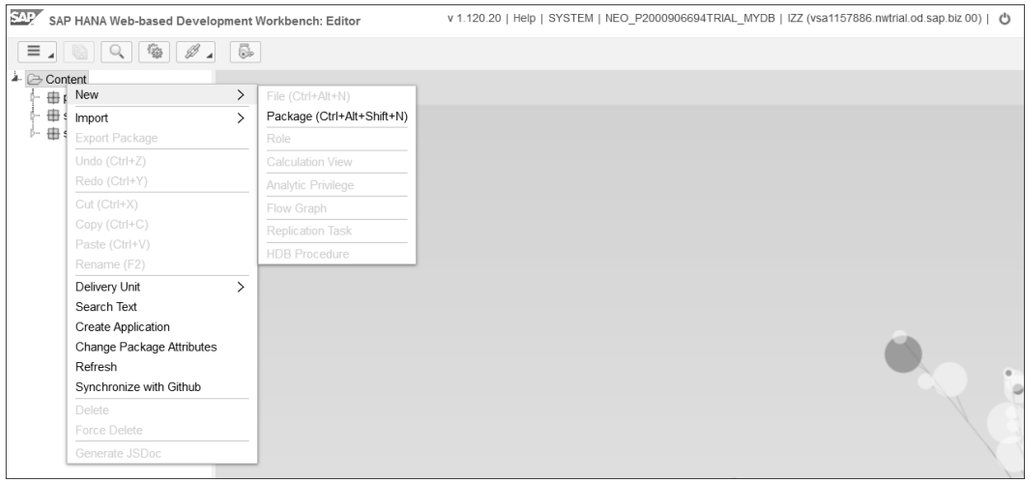


Figure 3.11 Create a New Editor Package

In the next step, you'll create a new SAP HANA XS application by right-clicking the package we just created and selecting **Create Application**, as shown in Figure 3.12.

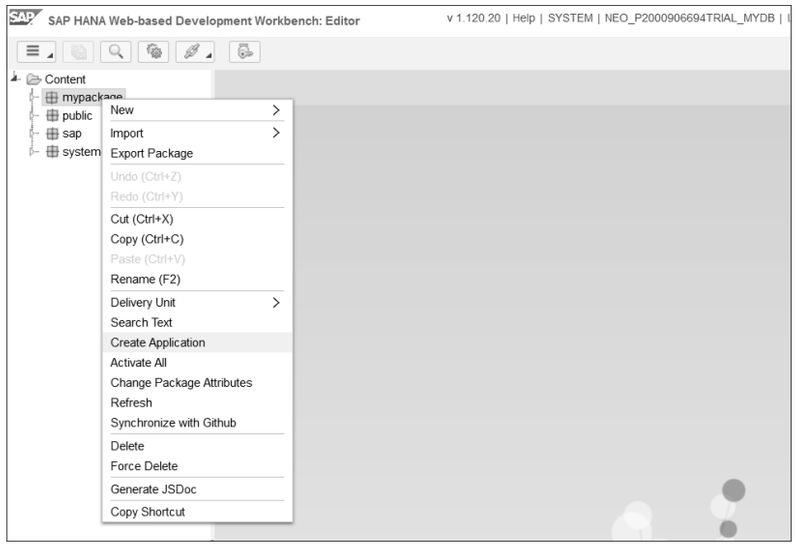


Figure 3.12 Create a New Application

In the popup window, we'll specify that this application is an empty application running in the SAP HANA XS engine that should be included in our package (**mypackage**), as shown in Figure 3.13.

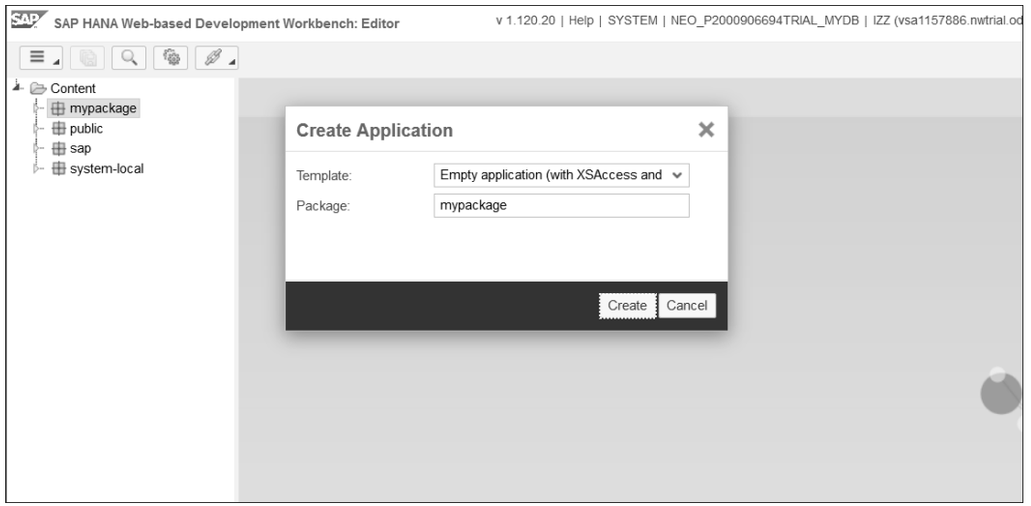


Figure 3.13 SAP HANA XS Blank Application Settings

The blank SAP HANA XS application comes with three objects:

- *.xsaccess* is the application access file where you'll specify who or what is authorized to access the content exposed by an SAP HANA XS application package and what content they are allowed to see.
- *.xsapp* is the application descriptor where you'll define the root folder of an SAP HANA XS application. This file has no content.
- *index.html* is the default page that is called once an SAP HANA XS application is accessed without specifying any specific service.

In addition to these files, you'll need to create a new file that will serve as the OData enabler to access to the SAP HANA database table `MATERIAL`. Right-click on **mypackage** to open the context menu again and select **New • File**, as shown in Figure 3.14.

Call this file "Material.xsodata," as shown in Figure 3.15. The .xsodata extension used to identify OData services created in the SAP HANA XS engine.

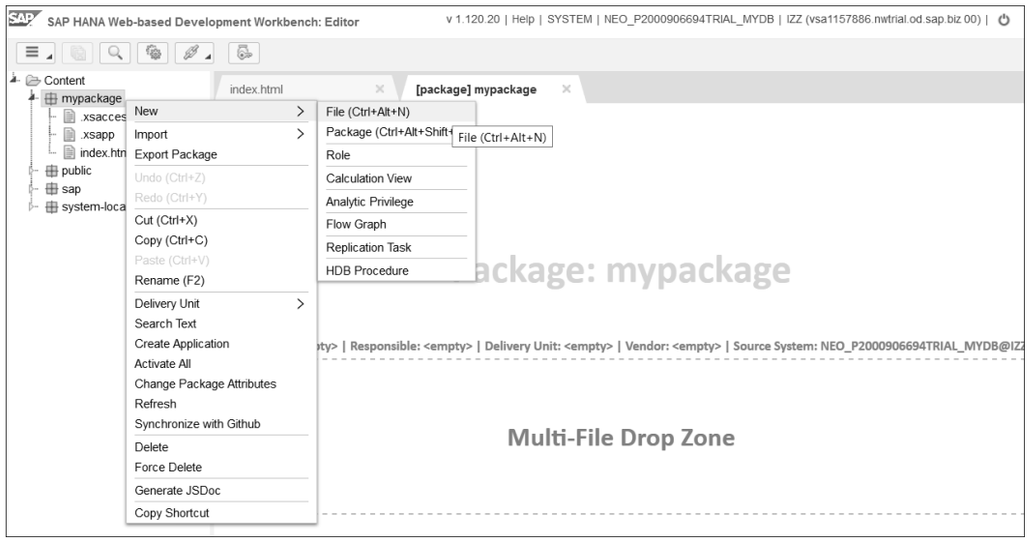


Figure 3.14 Create a New File

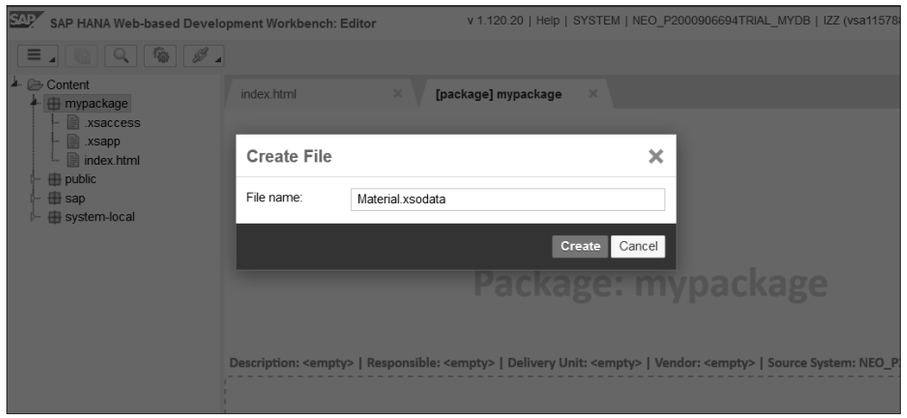


Figure 3.15 Create a .xsodata File

The file is a blank xsodata file, so you'll need to add some logic to tell SAP HANA XS engine the purpose and the source of the OData endpoint. In the file editor shown in Figure 3.16, insert the following snippet:

```
service namespace "Material" {
    "MYSCHEMA"."MATERIAL" as "Material";
}
```


Notice that, with a few lines, you can expose an SAP HANA database table in the OData format using the SAP HANA XS engine.

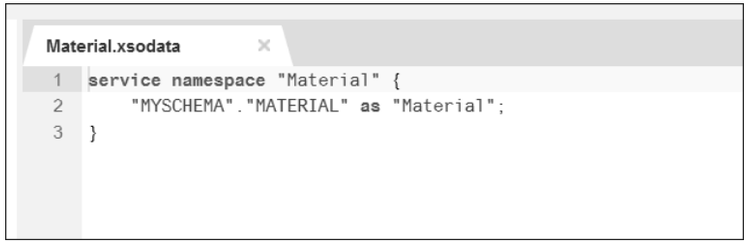


Figure 3.16 Xsodata Editor

Once the file is saved, you can access our OData service from a web browser to see the OData definition, shown in Figure 3.17, which in this case has only one entity, `Material`.

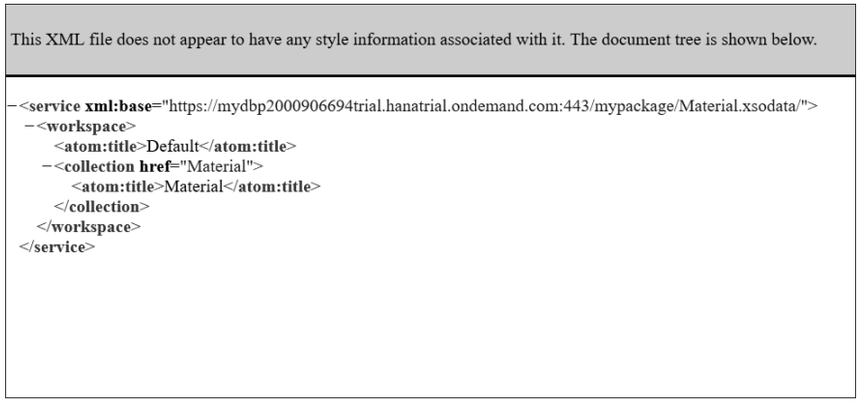


Figure 3.17 OData Service

The SAP HANA editor has an interesting tool that allows you to interact with your database table using a graphical OData explorer. As shown in Figure 3.18, the tool is accessed by right-clicking on the xsodata object and selecting **Open OData Explorer**.

The OData explorer not only offers you the ability to browse the content of a table via an OData service, but you can also generate dummy data to initialize the table and start playing with it. From the toolbar, click on **Generate Data**, and a popup window, shown in Figure 3.19, allows you to define the values and the number of dummy entries you want to create in the SAP HANA table.

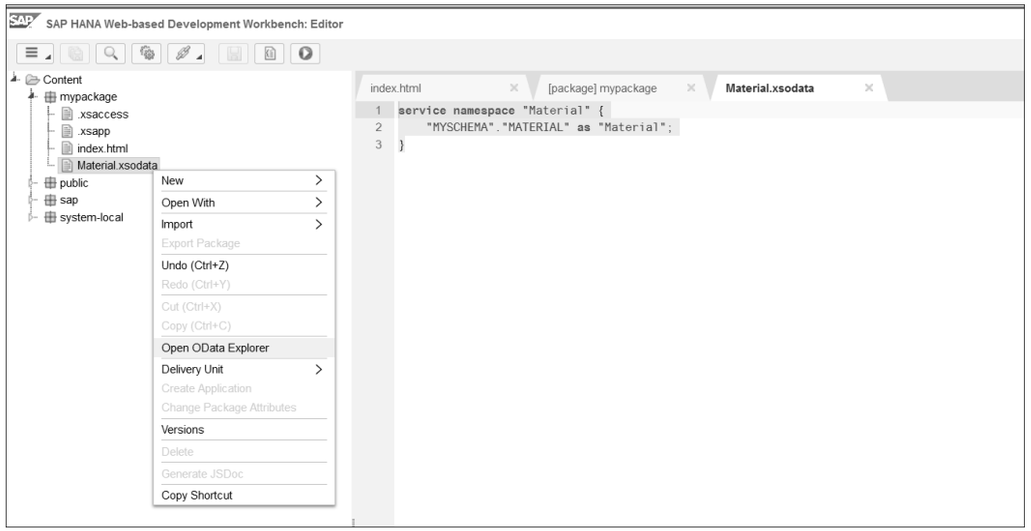


Figure 3.18 OData Explorer

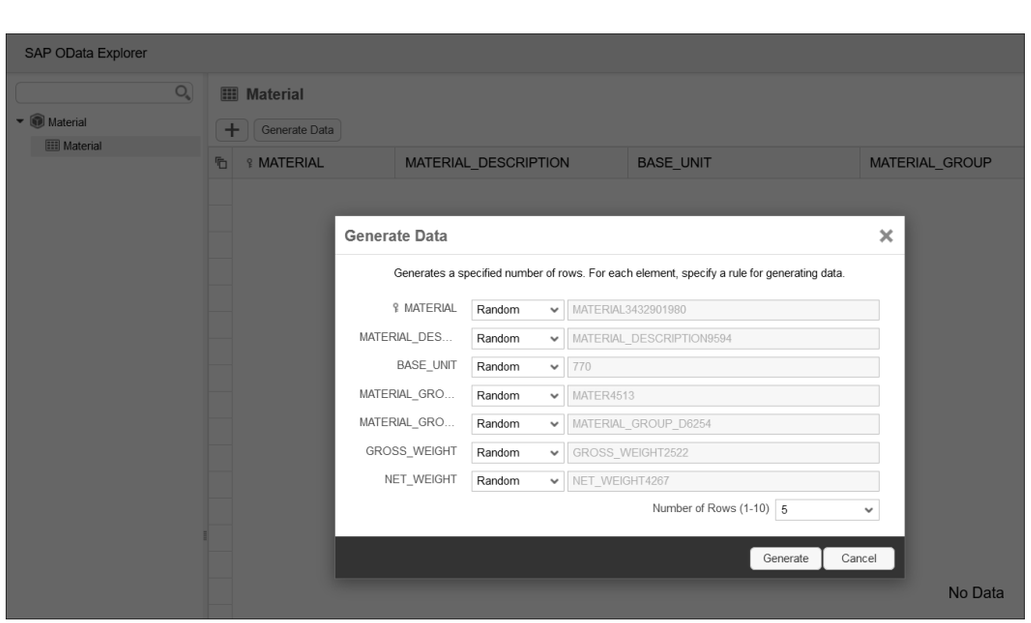


Figure 3.19 Create Dummy Data in the SAP HANA Table

Now that we've arrived at the end of this section, you should be able to create SAP HANA XS applications with the SAP Cloud Platform SDK for the Neo environment.

SAP HANA XS Advanced Development

In this section, we'll focus on the SAP HANA XS Advanced development with the SAP Cloud Platform SDK for the Cloud Foundry environment. The SAP HANA database in Cloud Foundry is based on SAP HANA 2.0, which, as explained in section "Introduction to SAP HANA Development," has substantial differences with SAP HANA 1.0.

This use case will be similar to the one we implemented in the Neo environment. We'll create a database table to store basic information about a material and expose this data using an OData endpoint leveraging the Bring Your Own License (BYOL) capability in the SAP Cloud Platform SDK for the Cloud Foundry environment using Node.js. In this example, we'll see how the development experience in the SAP Cloud Platform SDK for the Cloud Foundry environment can be centralized and simplified using just the SAP Web IDE.

Configure the Environment

To get started, you'll need to perform some configuration steps in the SAP Web IDE. You should already have your Cloud Foundry environment set up, as we did in Chapter 2, but you can check your setup from the **Workspace Preferences** under the **Cloud Foundry** section where the Cloud Foundry builder is installed and enabled; if not, to proceed, click **Install Builder**, as shown in Figure 3.20. Note that, in the trial period, the available services in Cloud Foundry are restricted. For this reason, make sure that your Cloud Foundry environment is active in the Frankfurt (**eu10**) datacenter, that you enable the builder in this datacenter, and that you refer to the trial organization and not the trial beta.

You'll need to complete few more steps to complete the configuration of the SAP Web IDE. You'll need to enable few plugins that allow the SAP Web IDE to develop and support SAP HANA development objects for the SAP HANA XS Advanced engine in Cloud Foundry. The concept of extensions or plugin enables the SAP Web IDE to support multiple development objects.

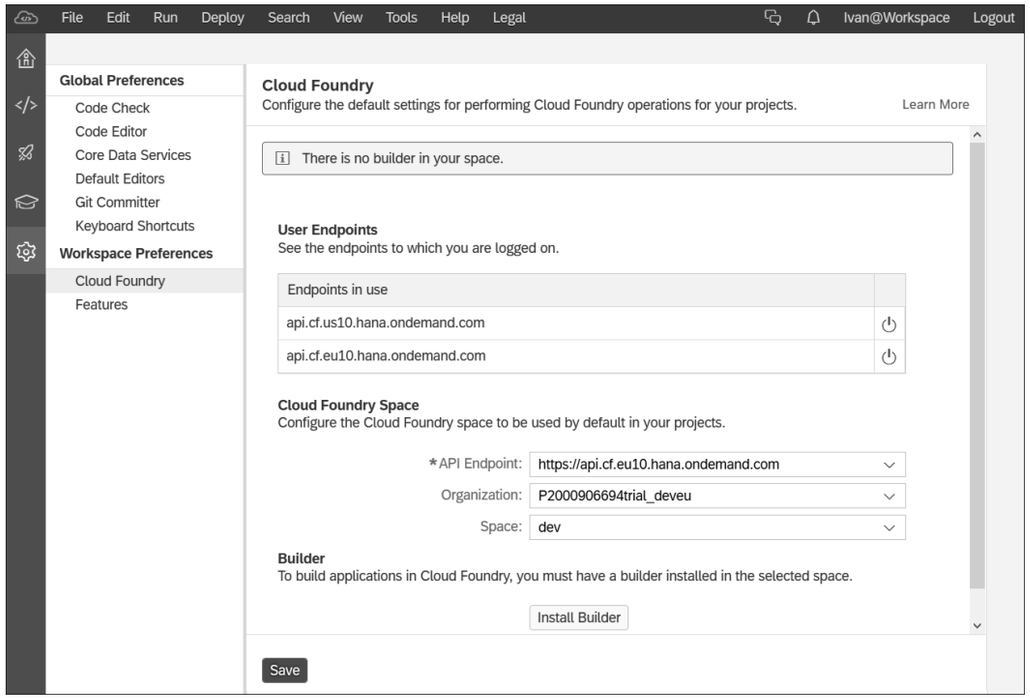


Figure 3.20 Install the SAP Web IDE Cloud Foundry Builder

Under the **Features** section in the **Workspace Preferences**, enable the following extensions:

- *SAP HANA database development tools*, which enables the support for SAP HANA database (HDB) artifacts
- *Tools for Node.js development*, which provides support for Node.js applications
- *SAP HANA database explorer*, which allows you to query information about the database as well as view information about your database's catalog objects.

Once all these extension are enabled, save these settings. When SAP Web IDE restarts, the new extensions should be loaded, as shown in Figure 3.21.

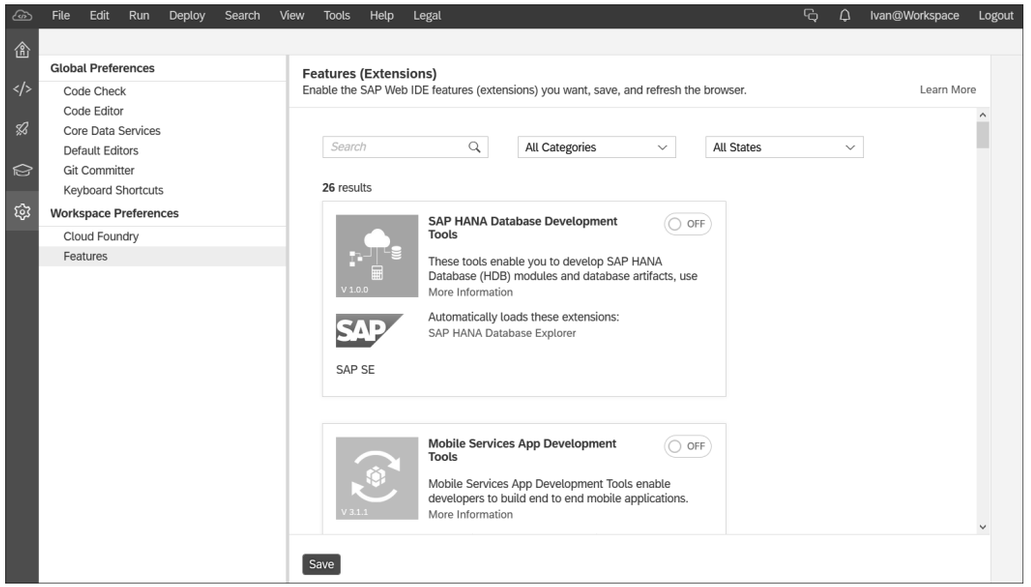


Figure 3.21 Enable SAP HANA Extensions in the SAP Web IDE

Create a New Project

Now, we're ready to start development for SAP HANA XS Advanced:

1. From the **Home** section in the SAP Web IDE, start a new project using the **New Project from Template** functionality.
2. In the Template Selection step, select the **Multi-Target Application** (MTA) template, as shown in Figure 3.22.

In the next sections, provide a project name, in our case, "myxsaapp," and some additional details, as shown in Figure 3.23.

Using the HTML5 Application Repository is disabled by default, but in our case, we'll select this option because we want to use this feature.

HTML5 Application Repository

The HTML5 Application Repository service allows you to manage the lifecycle of your HTML5 applications. During runtime, the repository enables the consuming application, typically the application router, to access static HTML5 application content, such as HTML, CSS, and JavaScript, in a secure and efficient manner.

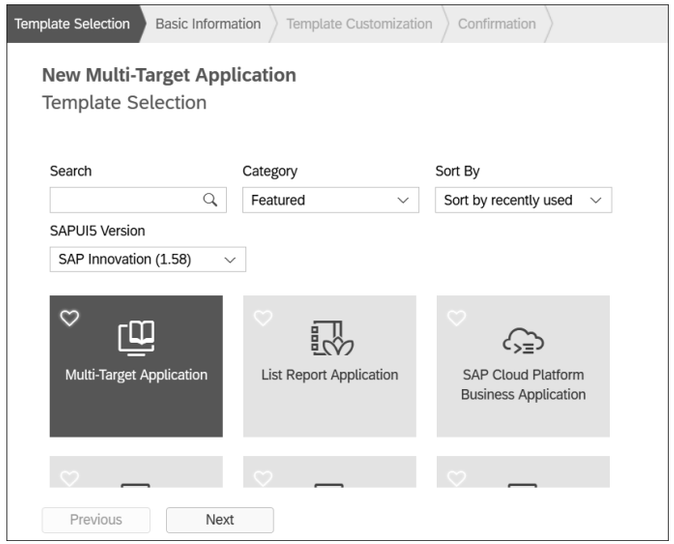


Figure 3.22 New Multi-Target Application Template Selection

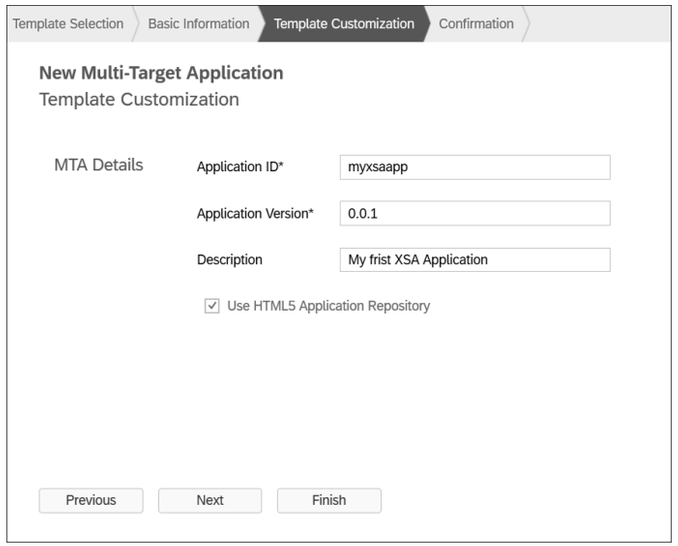


Figure 3.23 Template Customization

In the next step, confirm the creation, and now, you're ready with an empty MTA project.

Create the SAP HANA Database Module

As we learned in Chapter 2, an MTA project is composed of several modules. In our project, we'll start by creating a new SAP HANA database module, which enables us to interact with the SAP HANA database in Cloud Foundry. As in our previous exercise, from the context menu of our MTA, select **New • SAP HANA Database Module** from the available options, as shown in Figure 3.24.

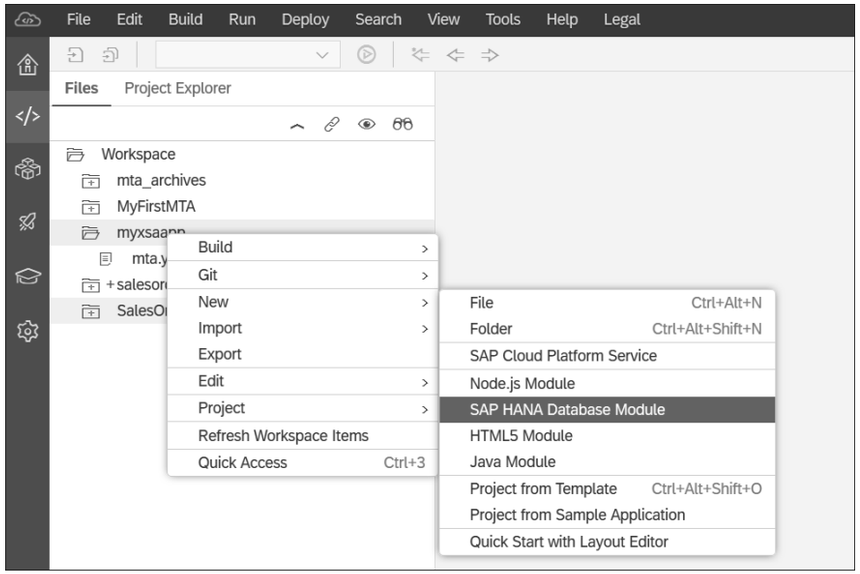


Figure 3.24 Create a New SAP HANA Database Module

As shown in Figure 3.25, provide some basic information, such as the **Module Name** (in our case, "mydb") and then the specific setting for the SAP HANA database artifacts, such as the **Schema Name** and **SAP HANA Database Version**. Note that we selected the *SAP HANA 2.0 SPS 00*, because, as of today, the SAP Cloud Platform trial supports this version. Finally, select the option **Build module after creation**.

Continue confirming the next steps, and the new SAP HANA database module will be created and associated with your MTA project. Because we selected the **Build module after creation** option, the module is immediately built as you can see from the console shown in Figure 3.26.

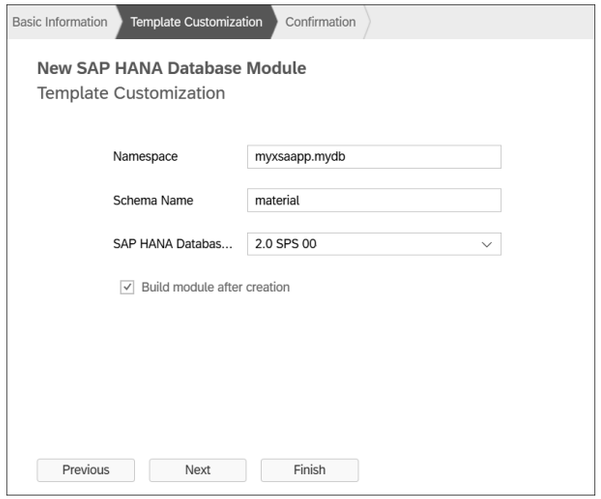


Figure 3.25 Template Customization

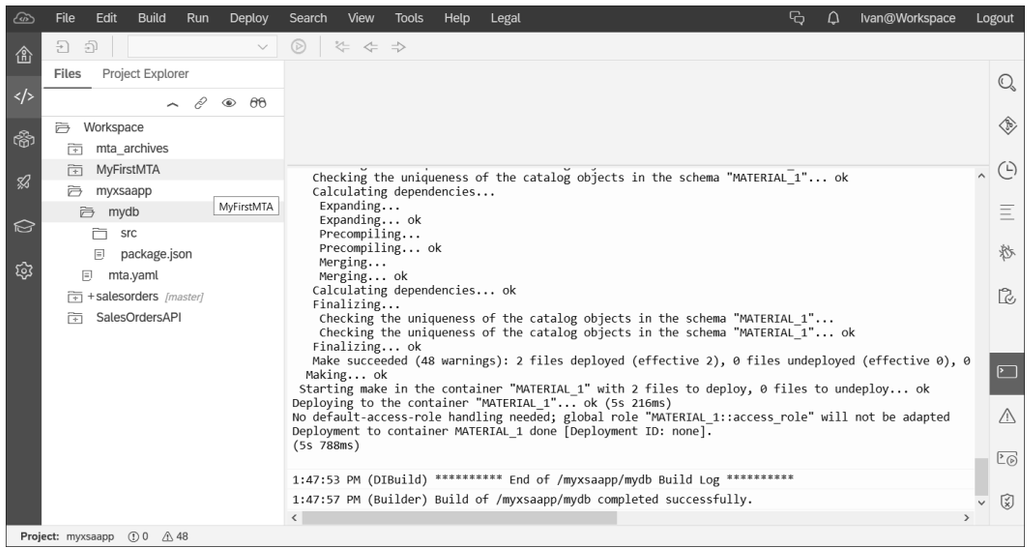


Figure 3.26 SAP HANA Database Module in the SAP Web IDE

You can also verify that the build operation has automatically created a new instance of an SAP HANA database in the Cloud Foundry environment. Switch back to the Could Foundry space in SAP Cloud Platform; notice that a new SAP HANA deployment infrastructure has been created and started, as shown in Figure 3.27.

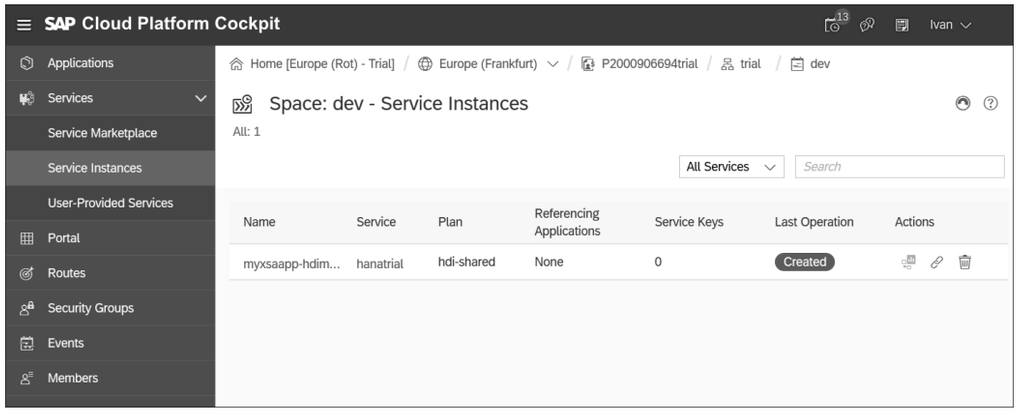


Figure 3.27 SAP HANA Database Instance in an SAP HANA Service in Cloud Foundry

Create the SAP HANA Artifacts

Let's switch back to the SAP Web IDE and start our development creating a new database table in SAP HANA database. In the project tree, right-click on **src** and select **New • HDB CDS Artifact** from the context menu, as shown in Figure 3.28, and call this artifact "cdsArtifact."

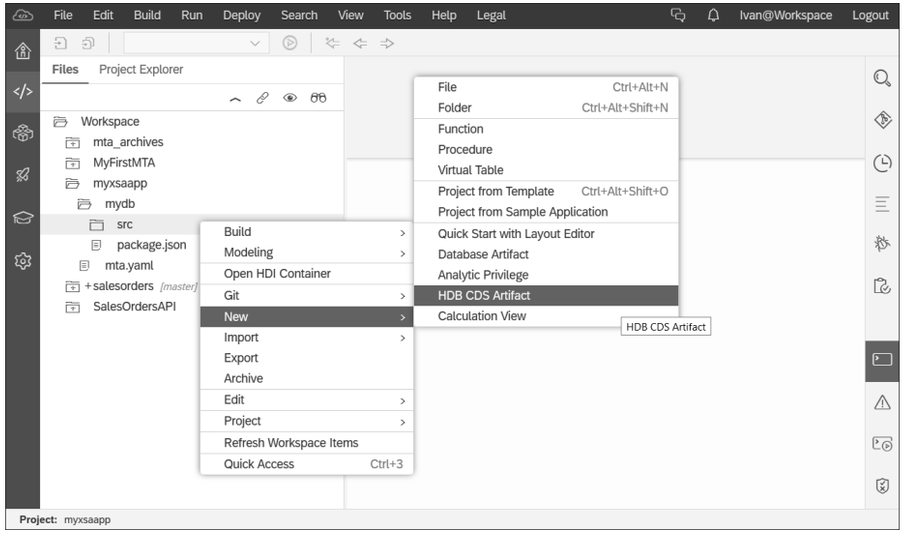


Figure 3.28 Create a New SAP HANA Database Artifact

The SAP Web IDE opens the newly created artifact in the development perspective. From this editor, double-click on the artifact to edit and add the actual database entities, as shown in Figure 3.29.

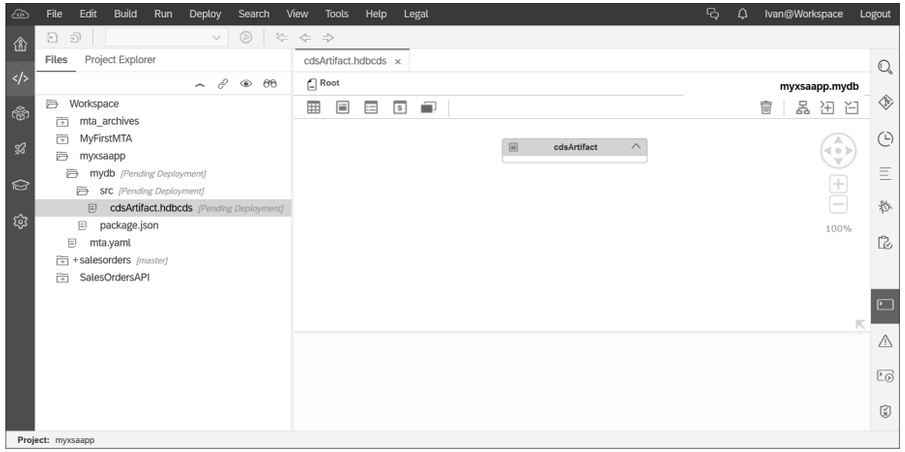


Figure 3.29 SAP HANA CDS Artifact

From the toolbar menu, select create entity button, as shown in Figure 3.30, and call it "MATERIALS."

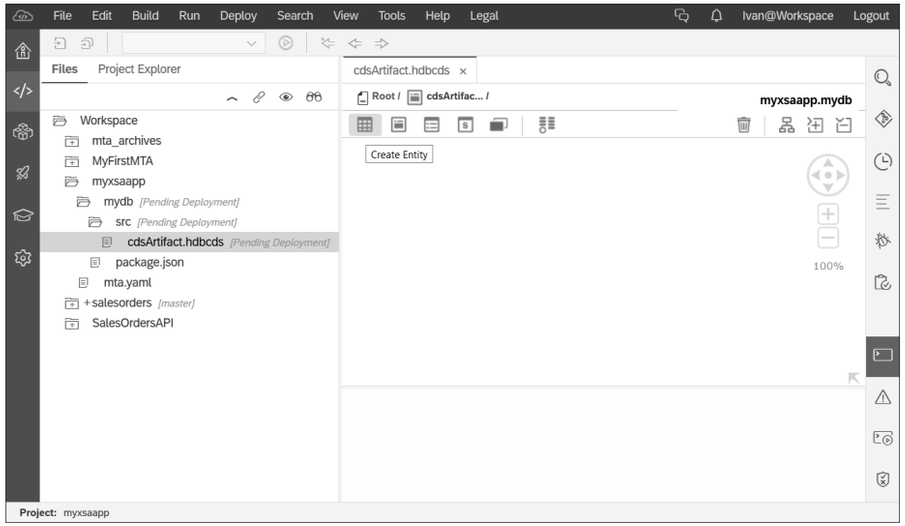


Figure 3.30 Create a New Entity in the SAP HANA Database

Now, double-click on the entity created again to navigate into the entity editor. From this editor, you can create the elements needed in your SAP HANA database table. Figure 3.31 shows the details for each element.

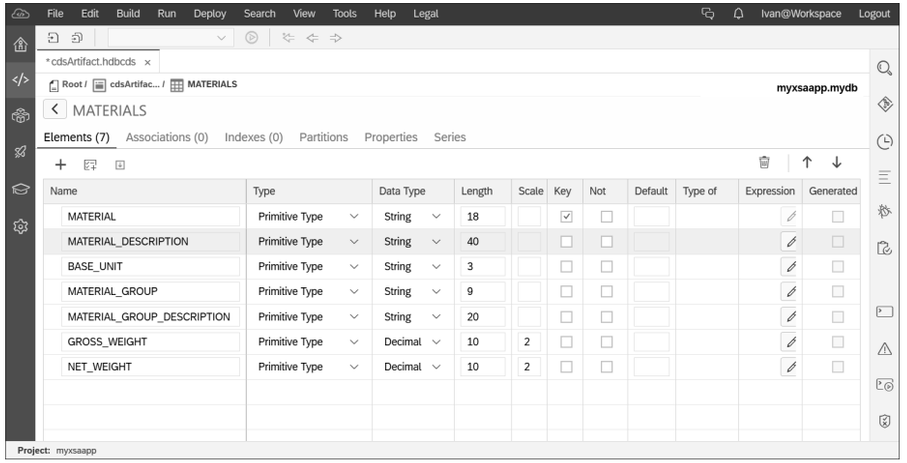


Figure 3.31 Entity Elements Configuration

Our SAP HANA module is now ready for deployment in the SAP HANA Cloud Foundry instance. Right-click on **mydb** and select **Build** • **Build** to trigger the deployment, as shown in Figure 3.32.

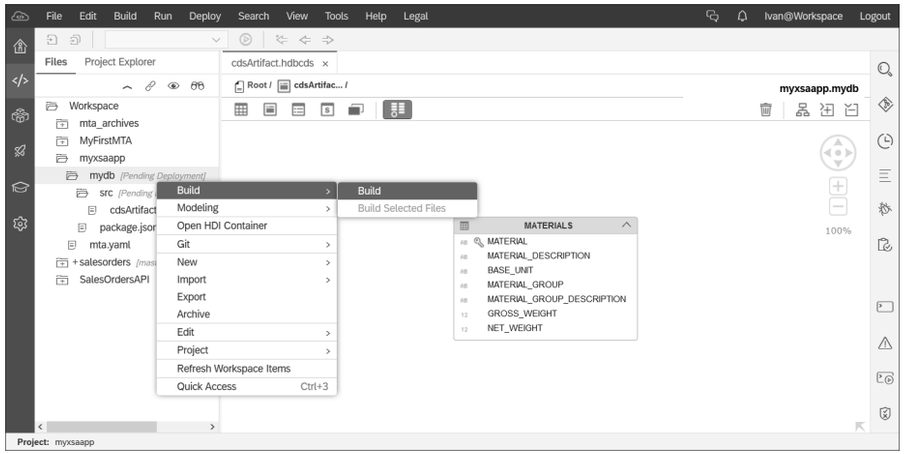


Figure 3.32 Build the SAP HANA Module in Cloud Foundry

The build console log provides a lot of information about the build and deployment processes. After a few seconds, you should see a success message, as shown in Figure 3.33.

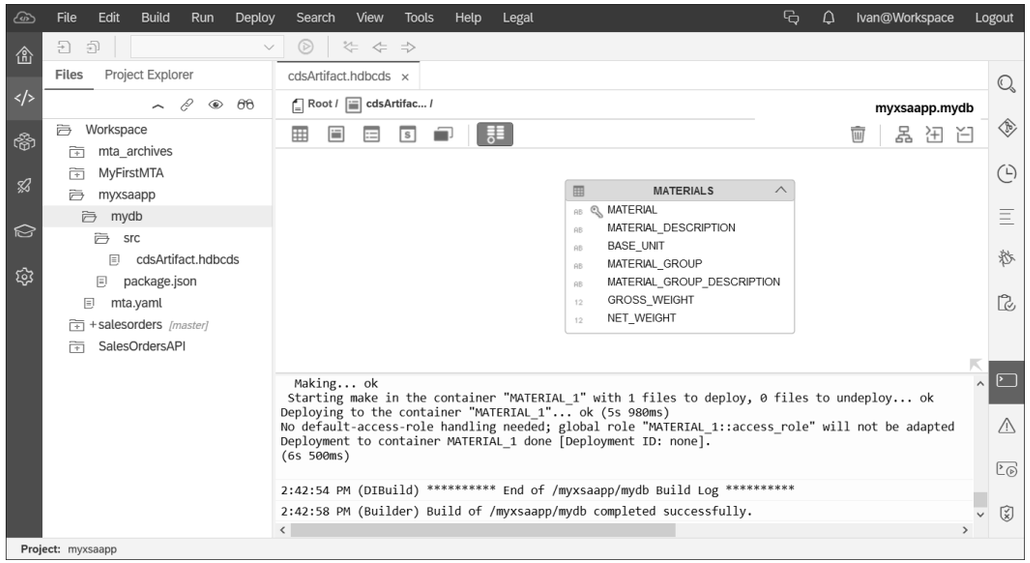


Figure 3.33 Build Completed Successfully

Use the SAP HANA Database Explorer

Before proceeding with the next steps, let's to test our database table and try to insert some data into the SAP HANA table we just created.

As shown in Figure 3.34, right-click on the SAP HANA database module and select **Open HDI Container** to navigate into the SAP HANA database explorer.

Once the SAP HANA database explorer is opened, navigate through the database objects list and select **Tables**. Our newly created database object is available. Right-click on it and select **Generate INSERT Statement**, as shown in Figure 3.35, to initialize the table with some dummy data.

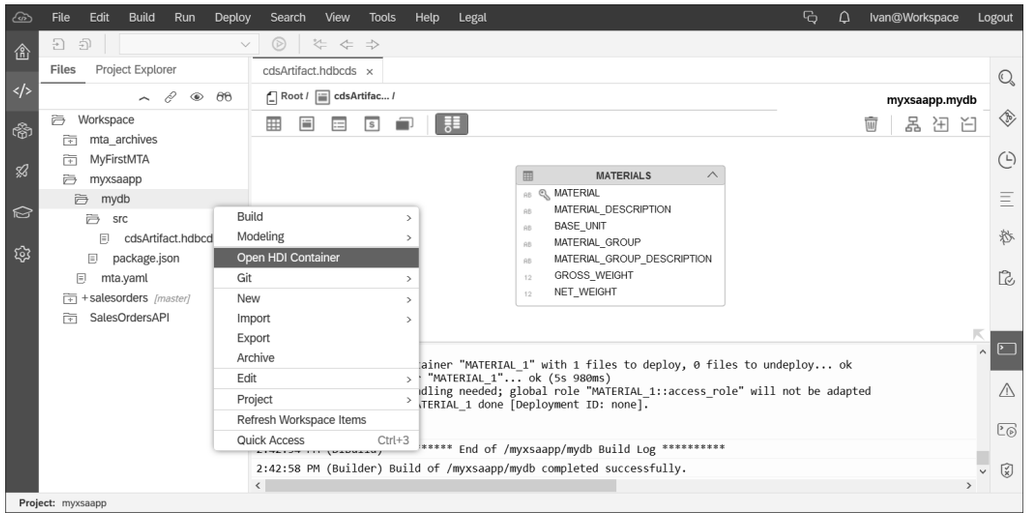


Figure 3.34 Open HDI Container

Figure 3.35 Generate INSERT Statement

In the SQL console, edit the template generated with some dummy values, as shown in Listing 3.2. Once ready, execute the SQL statement by clicking the **Run** button in the toolbar.

```
INSERT INTO "MATERIAL_1"."myxsaapp.mydb::cdsArtifact.MATERIALS" VALUES(
    '123456'/*MATERIAL <NVARCHAR(18)>*/,
    'DUMMY MATERIAL'/*MATERIAL_DESCRIPTION <NVARCHAR(40)>*/,
    'KG'/*BASE_UNIT <NVARCHAR(3)>*/,
    'MATGROUP1'/*MATERIAL_GROUP <NVARCHAR(9)>*/,
    'DUMMY GROUP'/*MATERIAL_GROUP_DESCRIPTION <NVARCHAR(20)>*/,
    20.00 /*GROSS_WEIGHT <DECIMAL>*/,
    16.00 /*NET_WEIGHT <DECIMAL>*/
);
```

Listing 3.2 Insert Statement into SAP HANA Table

The console shows the SQL statement output confirming that a new row has been inserted into the table, as shown in Figure 3.36.

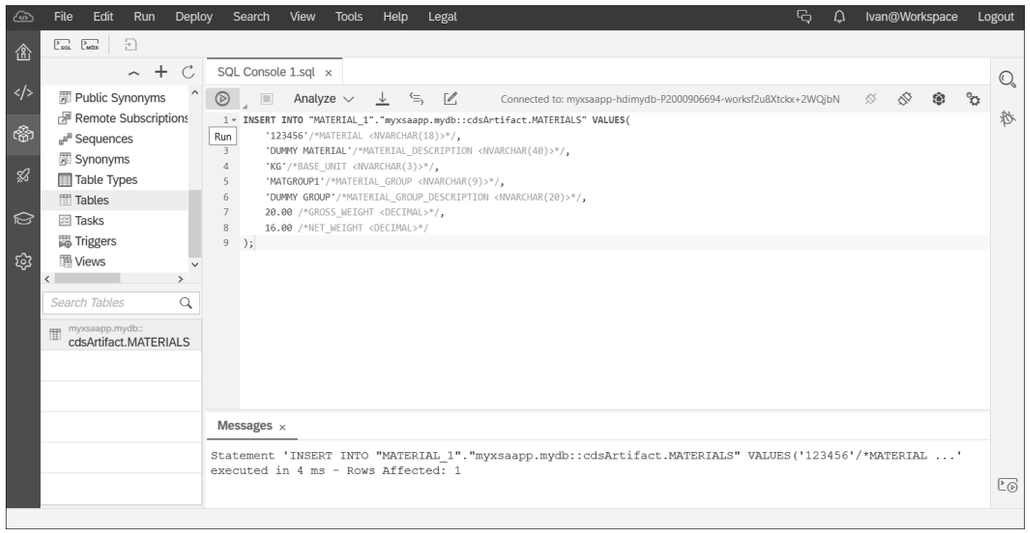


Figure 3.36 Create a New Material in the Table

To verify that the actual row is available in the SAP HANA table, from the context menu of the SAP HANA table, select the **Open Data** function to open the data explorer, as shown in Figure 3.37.

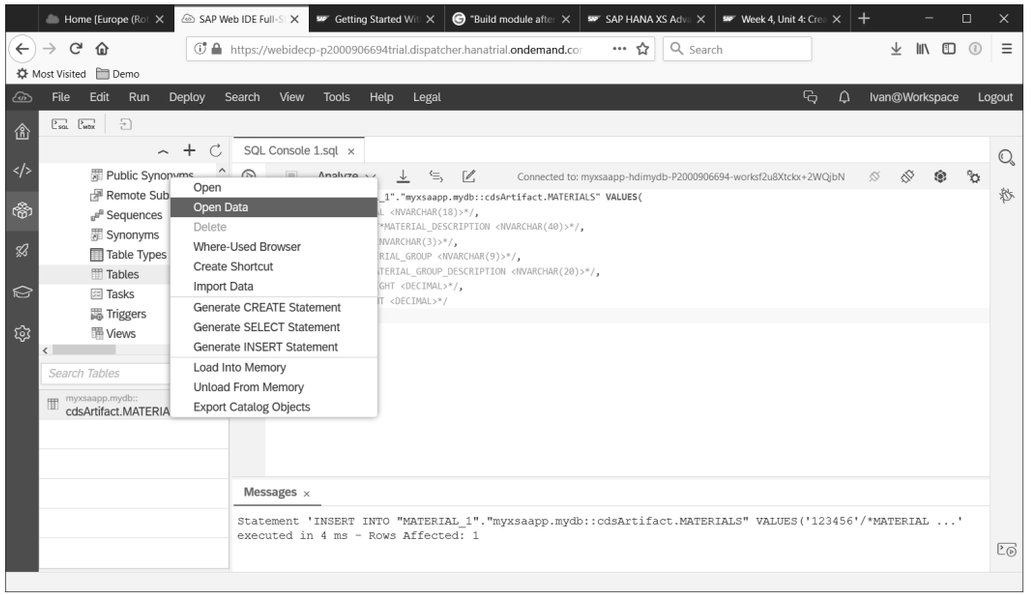


Figure 3.37 Open Data

The raw data explorer gives us a visualization of the actual content in our table, as shown in Figure 3.38, similar to what we might see in a typical SQL client.

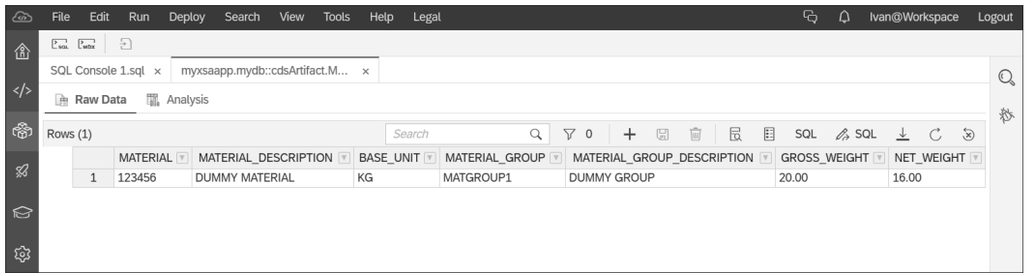


Figure 3.38 Visualize the Content of the Database Table

Create the Node.js Module

In this next step, we'll create a Node.js module to expose our SAP HANA data in an OData format, as we did previously in Section with SAP HANA XS.

Right-click on the SAP Web IDE project **myxsaapp** and select **New • Node.js Module**, as shown in Figure 3.39.

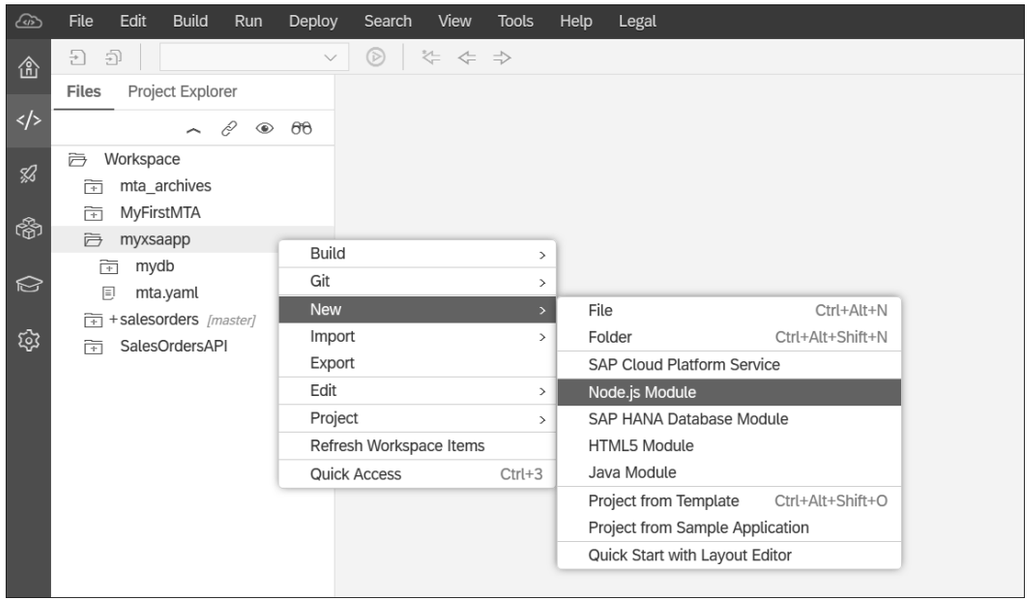


Figure 3.39 Create a New Node.js Module

In the creation wizard, provide a name for the module, such as "myxsjs" and, in the next step, provide a description of the module. Select the **Enable XSJS support** option as well, as shown in Figure 3.40.

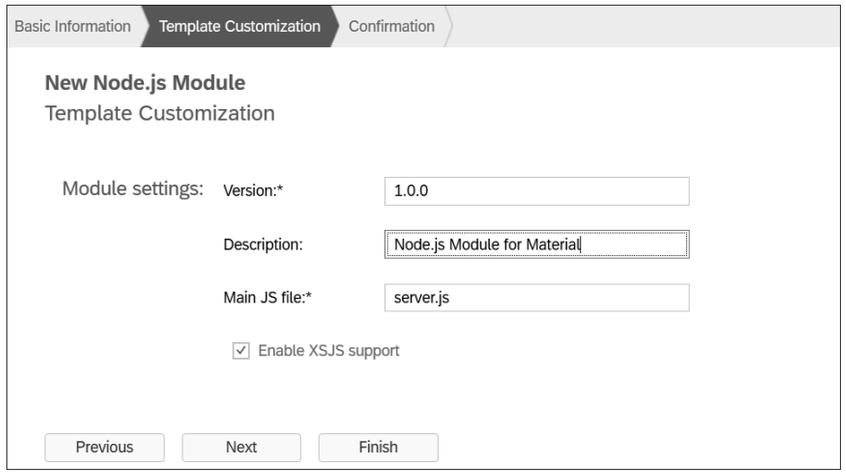


Figure 3.40 Node.js Module Template Customization

We've completed the steps in the wizard, and our new Node.js module has been added to your project. Now, to specify the dependencies of the Node.js module, as shown in Figure 3.41, open your *mta.yaml* file and insert the following dependencies:

- `hdi_mydb`, which refers to the SAP HANA deployment infrastructure
- `mydb`, which refers to the SAP HANA database artifact

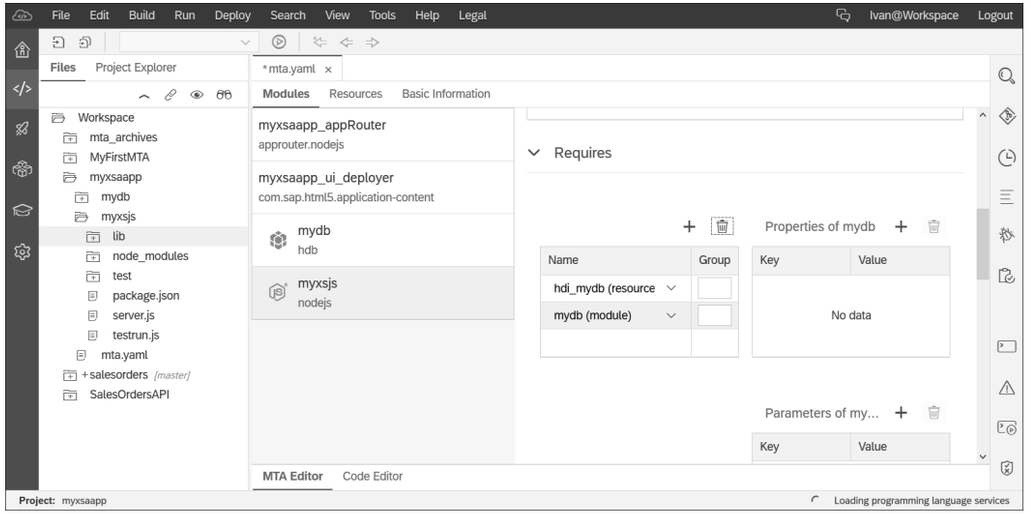


Figure 3.41 Add Dependencies for Node.js Module

Next, you'll need to create the xsodata access for our data. In the **lib** folder under **myxsjs**, create a new subfolder called "xsodata" and a new file called "material.xsodata" by right-clicking the **myxsjs** folder and then selecting the **New** • **File** in the context menu, as shown in Figure 3.42.

The file created is a blank file. As shown in Figure 3.43, you'll need to add the following lines of code for the implementation that allows access to the data using the OData protocol:

```
service {
    "myxsaapp.mydb::cdsArtifact.MATERIALS" as "Material";
}
```

This implementation is rather simple, with a single table, but in this file, you can define multiple associations among entities and define how these entities relate to each other.

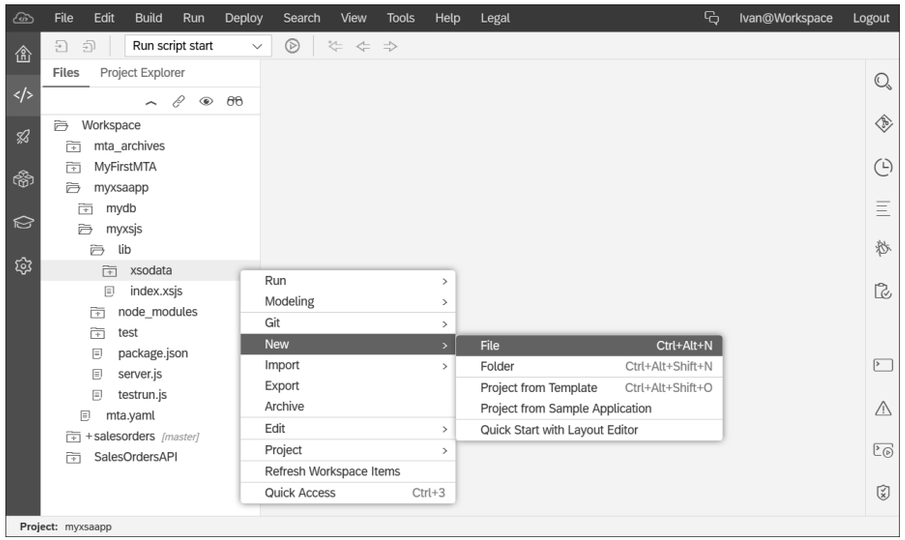


Figure 3.42 Create the xsodata File

```
material.xsodata ×
1  service {
2      "myxsaapp.mydb::cdsArtifact.MATERIALS" as "Material";
3  }
```

Figure 3.43 Configure the xsodata File

We're almost done! The last step requires you deploy the service in the SAP Cloud Platform SDK for the Cloud Foundry environment and test it.

Right-click on the Node.js module (**myxsjs**) and then select **Build**, as shown in Figure 3.44, and then wait for the service to be deployed.

Once the build process is complete, run the service by right-clicking on **myxsjs** and selecting **Run • Run as Node.js Application**, as shown in Figure 3.45.

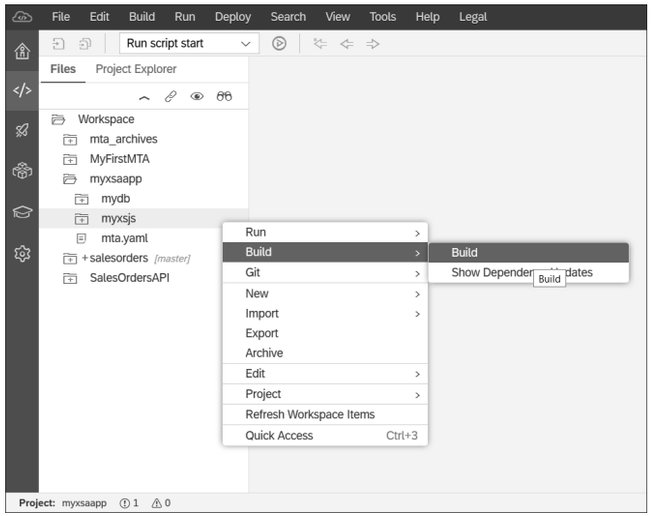


Figure 3.44 Build the Node.js Module

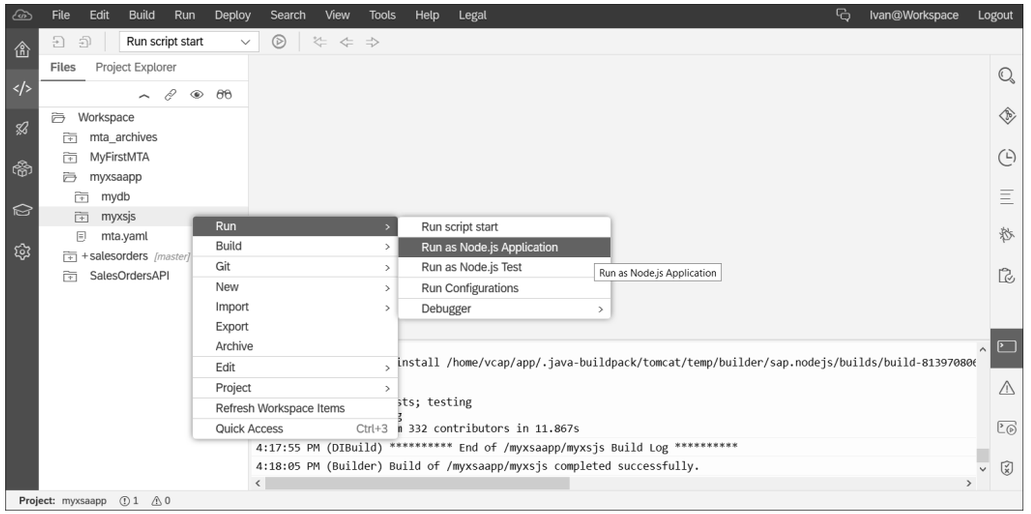


Figure 3.45 Start the Application

The application start takes few seconds, but after this phase, the system should return a success message and provide the address to access your Node.js service, as shown in Figure 3.46.

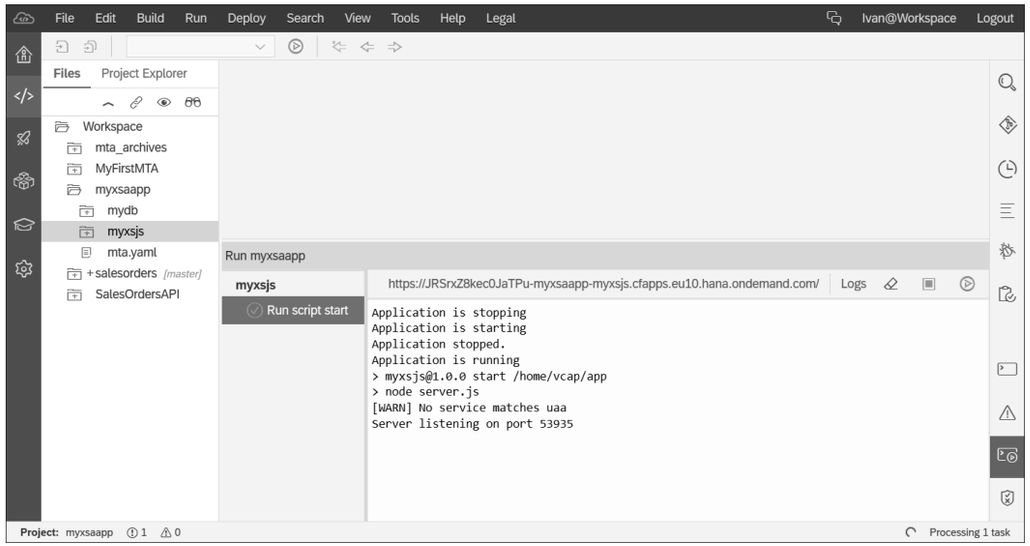


Figure 3.46 Application Running

The same information can also be retrieved from the SAP Cloud Platform cockpit application overview, as shown in Figure 3.47.

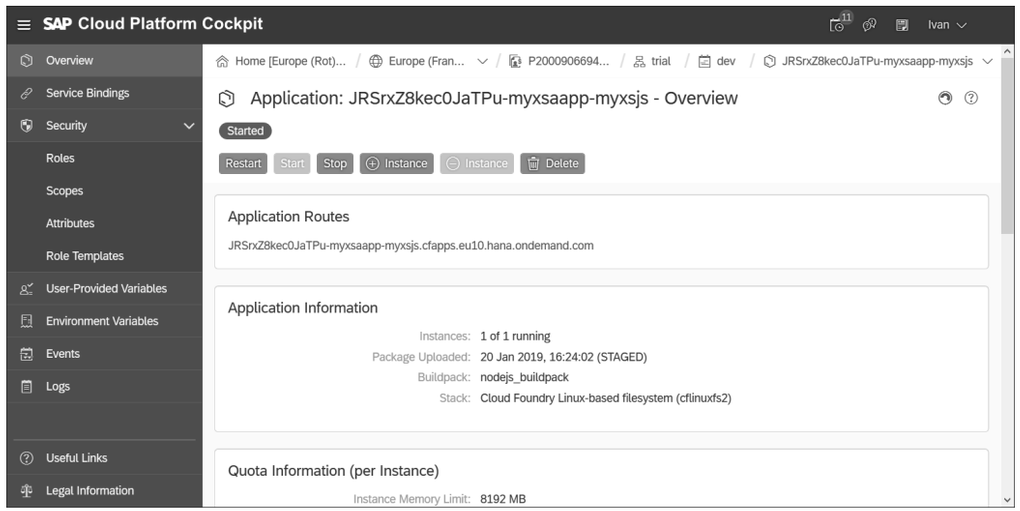


Figure 3.47 Application Overview in Cloud Foundry Cockpit

You can open the application route we just created and access the service. By default, as defined in the server.js file of the Node.js module, the service redirects to the index.xsjs file that outputs a classic *Hello World*. You can consume the xsodata service by manually specifying the route in the browser using the address *https://<myroute>-myxsaapp-myxsjs.cfapps.eu10.hana.ondemand.com/xsodata/material.xsodata?$format=json*. Furthermore, you can navigate to the OData service using the only entity currently available, Material, and see all the available materials in our SAP HANA table, as shown in Figure 3.48.

Figure 3.48 Consume Entity Material Using xsodata

We've concluded our overview of SAP HANA XS Advanced, which will be extremely important in the following chapters because we'll use the table and the OData service to integrate with other services in SAP Cloud Platform.

Many other additions can be considered for this SAP HANA XS Advanced application, for example, implementing security using the User Authentication and Authorization (UAA) service, which is beyond the scope of this book.

Important Terminology

In this chapter, the following terminology was used:

- **SAP HANA core data services (CDS)**
  A layer above the database to define semantically enriched data models.
- **SAP HANA deployment infrastructure (HDI)**
  A service layer of the SAP HANA database that simplifies the deployment of SAP

HANA database artifacts by providing a declarative approach for defining database objects and ensuring a consistent deployment into the database based on a transactional all-or-nothing deployment model and implicit dependency management.

- **Structured query language (SQL)**
  The standard language used to communicate with a database.
- **Open database connectivity (ODBC)**
  A standard application programming, that defines how a client may access a DBMS.
- **Java database connectivity (JDBC)**
  An application programming interface for the Java programming language, that defines how a client may access a DBMS.
- **Database view**
  A database object that defines a query. A view can be accessed as a database table, but it doesn't store any data.
- **Scalable software**
  The performances and overall system throughput continues to improve as more computing resources are made available for its use.
- **In-Memory database**
  A DBMS that keeps the entire contents of a database or table available in computer memory at all times while the database is opened.
- **Inner join**
  A database operation between two tables where all the records satisfying the join condition (typically expressed in the ON clause).
- **Left join**
  A database operation between two tables where all the records from the left table and only the matching records of right table are returned.
- **Right join**
  A database operation between two tables where all the records from the right table and only the matching records of left table are returned.
- **Full join**
  A database operation between two tables where all the records from both the tables are returned.
- **TomEE**
  The enterprise edition of the Apache Tomcat web server.

- **FastCGI**
  A binary protocol for interacting web applications to a web server.
- **Software development kit (SDK)**
  A set of software development tools that are used to develop an application in a specific hardware or software context.
- **Database schema**
  The layout or blueprint of a database that outlines the way data is organized into database artifacts.
- **Column-oriented DBMS**
  A DBMS that stores data tables by column rather than by row. It provides a faster access to the data and a optimization of the space used by the data.

Practice Questions

These practice questions will help you evaluate your understanding of the topics covered in this chapter. The questions shown are similar in nature to those found on the certification examination. Although none of these questions will be found on the exam itself, they will allow you to review your knowledge of the subject.

Select the correct answers and then check the completeness of your answers in the Practice Question Answers and Explanations section. Remember that, on the exam, you must select all correct answers and only correct answers to receive credit for the question.

1. What is the purpose of SHINE in SAP HANA extended application services (SAP HANA XS)?

☐ **A.** The SAP HANA integration network engine is used to control the connectivity of the SAP HANA engine.

☐ **B.** SAP HANA intelligent enterprise provides machine learning capabilities in the SAP HANA engine, which can be used to enable the intelligent enterprise.

☐ **C.** SAP HANA interactive education is a collection of prebuilt scenarios in SAP HANA for learning purposes.

☐ **D.** SAP HANA interactive node explore allows developers navigate across available SAP HANA artifacts and check their dependencies.

2. What of the following is *not* a reason for creating with SAP HANA extended application services (SAP HANA XS)?

☐ **A.** Embedding full-featured application servers, web servers, and development environments within the SAP HANA appliance itself

☐ **B.** Reducing the total cost of ownership (TCO) by simplifying the architecture

☐ **C.** Embracing the code pushdown paradigm and improving the efficiency of application development in all its aspects

☐ **D.** Completely replacing ABAP development in the business suite to keep the core clean

3. Which of the following is not a module of the SAP HANA Web-Based Development Workbench?

☐ **A.** Admin

☐ **B.** Editor

☐ **C.** Catalog

☐ **D.** Security

☐ **E.** Trace

4. Which statement correctly describe the SAP HANA deployment infrastructure?

☐ **A.** SAP HANA deployment infrastructure provides a Git source control repository embedded in the SAP HANA database.

☐ **B.** SAP HANA deployment infrastructure simplifies the deployment of SAP HANA database artifacts.

☐ **C.** SAP HANA deployment infrastructure is a service available in SAP Cloud Platform that provides connectivity for developing different applications.

☐ **D.** SAP HANA deployment infrastructure supports the deployment of database, JavaScript, and xsodata artifacts in the SAP Cloud Platform SDK for the Cloud Foundry environment.

☐ **E.** SAP HANA deployment infrastructure is supported in SAP HANA XS and has been then integrated into SAP HANA XS Advanced for backward compatibility.

5. What are features in the SAP Web IDE?

☐ **A.** Features are extensions that you can enable to support multiple features and development objects in the SAP Web IDE.

☐ **B.** Features are tools that provide support to the SAP Web IDE in Cloud Foundry.

☐ **C.** Features are settings that that you can adjust to personalize the SAP Web IDE's look and feel to improve the developer experience.

☐ **D.** Features are a collection of tools that simplify the development of any kind of development object with the usage of templates provided by SAP.

6. Which of the following are part of the architecture of SAP HANA XS Advanced?

☐ **A.** SAP Web IDE for SAP HANA

☐ **B.** SAP HANA web-based development workbench

☐ **C.** SAP HANA database

☐ **D.** SAP HANA cockpit

☐ **E.** External Git repository

7. When creating a web application using SAPUI5 in the SAP HANA XS engine, which library would be used for mobile and desktop usage?

☐ **A.** `sap.ui`

☐ **B.** `sap.mobile`

☐ **C.** `sap.adaptive`

☐ **D.** `sap.m`

☐ **E.** `sap.ui.flex`

8. Which of the following extensions did we activate to develop an SAP HANA XS Advanced application?

☐ **A.** SAP HANA database development tools

☐ **B.** OData model editor

☐ **C.** Tool for Node.js development

☐ **D.** SAP HANA database explorer

☐ **E.** SAP Cloud Platform services

9. What is the main purpose of the SQL console in SAP HANA XS?

☐ **A.** Execute read-only SQL statements
☐ **B.** Create tables, but not for querying tables, because we need an OData endpoint
☐ **C.** Execute most common SQL statements
☐ **D.** Perform advanced SQL statements to delete data from SAP HANA tables
☐ **E.** Connect to the SAP S/4HANA database and query the native tables

10. Which of the following database objects are available in SAP HANA XS Advanced when using the SAP HANA database explorer in the SAP Web IDE?

☐ **A.** Column views
☐ **B.** Sequences
☐ **C.** User roles
☐ **D.** Indexes
☐ **E.** Node.js

11. What is the role of association in a core data services (CDS)?

☐ **A.** Same as an SQL join operation
☐ **B.** Specify a constraint in the CDS
☐ **C.** Move application logic into the database
☐ **D.** Define a relationship between the data entities

12. Which tool in the SAP HANA web-based development workbench allows you to create an SAP HANA XS application?

☐ **A.** SAP HANA editor
☐ **B.** SAP HANA catalog
☐ **C.** Security
☐ **D.** SAP Web IDE
☐ **E.** Traces

13. Which of the following advantages does the SAP HANA deployment infrastructure provide?

☐ **A.** Allows concurrent deployment of different versions of the same application
☐ **B.** Provides version control and lifecycle management services
☐ **C.** Supports database development objects, JavaScript, xsodata, or other application-layer artifacts
☐ **D.** Provides a declarative approach for defining database objects

14. From the database overview page in the SAP Cloud Platform SDK for the Neo environment, you can which of the following tools?

☐ **A.** SAP HANA cockpit
☐ **B.** SAP HANA Web-Based Development Workbench
☐ **C.** SAP Web IDE
☐ **D.** SAP HANA Interactive Education (SHINE)
☐ **E.** None of the above

15. Which module type can be included in an MTA project to create an SAP HANA Deployment Infrastructure container?

☐ **A.** Java module
☐ **B.** Node.js module
☐ **C.** HTML5 module
☐ **D.** SAP HANA database module

Practice Question Answers and Explanations

1. Correct answer: **C**
SAP HANA Interactive Education (SHINE) for the SAP Cloud Platform provides educational content that can be deployed in your SAP HANA instance that includes SAP-created scenarios that you can use for learning.

2. Correct answer: **D**
   ABAP development are not deprecated or denied, but the new development paradigm controls how these must be performed to guarantee the possibility to do not jeopardize system upgrades.
3. Correct answer: **A**
   Admin is not a module of the SAP HANA Web-Based Development Workbench. Admin features are provided by the SAP HANA cockpit, which is a separate application in SAP HANA.
4. Correct answer: **B**
   The SAP HANA deployment infrastructure is a service layer of the SAP HANA database that simplifies the deployment of SAP HANA database artifacts by providing a declarative approach for defining database objects and ensuring a consistent deployment into the database based on a transactional all-or-nothing deployment model and implicit dependency management.
5. Correct answer: **A**
   Features in SAP Web IDE are also called extensions and provide support to different development artifacts and tools directly from the SAP Web IDE.
6. Correct answers: **A, C, E**
   SAP HANA web-based development workbench and SAP HANA cockpit are not available in SAP HANA XS Advanced, but are part of SAP HANA XS.
7. Correct answer: **D**
   The main library sap.m (formerly mobile) collects a set of components that are mobile ready and are responsive to the device that is used (e.g., smartphone, tablet, desktop) delivering the same user experience across all devices. The desktop library sap.ui has been used as the "desktop" experience for SAPUI5 applications. These components are designed for use on larger screens and do not adapt to the device being used.
8. Correct answer: **A, C, D**
   SAP HANA database development tools allows you to develop and deploy SAP HANA database artifacts into an SAP HANA database. SAP HANA database explorer is used to query information about the database, as well as to view information about the database's catalog objects. Tool for Node.js development allows you to develop and deploy Node.js modules in an MTA project. SAP Cloud Platform services and the OData model editor are valid SAP Web IDE plugins, but we didn't use them to develop our SAP HANA XS Advanced application.

9. Correct answer: **C**
   In the SQL console, you can execute most common SQL statements. It is available from the context menu or in the toolbar of the SAP HANA web-based development workbench.
10. Correct answers: **A, B, D**
    SAP HANA deployment infrastructure manages only database artifacts. In this case, Node.js and user roles are not part of the database artifacts library. Node.js is part of the SAP HANA XS engine but managed by the Node.js module; user roles are managed in the service panel in the SAP Cloud Platform SDK for the Cloud Foundry environment.
11. Correct answer: **D**
    Associations are used to define relationships between data entities in the CDS model. Associations create an easy way to navigate from one entity to another that is triggered only when needed. A join statement combines data between two or more entities and is performed even if the additional information is not used in the application.
12. Correct answer: **A**
    The SAP HANA editor is used to create, edit, debug, and manage SAP HANA repository artifacts. The SAP HANA catalog is designed to create, edit, debug, and manage SAP HANA SQL artifacts. Security aims to maintain user access roles to the SAP HANA instance. Traces allows you to maintain tracing settings and to monitor your SAP HANA applications. The SAP Web IDE is not part of the SAP HANA Web-Based Development Workbench.
13. Correct answers: **A, D**
    The goal of SAP HANA deployment infrastructure is to manage only database artifacts from design time, but in a way that allows multiple versions of the same core objects to be used on the same SAP HANA database at the same time. SAP HANA deployment infrastructure introduces the concept of the container as an abstraction of the schema. The container, in turn, dynamically generates the schema, a container-specific database user who owns all objects, and a password for that database user. Version control and lifecycle management are not provided by SAP HANA deployment infrastructure , but they are provided by Git.
14. Correct answers: **A, B, D**
    The SAP HANA database overview provides a quick look at your database's details, such as its version, its configuration, and the JDBC URL as well as quick access to the SAP HANA Cockpit, which provides an administrative overview of

the SAP HANA instance; the SAP HANA Web-Based Development Workbench, which enables SAP HANA developers create tables and SAP HANA XS applications; and SAP HANA Interactive Education (SHINE), which provides educational content (prebuilt scenarios) that you can deploy in your SAP HANA instance.

15. Correct answer: **D**
    The SAP HANA database module is a collection of related design-time database artifacts, such as data models, views, or procedures. When the build commend is triggered in SAP Web IDE, it generates an SAP HANA deployment infrastructure container service instance in the SAP Cloud Platform Cloud Foundry space configured in the SAP Web IDE.

Takeaways

At this point, you should have a good understanding of SAP HANA's development capabilities in SAP Cloud Platform. You learned to create SAP HANA XS and SAP HANA XS Advanced applications in SAP Cloud Platform, in both the Neo and Cloud Foundry environments.

You can now develop an entire application using the extended application services, and the SAP Web IDE as a single development environment in the cloud.

You also understand how to deploy an SAP HANA XS application in the cloud, and as mentioned earlier, you can use these same skills to understand on-premise SAP HANA development.

Summary

Now, with an understanding of SAP HANA development and of how to use SAP HANA's in-memory capabilities, you can create new cloud-based applications in conjunction with other services in SAP Cloud Platform. The knowledge you gained in this chapter enables the knowledge you'll learn in future chapters.

In the next chapter, we'll focus on extending both on-premise and cloud-based SAP products using SAP Cloud Platform so you can continuously innovate while keeping the core of your ERP clean.

Contents

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

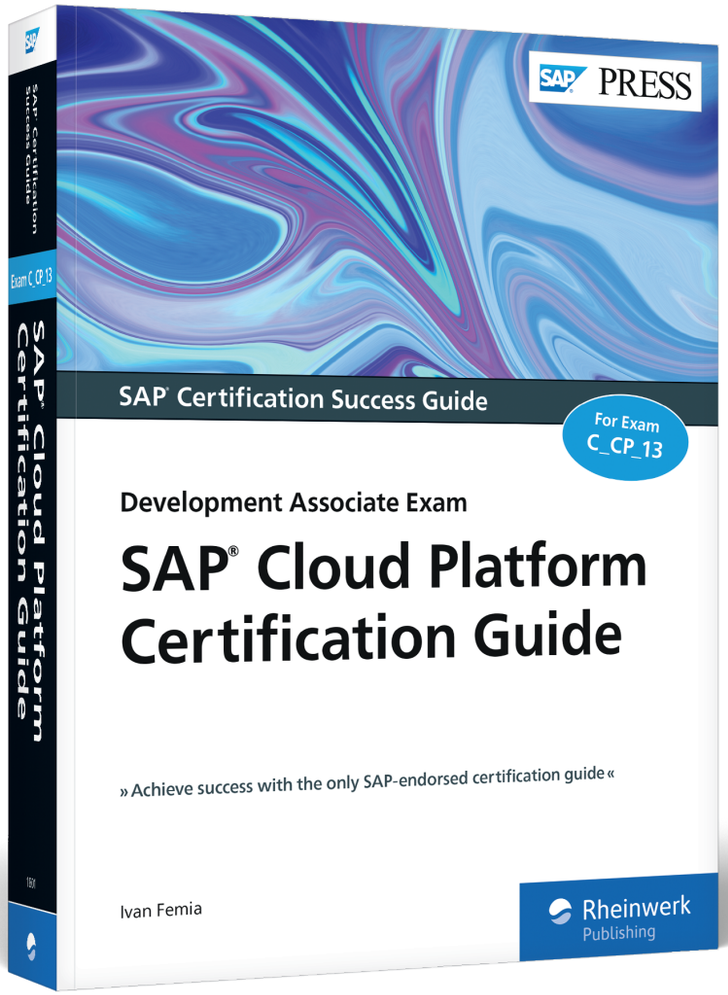
U

V

W

X

Y



Ivan Femia is a senior solution engineer, SAP technology ambassador, and SAP Mentors alum, with more than 20 years of experience in application architecture, design, and delivery management.

Ivan Femia

SAP Cloud Platform Certification Guide: Development Associate Exam

477 Pages, 2019, $79.95
ISBN 978-1-4932-1801-1

www.sap-press.com/4825