

Building SQL-Free Applications with MySQL REST Service

Frédéric Descamps

Community Manager

Oracle MySQL

preFOSDEM MySQL Belgian Days - January 2026





Who am I ?

about.me/lefred

Frédéric Descamps

-  @lefred
-  @lefredbe.bsky.social
-  @lefred@fosstodon.org
-  @lefred14:matrix.org
-  Evangelist
- using  since version 3.20
- *devops believer*
- *living in* 
- <https://lefred.be>



MySQL REST Service

Agenda

Agenda

- *About MySQL REST Service (MRS)*
- *Installation of MySQL Labs with MRS*
- *Installation of MySQL Shell for Visual Studio Code*
- *Loading sample database*
- *Building RESTful APIs with MRS*
 - *with MySQL Shell for VS Code*
 - *with SQL in MySQL Shell*
- *Using the Python SDK*
- *Q&A Session*



MySQL REST Service

About

MySQL REST Service (MRS)

Fast, Secure HTTPS Access for MySQL Data

MRS is built on the concepts and architecture of Oracle REST Data Services (ORDS)

It's available in three different deployment options:

- *as middleware using MySQL Router*
- *as part of MySQL HeatWave*
- *as a component to run in the MySQL Server (Lab release)*

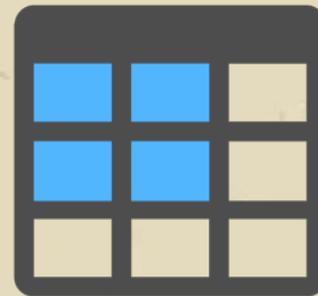
MySQL REST Service (MRS)

Features Overview



RESTful Web Services

- *auto REST for tables, view, procedures and functions*
- *{JSON} responses with paged results*
- *developer support (GUI, CLI, API)*
- *support for popular OAuth2 services*



Powerful REST Data Mapping

- *full SQL support for joins and nested JSON REST endpoints*
- *Visual Data Mapping Editor*
 - *build complex JSON structures with a few clicks*
- *SQL & SDK interface preview*

MySQL REST Service (MRS)

Features Overview (2)



Full SQL Support & SDK API

- *fully manageable through SQL*
- *tailored SDK for all RESTful Endpoints*
- *popular, Prisma-like API, like prototyping*

```
ts> myService.sakil
{
  "city": "A
  "cityId": 1
  "country":
    "country"
```

```
sql> CONFIGURE REST METADATA;
sql> CREATE REST SERVICE /myService;
sql> CREATE REST SCHEMA /sakila FROM `sakila`;
```

TypeScript SDK API with live prototyping of REST queries

Fully manageable through SQL

MySQL REST Service (MRS)

Visual REST Mapping View Editor

Intuitive WYSIWYG Editor to design REST Mappings

- creation of complex mappings with a few clicks
- automatic database schema analysis
- SQL Preview



① Add single relational table as REST object

The screenshot shows the 'MySQL REST Object' configuration window. It includes fields for REST Service Path, REST Schema Path, and REST Object Path. The 'MRS Object Flags' section has 'Enabled' checked and 'Requires Auth' selected. The 'JSON/Relational Duality' tab is active, showing a mapping between a database object 'city' and an MRS object 'MyServiceSakilaCity'. The mapping shows columns like 'cityId', 'countryId', and 'lastUpdate' being mapped to JSON fields. A 'Referenced Tables' section shows 'country' and 'address' tables being mapped to 'sakila.country' and 'sakila.address' respectively. A 'DDL Preview' button is also visible.

② Click referenced table to add nested JSON documents

This screenshot shows a detailed view of the nested JSON mapping configuration. It shows the 'country' table being mapped to a nested JSON structure. The 'country' table's columns are mapped to 'country_id', 'country', and 'last_update' in the nested JSON. The 'address' table is also mapped to 'sakila.address'.

③ Store REST Mapping

MySQL REST Service is part of the Server

The world's most popular open source database

Contact MySQL | Login | Register

MySQL.COM DOWNLOADS DOCUMENTATION **DEVELOPER ZONE**

Forums Bugs Worklog **Labs** Planet MySQL News and Events Community Blog Archive

MySQL Labs :: MySQL 9.4.0 Labs MRS 14

Warning! For testing purposes only!

These binaries were created by MySQL testing servers.
They are **NOT FIT FOR PRODUCTION**.
They are provided solely for testing purposes, to try the latest bug fixes and generally to keep up with the development.

- Please, DO NOT USE THESE BINARIES IN PRODUCTION.
- Instead, install them on a spare server.
- If you are looking for production ready binaries, please visit [MySQL Downloads](#).
- MySQL Software is provided under the [GPL License](#)

MySQL 9.4.0 Labs MRS 14

MySQL 9.4.0 Labs release with MySQL REST Service (MRS) 14

14

- smarter default HTTP port
- quieter error log
- handles server restarts gracefully
- safer install when account exists
- race condition fixed
- build system refactor
- other improvements, refactors and fixes

MySQL REST Service is part of the Server

MySQL Labs :: MySQL 9.5.0 Labs MRS 15

 **Warning! For testing purposes only!**

These binaries were created by MySQL testing servers.

They are **NOT FIT FOR PRODUCTION**.

They are provided solely for testing purposes, to try the latest bug fixes and generally to keep up with the development.

- Please, DO NOT USE THESE BINARIES IN PRODUCTION.
- Instead, install them on a spare server.
- If you are looking for production ready binaries, please visit [MySQL Downloads](#).
- MySQL Software is provided under the [GPL License](#)

MySQL 9.5.0 Labs MRS 15 ▾

Filename	Version
mysql-community-9.5.0-labs-mrs-docker.tar.gz	9.5.0



15

- based on MySQL 9.5.0
- full support of MySQL Database Architectures (InnoDB Cluster, ReplicaSet, ClusterSet)
- new system variable
- new status variable
- various improvements, refactors and fixes

<https://blogs.oracle.com/mysql/post/uplevel-the-mysql-rest-service>

MySQL REST Service is part of the Server

MySQL Labs :: MySQL 9.6.0 Labs MRS 16

 **Warning! For testing purposes only!**

These binaries were created by MySQL testing servers.

They are **NOT FIT FOR PRODUCTION**.

They are provided solely for testing purposes, to try the latest bug fixes and generally to keep up with the development.

- Please, DO NOT USE THESE BINARIES IN PRODUCTION.
- Instead, install them on a spare server.
- If you are looking for production ready binaries, please visit [MySQL Downloads](#).
- MySQL Software is provided under the [GPL License](#)

MySQL 9.6.0 Labs MRS 16 ▾

Filename	Version	Date	Platform	Size	MD5sum
mysql-community-9.6.0-labs-mrs-docker.tar.gz	9.6.0	2026-01-27	Docker tar.gz	307.2M	ac55d4eb4bf650d8a4db40dcaa44b53f



16

- based on MySQL 9.6
- integrated with MySQL
server Keyring

MySQL Server & MySQL Shell

Installation

Installation: MySQL Server

Demo using MySQL Labs 9.5.0 with MRS included that you can download from:

<https://labs.mysql.com/>

And we will use docker in the examples but you can install it natively on your OS as well.



docker

Installation: MySQL Server (using docker)



```
$ docker load --input mysql-community-9.6.0-labs-mrs-docker.tar.gz
```

```
$ docker image list
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
cont[...]/community-server	9.6-amd64	3cbcdeb1a24d	5 hours ago	1.1 GB

MySQL Server Keyring - **NEW**

The MRS component is now (since 9.6.0) using the [MySQL Server Keyring](#) to store passwords of the internal accounts used by MRS.

We need then to configure the keyring manifest when starting the container.

We use `keyring-file`.

mysqld.my

```
{  
  "components": "file://component_keyring_file"  
}
```

MySQL Server Keyring (2) - **NEW**

component_keyring_file.cnf

```
{  
  "path": "/var/lib/mysql-keyring/component_keyring_file",  
  "read_only": false  
}
```

And we create the directory on the host machine:

```
$ mkdir -p ./mysql-keyring  
$ chmod 777 ./mysql-keyring
```

Launching the MySQL Server in Docker

```
$ docker run -d --name mrs -e MYSQL_ROOT_PASSWORD=F0sd3M \  
-p 3306:3306 -p 33060:33060 -p 33061:33061 \  
--mount type=bind,src="mysqld.my",dst=/usr/sbin/mysqld.my,ro \  
--mount type=bind,src="component_keyring_file.cnf",dst=/usr/lib64/mysql/plugin/component_keyring_file.cnf,ro \  
--mount type=bind,src="mysql-keyring",dst=/var/lib/mysql-keyring \  
container-registry.oracle.com/mysql/community-server:9.6-amd64
```

Launching the MySQL Server in Docker

```
$ docker run -d --name mrs -e MYSQL_ROOT_PASSWORD=F0sd3M \  
-p 3306:3306 -p 33060:33060 -p 33061:33061 \  
--mount type=bind,src="mysqld.my",dst=/usr/sbin/mysqld.my,ro \  
--mount type=bind,src="component_keyring_file.cnf",dst=/usr/lib64/mysql/plugin/component_keyring_file.cnf,ro \  
--mount type=bind,src="mysql-keyring",dst=/var/lib/mysql-keyring \  
container-registry.oracle.com/mysql/community-server:9.6-amd64
```

```
$ printf "%-12s %-10s %-10s %-20s %-12s\n" ID NAME COMMAND CREATED SIZE  
docker ps -sa --format \  
"{{printf \ "%-12s %-10s %-10s %-20s %-12s\ " .ID .Names .Command .RunningFor .Size}}"  
ID NAME COMMAND CREATED SIZE  
68d33d0dc412 mrs mysqld 2 minutes ago 215MB (virtual 1.31GB)
```

Installation: MySQL Server (using docker) (2)



Now we need to create a user that will be able to connect from your machine to to database server in the container:

```
$ docker exec -it mrs mysql -uroot -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.6.0-labs-mrs-16 MySQL Community Server - GPL

mysql> create user fosdem identified by 'F0sd3M';
Query OK, 0 rows affected (0.018 sec)

mysql> grant all privileges on *.* to fosdem with grant option;
Query OK, 0 rows affected (0.014 sec)
```

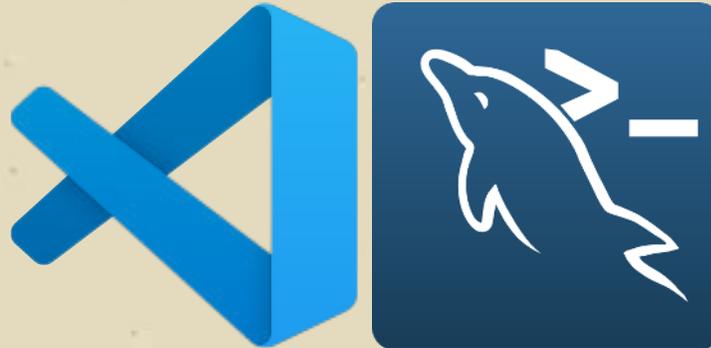
Installation: MySQL Shell for Visual Studio Code



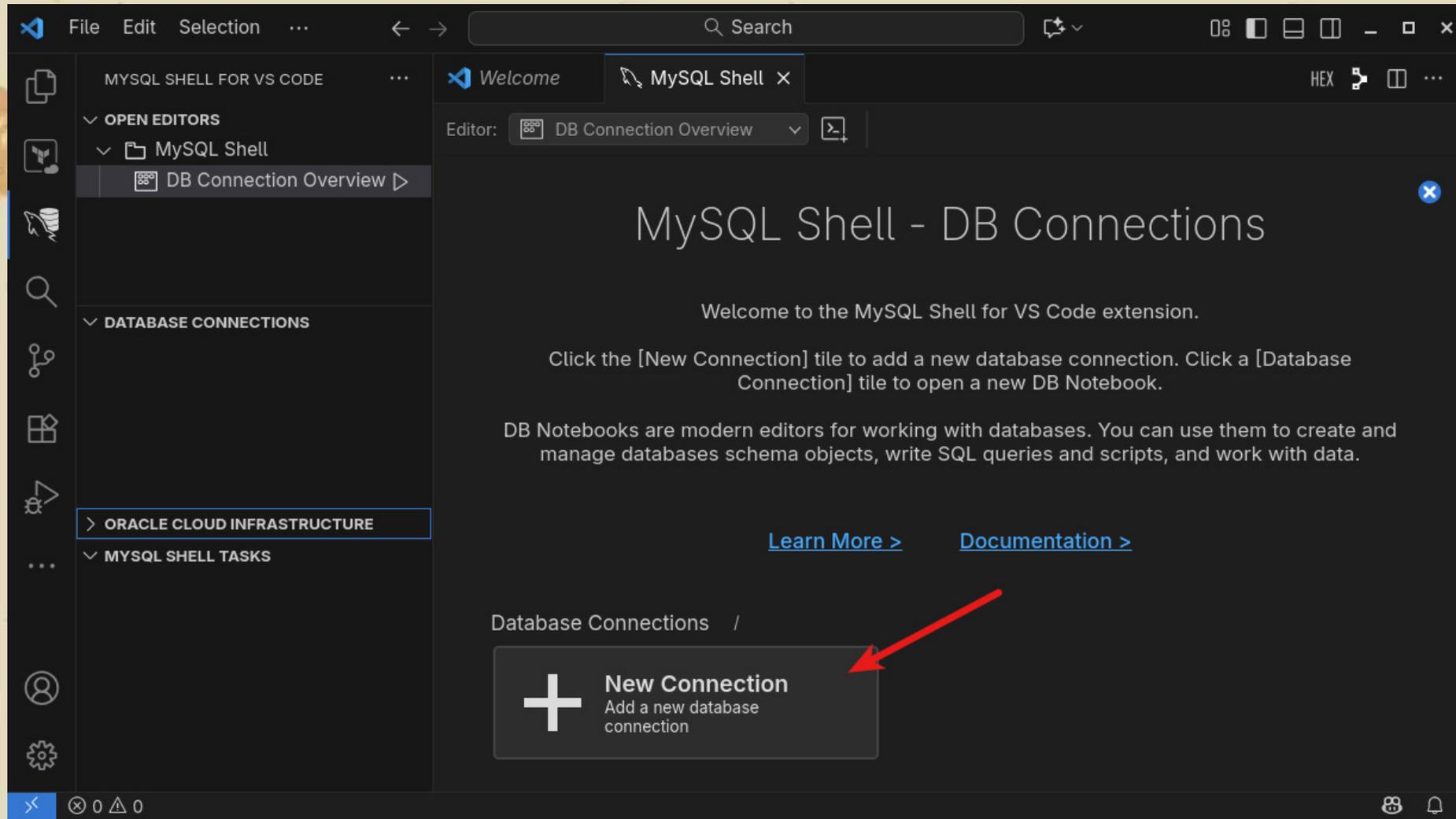
You need to install Visual Studio Code and then the [MySQL Shell](#) extension from the marketplace.

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter:

```
ext install Oracle.mysql-shell-for-vs-code
```



Connecting to MySQL Server



MySQL Shell - DB Connections

Welcome to the MySQL Shell for VS Code extension.

Click the [New Connection] tile to add a new database connection. Click a [Database Connection] tile to open a new DB Notebook.

DB Notebooks are modern editors for working with databases. You can use them to create and manage databases schema objects, write SQL queries and scripts, and work with data.

[Learn More >](#) [Documentation >](#)

Database Connections /

+ New Connection
Add a new database connection

Connecting to MySQL Server



The screenshot shows the MySQL Shell configuration dialog in VS Code. The dialog is titled "Database Connection Configuration" and has several sections:

- Caption:** mrs - fosdem
- Description:** My MRS Server for preFOSEDEM MySQL Belgian Days 2026
- Default Editor:** DB Notebook
- Folder Path:** (empty)
- Database Type:** MySQL

The "Basic" tab is selected, showing the following fields:

- Host Name or IP Address:** 127.0.0.1
- Protocol:** mysql
- Port:** 3306
- User Name:** fosdem
- Default Schema:** (empty)

There are "Store Password" and "Clear Password" buttons next to the User Name field. At the bottom right, there are "Cancel" and "OK" buttons, with a red arrow pointing to the "OK" button.

Connecting to MySQL Server



MySQL Shell - DB Connections

Welcome to the MySQL Shell for VS Code extension.

Click the [New Connection] tile to add a new database connection. Click a [Database Connection] tile to open a new DB Notebook.

DB Notebooks are modern editors for working with databases. You can use them to create and manage databases schema objects, write SQL queries and scripts, and work with data.

[Learn More >](#) [Documentation >](#)

Database Connections /

-  **mrs - fosdem**
My MRS Server for preFOSEM MySQL Belgian Days 2026
-  **New Connection**
Add a new database connection

File Edit Selection ... Search

MySQL Shell x

Editor: DB Connection Overview

File Edit Selection ...

OPEN EDITORS

- MySQL Shell
- DB Connection Over...

DATABASE CONNECTIONS

- mrs - fosdem

ORACLE CLOUD INFRASTRUCTURE

MYSQL SHELL TASKS

0 0 "Building SQL-Free Applications with MySQL REST Service.html" 1187L

Connecting to MySQL Server



The screenshot shows the Visual Studio Code interface with the MySQL Shell extension. The left sidebar is open to the 'MySQL Administration' section, which includes options for 'Server Status', 'Client Connections', 'Performance Dashboard', and 'Lakehouse Navigator'. The main editor area displays a 'DB Notebook' with the following text:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>
```

The status bar at the bottom indicates 'Ln 1, Col 1 Spaces: 4 LF mixed/mysql'.

Connecting to MySQL Server (2)

If we prefer to use *MySQL Shell* from command line, we can do it like this if we have it installed (from <https://dev.mysql.com/downloads/shell/>):

```
$ mysqlsh mysql://fosdem@127.0.0.1:3306
Please provide the password for 'fosdem@127.0.0.1:3306': *****
Save password for 'fosdem@127.0.0.1:3306'? [Y]es/[N]o/[e]ver (default No): yes
MySQL Shell 9.6.0

Type '\help' or '\?' for help; '\quit' to exit.
Creating a Classic session to 'fosdem@127.0.0.1:3306'
Fetching global names for auto-completion... Press ^C to stop.
Your MySQL connection id is 12
Server version: 9.6.0-labs-mrs-16 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL>
```



Sample Database

Sakila

Loading Sample Database: Sakila

We will use the Sakila sample database that you can download from:

<https://downloads.mysql.com/docs/sakila-db.zip>

The Sakila sample database was initially developed by Mike Hillyer, a former member of the MySQL AB documentation team. It is intended to provide a standard schema that can be used for examples in books, tutorials, articles, samples, and so forth. The Sakila sample database also serves to highlight features of MySQL such as Views, Stored Procedures, and Triggers.

Loading Sample Database: Sakila



We will use the Sakila sample database that you can download from:

<https://downloads.mysql.com/docs/sakila-db.zip>

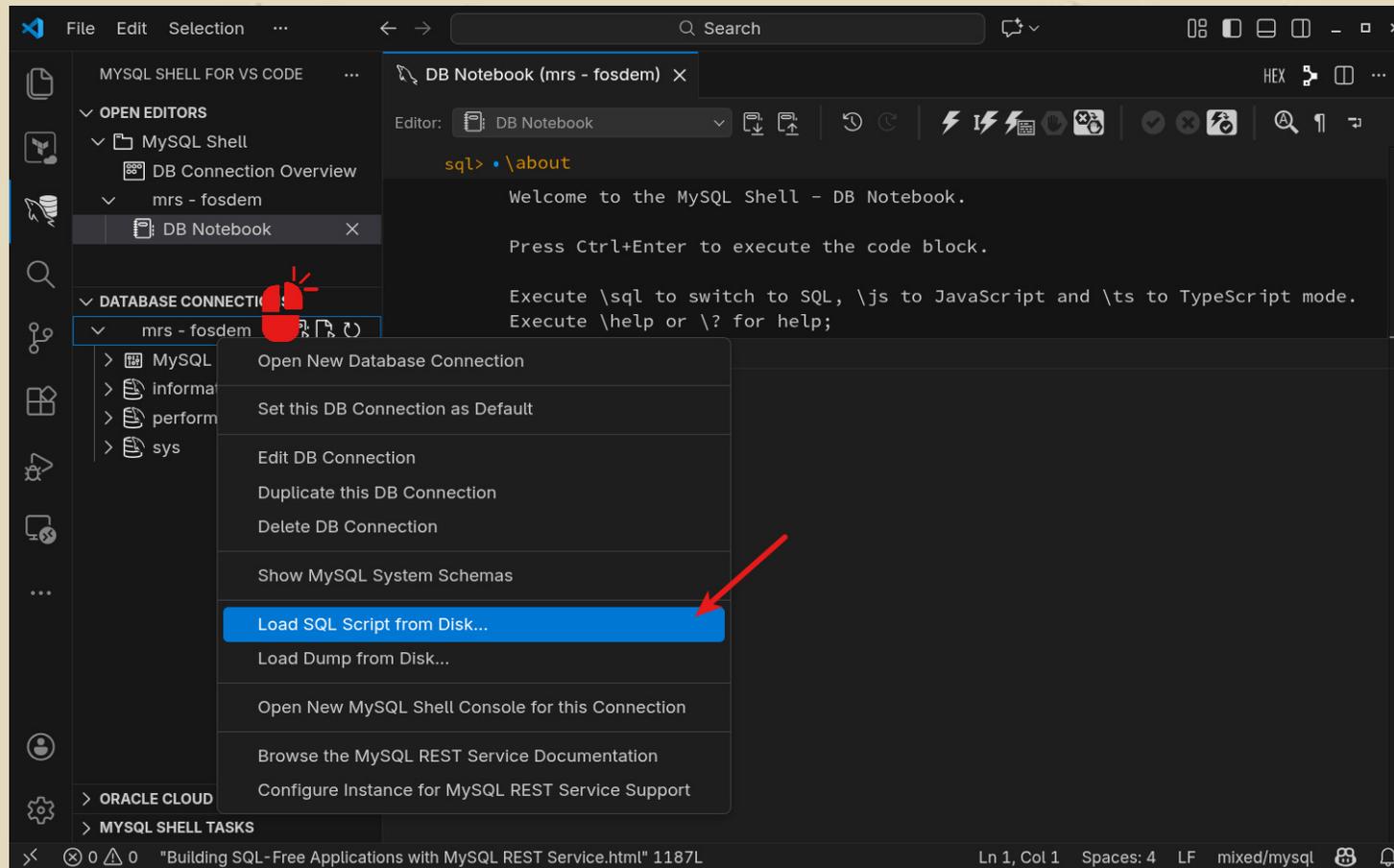
The Sakila sample database was initially developed by Mike Hillyer, a former member of the MySQL AB documentation team. It is intended to provide a standard schema that can be used for examples in books, tutorials, articles, samples, and so forth. The Sakila sample database also serves to highlight features of MySQL such as Views, Stored Procedures, and Triggers.

Download it and unzip it!

Loading Sample Database: Sakila (2)



In *MySQL Shell* for VS Code, open the `sakila-schema.sql` file:



Loading Sample Database: Sakila (2)



```
679 • DELIMITER ;
680
681 • SET SQL_MODE=@OLD_SQL_MODE;
682 • SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
683 • SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
684
685
686
687
```

press CTRL+ENTER

Output

```
#34: OK, 0 records retrieved in 21.694ms
#35: OK, 0 records retrieved in 17.625ms
#36: OK, 0 records retrieved in 21.621ms
#37: OK, 0 records retrieved in 18.909ms
#38: OK, 0 records retrieved in 14.797ms
#39: OK, 0 records retrieved in 19.945ms
#41: OK, 0 records retrieved in 14.964ms
#44: OK, 0 records retrieved in 23.791ms
#47: OK, 0 records retrieved in 14.934ms
#50: OK, 0 records retrieved in 14.776ms
#53: OK, 0 records retrieved in 21.771ms
#56: OK, 0 records retrieved in 23.02ms
#58: OK, 0 records retrieved in 0.347ms
#59: OK, 0 records retrieved in 0.386ms
#60: OK, 0 records retrieved in 0.365ms
```

✓ SQL Script execution completed in 5.083s. 46 statements executed successfully.

Loading Sample Database: Sakila (2)



Then load the `sakila-data.sql` file:

The screenshot shows the MySQL Shell for VS Code interface. The editor displays the `sakila-data.sql` file with SQL data. Red circles and arrows highlight the file name in the editor (2) and the Run button in the toolbar (1). A red circle highlights the text "press CTRL+ENTER" (3) in the output window.

```
688 (438,'596 Huixquilucan Place','','Nampula',351,'65892','342709348083',/*!50705 0x0000
689 (439,'1351 Aparecida de Goinia Parkway','','Northern Mindanao',391,'41775','9598
690 (440,'722 Bradford Lane','','Shandong',249,'90920','746251338300',/*!50705 0x0000
691 (441,'983 Santa F Way','','British Columbia',565,'47472','145720452260',/*!50705
692 (442,'1245 Ibrit Way','','La Romana',290,'40926','331888642162',/*!50705 0x0000
693 (443,'1836 Korla Parkway','','Copperbelt',272,'55405','689681677428',/*!50705 0x
694 (444,'231 Kaliningrad Place','','Lombardia',70,'57833','575081026569',/*!50705 0
695 (445,'495 Bhimavaram Lane','','Maharashtra',144,'3','82088937724',/*!50705 0x0000
696 (446,'1924 Shimonoseki Drive','','Batna',59,'52625','406785440',/*!50705 0x0000
---
```

Output

```
#1: OK, 0 records retrieved in 0.402ms
#2: OK, 0 records retrieved in 0.426ms
#3: OK, 0 records retrieved in 0.524ms
#4: OK, 0 records retrieved in 0.632ms
#5: OK, 0 records retrieved in 0.416ms
#6: OK, 0 records retrieved in 0.451ms
#7: OK, 0 records retrieved in 0.496ms
```

Loading Sample Database: Sakila (2)



```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql> use sakila
OK, 0 records retrieved in 0.509ms

sql> select count(*) from actor
count(*)
200

OK, 1 record retriev... View: [table icon] Pages: [left] [right] Edit: [check] [eye] [x] [refresh] [copy] [menu]
```

Building RESTful APIs with MRS

Preparing the Server

Creating the REST metadata

The very first step for the [MySQL REST Service](#) component to work is to create the metadata schema.

We perform this operation even before loading the component.

Creating the REST metadata

The very first step for the [MySQL REST Service](#) component to work is to create the metadata schema.

We perform this operation even before loading the component.

This operation can be done in [SQL](#) like this but only using [MySQL Shell](#):

```
SQL> CONFIGURE REST METADATA;
```

But we will do it using [MySQL Shell](#) for VS Code as it automates more things for us.

Creating the REST metadata - VS Code



The screenshot shows the VS Code interface with the MySQL Shell extension. The 'DB Notebook (mrs - fosdem)' editor is open, displaying a welcome message and instructions. The 'DATABASE CONNECTIONS' sidebar is expanded, showing a tree view of the 'mrs - fosdem' connection. A context menu is open over the 'mrs - fosdem' connection, listing various actions. The 'Configure Instance for MySQL REST Service Support' option is highlighted in blue, and a red arrow points to it.

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.

help;
```

- Open New Database Connection
- Set this DB Connection as Default
- Edit DB Connection
- Duplicate this DB Connection
- Delete DB Connection
- Show MySQL System Schemas
- Load SQL Script from Disk...
- Load Dump from Disk...
- Open New MySQL Shell Console for this Connection
- Browse the MySQL REST Service Documentation
- Configure Instance for MySQL REST Service Support**

Creating the REST metadata - VS Code



The screenshot shows the VS Code interface with a 'MySQL REST Service' configuration dialog open. The dialog is titled 'Configure Instance for MySQL REST Service Support'. It features several configuration options:

- MySQL REST Service Status:** Set to 'Enabled'.
- Version:** Set to '4.1.5'.
- Create default REST authentication app:** This checkbox is checked and highlighted with a red arrow. Below it, a note states: 'Select this option to create a REST authentication app using the built-in MRS vendor. REST authentication apps are required to allow end users to login into the MySQL REST Service and access REST endpoints that require authentication.'
- REST User Name:** The text 'fosdem_rest_user' is entered in the field and highlighted with a red box. Below it, a note states: 'The user name of the REST user account. This account will be enabled to login into the MySQL REST Service and access all REST endpoints of the linked REST services.'
- REST User Password:** The password field is masked with dots. Below it, a note states: 'The password of the REST user account.'

At the bottom right of the dialog, there are 'Cancel' and 'OK' buttons. The 'OK' button is highlighted with a red arrow. The background shows the VS Code editor with a 'DB Notebook' open, displaying a tree view of database connections and tables.

Creating the REST metadata - VS Code



The screenshot shows the Visual Studio Code interface with a MySQL Shell notebook open. The notebook content includes a welcome message and instructions for using the shell. A red box highlights a status message at the bottom of the notebook: "MySQL REST Service configured successfully." The left sidebar shows the "DATABASE CONNECTIONS" tree with the "MySQL REST Service" connection selected. The status bar at the bottom indicates the current file is "Building SQL-Free Applications with MySQL REST Service.html" and shows the current cursor position as "Ln 1, Col 1".

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>
```

MySQL REST Service configured successfully.

Creating the REST metadata - VS Code



The screenshot shows the Visual Studio Code interface. On the left, the Database Explorer is open, displaying a tree view of database connections. The 'MySQL REST Service' folder is expanded, and the 'fosdem_rest_user' connection is selected. The main editor area shows a 'DB Notebook (mrs - fosdem)' window. The notebook content includes a SQL prompt 'sql>' followed by a welcome message: 'Welcome to the MySQL Shell - DB Notebook. Press Ctrl+Enter to execute the code block. Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode. Execute \help or \? for help;'. The status bar at the bottom indicates the current file is 'Building SQL-Free Applications with MySQL REST Service.html' with 1187L lines.

Loading the MRS Component

This lab version provides a MRS component pre-installed but not yet loaded. This is a nice enhancement as you don't need to use [MySQL Router](#) as middleware between your application and the database server to provide RESTful services.

Loading the MRS Component



This lab version provides a MRS component pre-installed but not yet loaded. This is a nice enhancement as you don't need to use [MySQL Router](#) as middleware between your application and the database server to provide RESTful services.

```
MySQL > install component "file://component_mysql_rest_service";
```

```
MySQL > select * from mysql.component;
```

```
+-----+-----+-----+
| component_id | component_group_id | component_urn |
+-----+-----+-----+
|          1 |          1 | file://component_mysql_rest_service |
+-----+-----+-----+
1 row in set (0.0014 sec)
```

MRS Component and Keyring - **required**

If the keyring service is not properly configured, you will get an error like this when loading the component:

```
MySQL > install component "file://component_mysql_rest_service";  
ERROR: 6557 (HY000): Failed to initialize Keyring.
```

And in the error log:

```
MySQL Rest Service Component: Unable to initialize keyring:  
MySQL Keyring is required to run MRS.  
Install and initialize a supported MySQL keyring backend beforehand.  
See https://dev.mysql.com/doc/refman/en/keyring.html
```

MRS Component and Keyring

Now the user for the REST Service internal operations is automatically created and its password is stored in the Keyring:

```
| System | MY-015163 | Server |  
MySQL Rest Service Component: Creating MRS account: mysql_mrs_1
```

```
SQL > SELECT to_user Users, GROUP_CONCAT(from_user SEPARATOR ', ') Roles  
FROM mysql.role_edges where to_user = 'mysql_mrs_1' GROUP BY to_user;
```

```
+-----+-----+  
| Users      | Roles                                     |  
+-----+-----+  
| mysql_mrs_1 | mysql_rest_service_data_provider, mysql_rest_service_meta_provider |  
+-----+-----+  
1 row in set (0.0005 sec)
```

MRS Component and Keyring (2)

```
$ cat mysql-keyring/component_keyring_file | jq
{
  "version": "1.0",
  "elements": [
    {
      "user": "",
      "data_id": "mrs/rest-user/jwt_secret",
      "data_type": "SECRET",
      "data": "736563726574",
      "extension": []
    },
    {
      "user": "",
      "data_id": "mrs/mysql_mrs_1/password",
      "data_type": "SECRET",
      "data": "6D5F353355452C2C36404C416C716B534661744B",
      "extension": []
    }
  ]
}
```

Building RESTful APIs with MRS

MySQL without SQL

Our first RESTful Endpoint

*We will create a first simple RESTful Endpoint that will expose the **actor** table from the **sakila** database.*

We won't use any authentication for this first example to keep it simple.

Our first RESTful Endpoint

*We will create a first simple RESTful Endpoint that will expose the **actor** table from the **sakila** database.*

We won't use any authentication for this first example to keep it simple.

We need to create a REST Service first.



Our first REST Service



The screenshot shows the VS Code interface with the MySQL REST Service configuration menu open. The menu is located in the 'DATABASE CONNECTIONS' sidebar and is expanded to show the following options:

- Configure MySQL REST Service
- Add REST Service...** (highlighted with a red arrow)
- Load from Disk
- Bootstrap Local MySQL Router Instance
- Start Local MySQL Router Instance
- Stop Local MySQL Router Instance
- Kill Local MySQL Router Instances
- Show Private Items
- Browse the MySQL REST Service Documentation

The main editor area shows the MySQL Shell prompt with the following text:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;
```

Our first REST Service



The screenshot shows the 'MySQL REST Service' configuration dialog box. The 'REST Service Path' is set to `/fosdem`. The 'REST Service Name' is 'MyService for FOSDEM'. The 'REST Service Flags' section has 'Enabled', 'Default', and 'Published' checked. The 'Linked REST Authentication Apps' section has 'MRS' and 'MySQL' checked. The 'OK' button is highlighted with a red arrow.

Enter Configuration Values for the New REST Service

REST Service Path: `/fosdem`
The URL context root of this service, has to start with / and needs to be unique.

REST Service Name: MyService for FOSDEM
The descriptive name of the REST service.

REST Service Flags:
 Enabled
 Default
 Published

Settings Options Authentication Details Advanced

Linked REST Authentication Apps:
 MRS
 MySQL
Select one or more REST authentication app. This allows REST users of those applications to authenticate.

Comments:
Comments to describe this REST Service.

Cancel OK

Our first REST Service



The screenshot displays the MySQL Shell interface within VS Code. The left sidebar shows the 'DATABASE CONNECTIONS' tree with the following structure:

- mysql - fosdem
 - MySQL REST Service
 - /fosdem Published
 - MySQL Routers
 - REST Authentication A...
 - MRS MRS
 - fosdem_rest_user
 - MySQL MySQL Internal
 - MySQL Administration
 - information_schema
 - performance_schema
 - sakila
 - Tables
- ORACLE CLOUD INFRASTRUCTURE
- MYSQL SHELL TASKS

The main editor shows the MySQL Shell prompt with the command `\about` entered. The output is:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;
```

The status bar at the bottom right shows a message: `The MRS service has been created.`

Our first RESTful Endpoint



The screenshot displays the Visual Studio Code interface with the MySQL Shell extension. The 'DATABASE CONNECTIONS' sidebar on the left shows a tree view of database objects. A context menu is open over the 'actor' table, with the option 'Add Database Object to REST Service' highlighted in blue and pointed to by a red arrow. The main editor shows the MySQL Shell prompt and a welcome message.

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
or \? for help;
```

Our first RESTful Endpoint



The screenshot shows the MySQL Shell for VS Code interface. The main editor displays the MySQL Shell help text:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;
```

A notification dialog is open in the bottom right corner, asking if the user wants to add the sakila schema to the REST Service. The dialog text is:

The database sakila has not been added to the REST Service. Do you want to add the schema now?

Source: MySQL Shell for VS Code

The **Yes** button is highlighted with a red arrow.

Our first RESTful Endpoint



MySQL REST Object

REST Service Path: `/fosdem`
REST Schema Path: `/sakila`
REST Object Path: `/actor`
Access Control: Access ENABLED
 Auth. Required

Data Mapping

DB Object: `actor` SDK Language: `SQL Preview`

```
FosdemSakilaActor {
  actorId
  firstName
  lastName
  lastUpdate
  filmActorActorId
}
```

actorId → actor_id
firstName → first_name
lastName → last_name
lastUpdate → last_update
filmActorActorId → sakila.film_actor

Cancel OK

Our first RESTful Endpoint



The screenshot shows the VS Code DB Notebook interface. The left sidebar displays the 'DATABASE CONNECTIONS' tree, including 'mrs - fosdem', 'MySQL REST Service', and 'REST Authentication A...'. The main editor area is titled 'MySQL REST Object' and shows the following configuration:

- REST Service Path: /fosdem
- REST Schema Path: /sakila
- REST Object Path: /actor
- Access Control: Access ENABLED

Below the configuration, there are tabs for 'Data Mapping', 'Settings', 'Authorization', and 'Options'. The 'Data Mapping' tab is active, showing the 'DB Object: actor' and a preview of the SQL statement:

```
CREATE OR REPLACE REST VIEW /actor
ON SERVICE /fosdem SCHEMA /sakila
AS sakila.actor CLASS FosdemSakilaActor {
  `actorId`: `actor_id` @SORTABLE @KEY,
  `firstName`: `first_name`,
  `lastName`: `last_name`,
  `lastUpdate`: `last_update`
```

A red arrow points to the 'SQL Preview' button in the top right corner of the SQL editor area. The status bar at the bottom indicates the current file is 'Building SQL-Free Applications with MySQL REST Service.html' at line 1, column 1.

Our first RESTful Endpoint



The screenshot displays the MySQL Shell interface within VS Code. The left sidebar shows the 'DATABASE CONNECTIONS' tree with 'sakila' selected. The main editor shows the MySQL Shell prompt 'sql>' and a message: 'Welcome to the MySQL Shell - DB Notebook. Press Ctrl+Enter to execute the code block. Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode. Execute \help or \? for help;'. A red box highlights a notification at the bottom right: 'The MRS Database Object actor was successfully updated.'

Our first RESTful Endpoint



The screenshot displays the Visual Studio Code interface with the MySQL REST Service extension. The left sidebar shows a tree view of database connections, with the 'MySQL REST Service' section expanded and the '/sakila (sakila)' connection selected. The main editor area shows the 'DB Notebook' interface with a 'sql>' prompt and a welcome message.

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>
```

Our first RESTful Endpoint - testing



We can now try to browse to our first RESTful Endpoint using curl or a browser:

<https://127.0.0.1:33060/fosdem/sakila/actor>

The certificate is self-signed so you need to accept the risk if you use a browser.

```
$ curl -s -k https://127.0.0.1:33060/fosdem/sakila/actor | jq
```

Our first RESTful Endpoint - testing



```
File Edit View History Bookmarks Profiles Tools Help
127.0.0.1:33060/fosdem/ ×
127.0.0.1:33060/fosdem/sakila/a ☆ Search
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
items:
  0:
    links:
      0:
        rel: "self"
        href: "/fosdem/sakila/actor/1"
    actorId: 1
    lastName: "GUINNESS"
    firstName: "PENELOPE"
    lastUpdate: "2006-02-15 04:34:33.000000"
    _metadata:
      etag: "FF81D7836DE842623AE9D92437EE92F85568EC52960349A031FA74A23D604D1C"
  1:
    links:
      0:
        rel: "self"
        href: "/fosdem/sakila/actor/2"
    actorId: 2
    lastName: "WAHLBERG"
    firstName: "NICK"
    lastUpdate: "2006-02-15 04:34:33.000000"
    _metadata:
      etag: "C87047710CF2528D34AFB8FB964397ED7958A6C9891F9EA52D7BA1E11F110961"
  2:
    links:
      0:
        rel: "self"
```

Our first RESTful Endpoint - filtering

We can request an actor by its primary key like this:

```
fred@dell:~  
[fred@dell ~] $ curl -s -k https://127.0.0.1:33060/fosdem/sakila/actor/44 | jq  
{  
  "links": [  
    {  
      "rel": "self",  
      "href": "/fosdem/sakila/actor/44"  
    }  
  ],  
  "actorId": 44,  
  "lastName": "STALLONE",  
  "firstName": "NICK",  
  "lastUpdate": "2006-02-15 04:34:33.000000",  
  "_metadata": {  
    "etag": "C08FCE302FB2C02367272B76ADCD565A94B4555F348BB76293FAFDB13F0592DE"  
  }  
}
```

Our first RESTful Endpoint - filtering (2)

Or even filter out on fields like this:

```
$ curl -s -k --get 'https://127.0.0.1:33060/fosdem/sakila/actor/' \
  --data-urlencode 'q={"firstName":"NICK"}' | \
  jq -r '.items[] | "\(.actorId)\t\(.firstName)\t\(.lastName)">'
2   NICK   WAHLBERG
44  NICK   STALLONE
166 NICK   DEGENERES
```

Our first RESTful Endpoint - filtering (2)

Or even filter out on fields like this:

```
$ curl -s -k --get 'https://127.0.0.1:33060/fosdem/sakila/actor/' \
  --data-urlencode 'q={"firstName":"NICK"}' | \
  jq -r '.items[] | "\(.actorId)\t\(.firstName)\t\(.lastName)">'
2   NICK   WAHLBERG
44  NICK   STALLONE
166 NICK   DEGENERES
```

```
MySQL > select count(*) from actor where first_name like 'nick';
+-----+
| count(*) |
+-----+
|         3 |
+-----+
1 row in set (0.0005 sec)
```

Building RESTful APIs with MRS

MySQL without SQL and without GUI

All in Command Line in SQL

*We can also create our RESTful Endpoint using only **SQL**.*

This is useful if we want to automate the creation of RESTful Endpoints.

*We can get all information about the RESTful Endpoints from **MySQL Shell** in VS Code by dumping your services:*

All in Command Line in SQL (2)

```
CREATE OR REPLACE REST SERVICE /fosdem PUBLISHED
OPTIONS {
  "http": { "allowedOrigin": "auto" },
  "headers": {
    "Access-Control-Allow-Headers": "Content-Type, Authorization, X-Requested-With, Origin, X-Auth-Token",
    "Access-Control-Allow-Methods": "GET, POST, PUT, DELETE, OPTIONS",
    "Access-Control-Allow-Credentials": "true"
  },
  "logging": {
    "request": {
      "body": true,
      "headers": true
    },
    "response": {
      "body": true,
      "headers": true
    },
    "exceptions": true
  },
  "includeLinksInResults": false,
  "returnInternalErrorDetails": true
}
```

All in Command Line in SQL (3)

```
ADD AUTH APP `MRS` IF EXISTS
ADD AUTH APP `MySQL` IF EXISTS;

CREATE OR REPLACE REST SCHEMA /sakila ON SERVICE /fosdem
FROM `sakila`
AUTHENTICATION NOT REQUIRED;

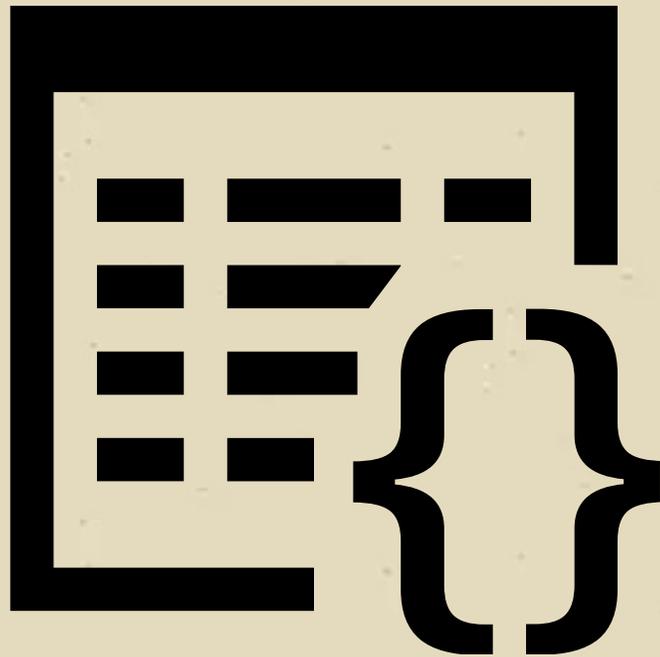
CREATE OR REPLACE REST VIEW /actor
ON SERVICE /fosdem SCHEMA /sakila
AS sakila.actor CLASS FosdemSakilaActor @INSERT @UPDATE {
  firstName: first_name,
  lastName: last_name,
  actorId: actor_id @KEY @SORTABLE,
  lastUpdate: last_update
}
AUTHENTICATION REQUIRED;
```

Using the Python SDK

Modify our RESTful Endpoint

We will now modify our RESTful Endpoint to return a more complex JSON structure and enable authentication.

We will start by editing the REST Object `/actor` we created before.



Modify our RESTful Endpoint



The screenshot displays the MySQL REST Service extension in VS Code. The left sidebar shows a tree view of database connections, with the following structure:

- OPEN EDITORS
 - DB Notebook (mrs - fosdem)
 - DB Connection Overview
 - mrs - fosdem
 - DB Notebook
- DATABASE CONNECTIONS
 - mrs - fosdem
 - MySQL REST Service
 - /fosdem Published
 - /sakila (sakila)
 - /actor actor** (selected)
 - MRS MRS
 - MySQL MySQL Internal
 - MySQL Routers
 - REST Authentication A..
 - MRS MRS
 - fosdem_rest_user
 - MySQL MySQL Internal
 - MySQL Administration
 - ORACLE CLOUD INFRASTRUCTURE
 - MYSQL SHELL TASKS

The main editor shows the MySQL Shell interface with the following text:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>
```

The context menu for the selected '/actor actor' connection includes the following options:

- Edit REST Object...** (highlighted in blue)
- Open REST Object Request Path in Web Browser
- Dump to Disk
- Copy to Clipboard
- Delete REST Object

Modify our RESTful Endpoint



The screenshot shows the Oracle SQL Developer interface with the 'MySQL REST Object' dialog box open. The dialog is configured for a REST endpoint at '/fosdem/sakila/actor'. The 'Authorization' tab is selected, and the 'Auth. Required' checkbox is checked. The 'Data Mapping' tab shows the mapping between the REST object 'ContechSakilaActor' and the database object 'sakila.actor'. The 'UPDATE' button is highlighted with a red arrow.

MySQL REST Object Configuration:

- REST Service Path: /fosdem
- REST Schema Path: /sakila
- REST Object Path: /actor
- Access Control: Access ENABLED
- Auth. Required:

Data Mapping:

REST Object Field	Database Object Field
actorId	actor_id
firstName	first_name
lastName	last_name
lastUpdate	last_update
filmActor	sakila.film_actor

Modify our RESTful Endpoint



MySQL SHELL FOR VS CODE

DB Notebook (mrs - fosdem) X

Editor: DB Notebook

```
sql> \about
```

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>

The MRS Database Object actor was successfully u...

0 Loading MRS auth apps ... "Building SQL-Free Applications with MySQL REST Service.html" 1187L 2332f Ln 1, Col 1 Spaces: 4 LF mixed/mysql

Get the SDK



The screenshot shows the Visual Studio Code interface with the MySQL Shell extension. The 'DB Notebook' editor is active, displaying the following text:

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;
```

The 'MySQL REST Service' connection is selected in the 'DATABASE CONNECTIONS' sidebar. A context menu is open, showing the following options:

- Edit REST Service...
- Set as Current REST Service
- Load from Disk >
- Dump to Disk >**
- Copy to Clipboard >
- Add and Link REST Authentication App...
- Link REST Authentication App...
- Add New REST Content Set
- Deploy OpenAPI Web UI
- Delete REST Service...
- MRS Service Documentation

The 'Dump to Disk' sub-menu is open, showing the following options:

- Dump REST Client SDK Files ...**
- Dump REST SERVICE SQL Script...
- Dump REST Service as REST Project ...

Get the SDK



Visual Studio Code interface showing the 'Export MRS SDK for /fosdem' dialog box. The dialog is open over a 'DB Notebook' editor. The 'REST Service URL' field is highlighted with a red box and contains 'https://127.0.0.1:33060/fosdem'. The 'SDK Client API Language' dropdown is set to 'Python'. The 'Directory' field contains '/home/fred/fosdem.mrs.sdk'. The 'SDK File Header' field contains '/* Copyright (c) 2026, Oracle and/or its affiliates. */'. A red arrow points to the 'OK' button at the bottom right of the dialog.

Get the SDK



The screenshot displays the Visual Studio Code environment with the MySQL Shell for VS Code extension. The left sidebar shows the 'DATABASE CONNECTIONS' tree, with the 'actor' connection selected. The main editor shows the 'DB Notebook' with a code block containing the command `\about`. The output of the command is displayed below the code block, showing a welcome message and instructions for using the shell. A red box highlights a status bar message at the bottom right: "MRS SDK Files exported successfully."

```
sql> \about

Welcome to the MySQL Shell - DB Notebook.

Press Ctrl+Enter to execute the code block.

Execute \sql to switch to SQL, \js to JavaScript and \ts to TypeScript mode.
Execute \help or \? for help;

sql>
```

MRS SDK Files exported successfully.

Get the SDK

We can see that the SDK has been saved in our file system:

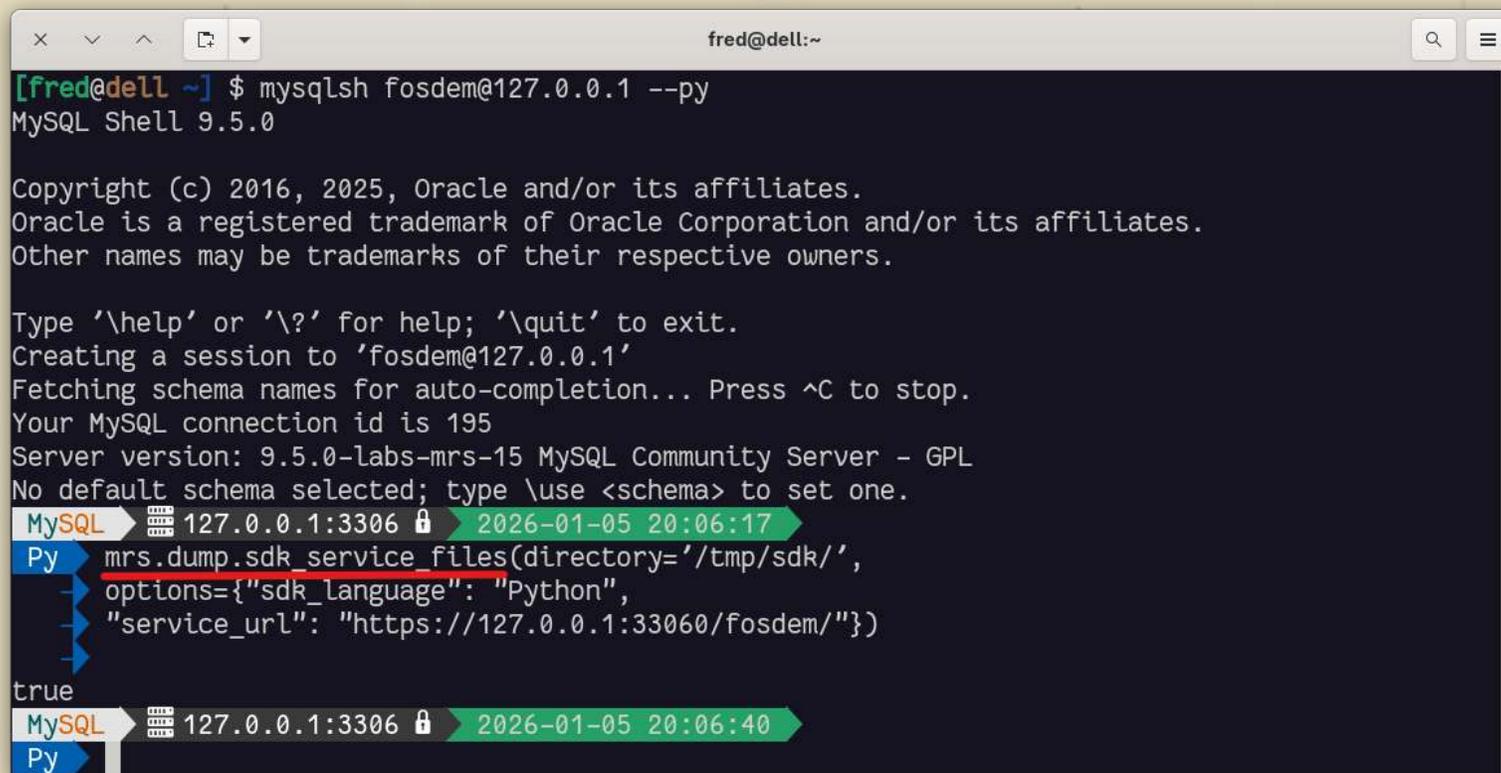
```
$ du -sh fosdem.mrs.sdk
128K   fosdem.mrs.sdk

$ tree fosdem.mrs.sdk/
fosdem.mrs.sdk/
├── fosdem.py
├── __init__.py
├── mrs_base_classes.py
└── mrs.config.json

1 directory, 4 files
```

Get the SDK - cmd line

It's also possible to export the SDK using *MySQL Shell* command Line without using *MySQL Shell* for VS Code like this;



```
fred@dell:~  
[fred@dell ~] $ mysqlsh fosdem@127.0.0.1 --py  
MySQL Shell 9.5.0  
  
Copyright (c) 2016, 2025, Oracle and/or its affiliates.  
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.  
Other names may be trademarks of their respective owners.  
  
Type '\help' or '\?' for help; '\quit' to exit.  
Creating a session to 'fosdem@127.0.0.1'  
Fetching schema names for auto-completion... Press ^C to stop.  
Your MySQL connection id is 195  
Server version: 9.5.0-labs-mrs-15 MySQL Community Server - GPL  
No default schema selected; type \use <schema> to set one.  
MySQL 127.0.0.1:3306 2026-01-05 20:06:17  
Py mrs.dump.sdk_service_files(directory='/tmp/sdk/',  
  options={"sdk_language": "Python",  
  "service_url": "https://127.0.0.1:33060/fosdem/"})  
true  
MySQL 127.0.0.1:3306 2026-01-05 20:06:40  
Py
```

Using the SDK



We create a directory for our Python code and copy the SDK there:

```
[~] $ mkdir fosdem
[~] $ mv fosdem.mrs.sdk fosdem/sdk
[~] $ cd fosdem/
[~/fosdem] $ touch fosdem.py
```

We can edit our code in the terminal (using Vi) or using VS Code.

Using the SDK



```
from sdk.fosdem import *

my_service = Fosdem(verify_tls_cert=False)
async def main():
    await my_service.authenticate(
        username = "fosdem_rest_user",
        password = "Fosd3m2026!",
        app = "MRS",
    )
    records = await my_service.sakila.actor.find(take=10)
    for record in records:
        print(record.first_name + " " + record.last_name)

asyncio.run(main())
```

Using the SDK - running it

```
$ python3.13 fosdem.py  
PENELOPE GUINNESS  
NICK WAHLBERG  
ED CHASE  
JENNIFER DAVIS  
JOHNNY LOLLOBRIGIDA  
BETTE NICHOLSON  
GRACE MOSTEL  
MATTHEW JOHANSSON  
JOE SWANK  
CHRISTIAN GABLE
```

Using the SDK - updates

We can also try to update a record like this:

`fosdem_update.py`

```
from sdk.fosdem import *

my_service = Fosdem(verify_tls_cert=False)

async def main():
    await my_service.authenticate(
        username = "fosdem_rest_user", password = "Fosd3m2026!", app = "MRS",
    )
    records = await my_service.sakila.actor.find(where={"actor_id": {"$eq": 4}})
    for record in records:
        print(record.first_name + " " + record.last_name)
        record.first_name = "ANDRA"
        await record.update();

asyncio.run(main())
```

Using the SDK - updates (2)

```
$ python3.13 fosdem_update.py  
JENNIFER DAVIS
```

Using the SDK - updates (2)

```
$ python3.13 fosdem_update.py  
JENNIFER DAVIS
```

```
$ python3.13 fosdem.py  
PENELOPE GUINNESS  
NICK WAHLBERG  
ED CHASE  
ANDRA DAVIS  
JOHNNY LOLLOBRIGIDA  
BETTE NICHOLSON  
GRACE MOSTEL  
MATTHEW JOHANSSON  
JOE SWANK  
CHRISTIAN GABLE
```

Playing with other database objects

We can also use the SDK to play with other database objects like Views, Stored Procedures and Functions.

The full documentation is available at:

<https://dev.mysql.com/doc/dev/mysql-rest-service/latest/sdk.html>

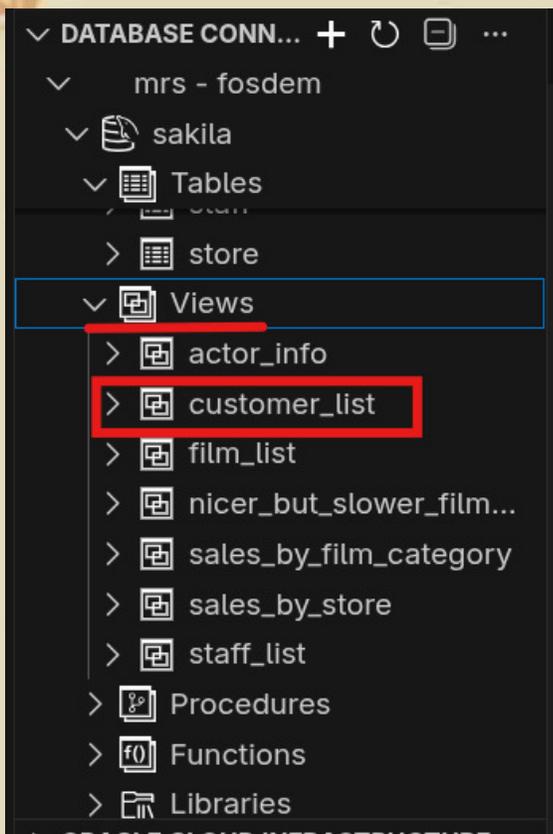
Using Views

Let's use one of the Views available in the Sakila database: `customer_list`

Using Views



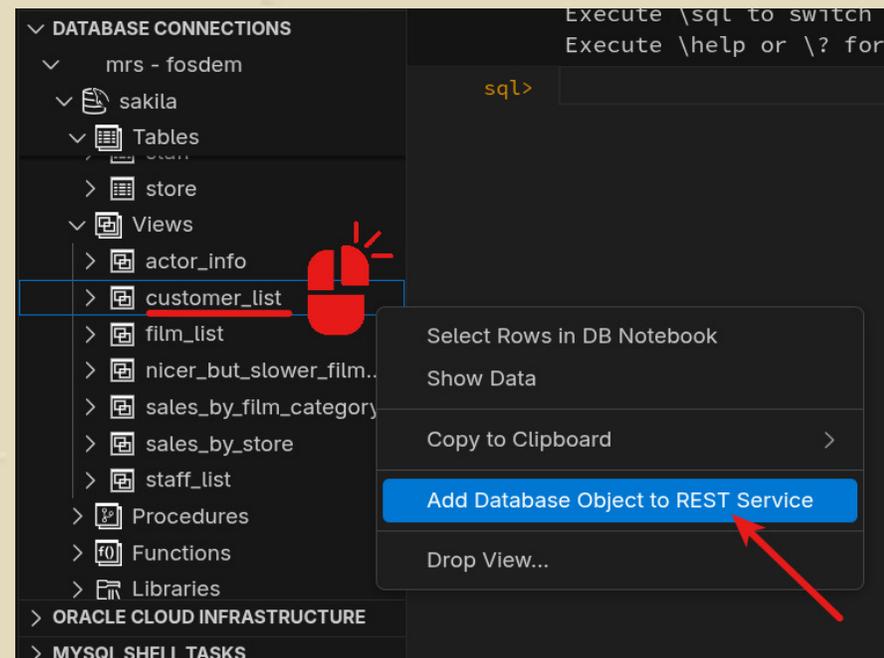
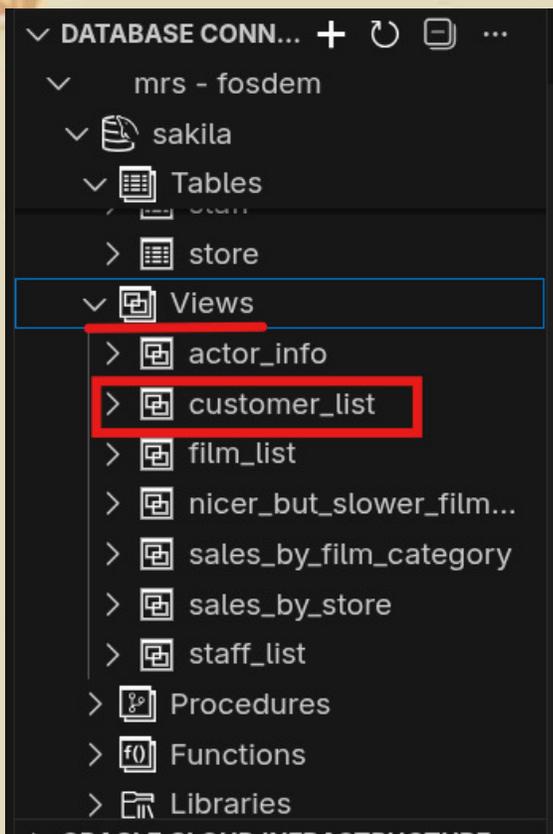
Let's use one of the Views available in the Sakila database: `customer_list`



Using Views



Let's use one of the Views available in the Sakila database: `customer_list`



Using Views (2)



MySQL REST Object

REST Service Path: /fosdem
REST Schema Path: /sakila
REST Object Path: /customerList
Access Control: Access ENABLED
Auth. Required:

Data Mapping Settings Authorization Options

DB Object: customer_list SDK Language SQL Preview

FosdemSakilaCustomerList {	...	sakila.customer_list	INSERT	UPDATE	DELETE
<input checked="" type="checkbox"/> ID	...	→ ● ID			
<input checked="" type="checkbox"/> name	...	→ ○ name			
<input checked="" type="checkbox"/> address	...	→ ● address			
<input checked="" type="checkbox"/> zip code	...	→ ○ zip code			
<input checked="" type="checkbox"/> phone	...	→ ● phone			
<input checked="" type="checkbox"/> city	...	→ ● city			

Cancel OK

Using Views (3)



Create a Python program to use the view and display the name and the county of customers.

Don't forget to export again the SDK !!

Using Views (3)



Create a Python program to use the view and display the name and the county of customers.

Don't forget to export again the SDK !!

```
$ python fosdem_view.py  
VERA MCCOY, Afghanistan  
MARIO CHEATHAM, Algeria  
JUDY GRAY, Algeria  
JUNE CARROLL, Algeria  
ANTHONY SCHWAB, American Samoa  
CLAUDE HERZOG, Angola  
MARTIN BALES, Angola  
...
```

Using Views (4)

fosdem_view.py

```
from sdk.fosdem import *
my_service = Fosdem(verify_tls_cert=False)
async def main():
    await my_service.authenticate(
        username = "fosdem_rest_user",
        password = "Fosd3m2026!",
        app = "MRS",
    )
    records = await my_service.sakila.customer_list.find()
    for record in records:
        print( record.name + ", " + record.country)

asyncio.run(main())
```

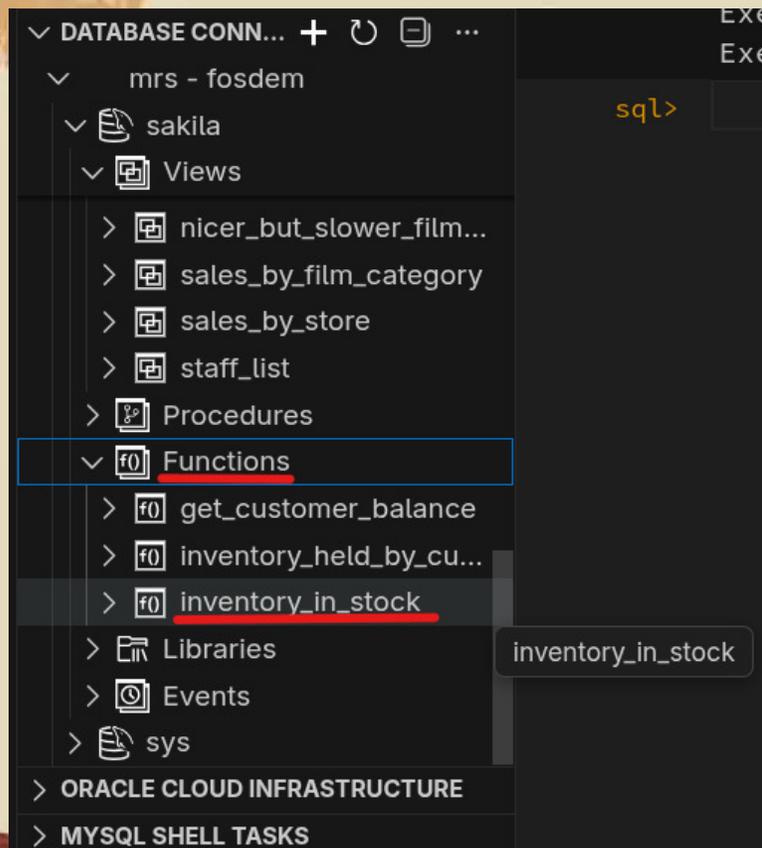
Using Routines

Now we will use a function from the Sakila database: `inventory_in_stock`

Using Routines



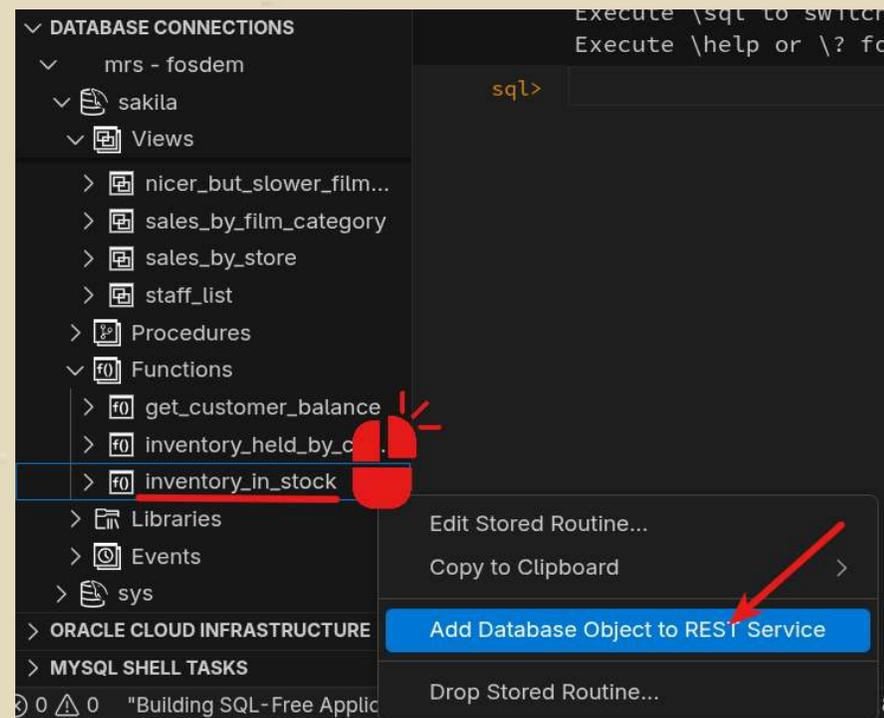
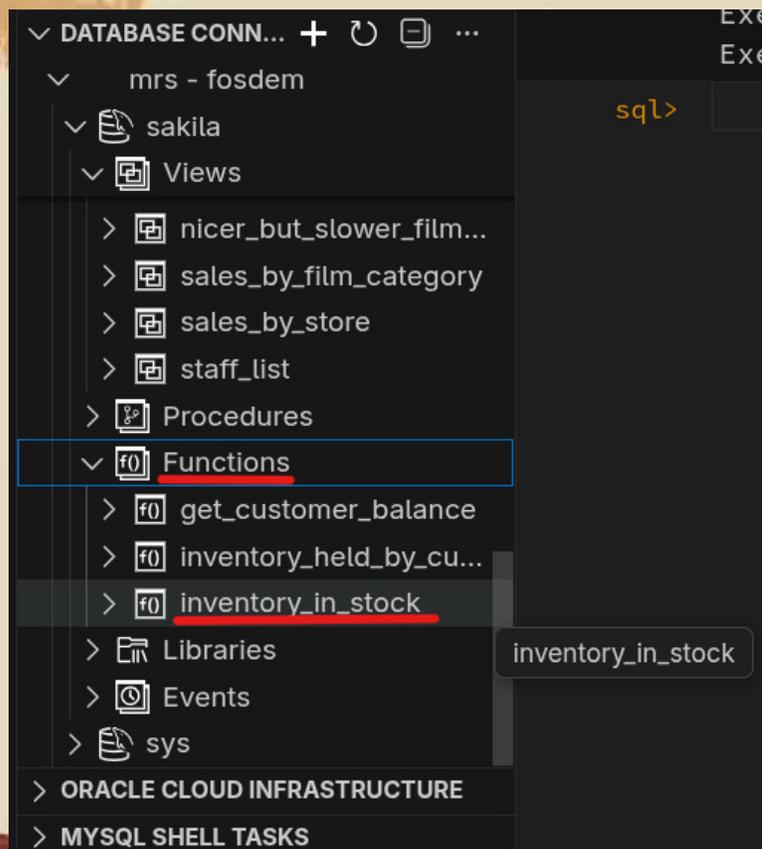
Now we will use a function from the Sakila database: `inventory_in_stock`



Using Routines



Now we will use a function from the Sakila database: `inventory_in_stock`



Using Routines (2)



MySQL REST Object

REST Service Path: /fosdem
REST Schema Path: /sakila
REST Object Path: /inventoryInStock
Access Control: Access ENABLED
 Auth. Required

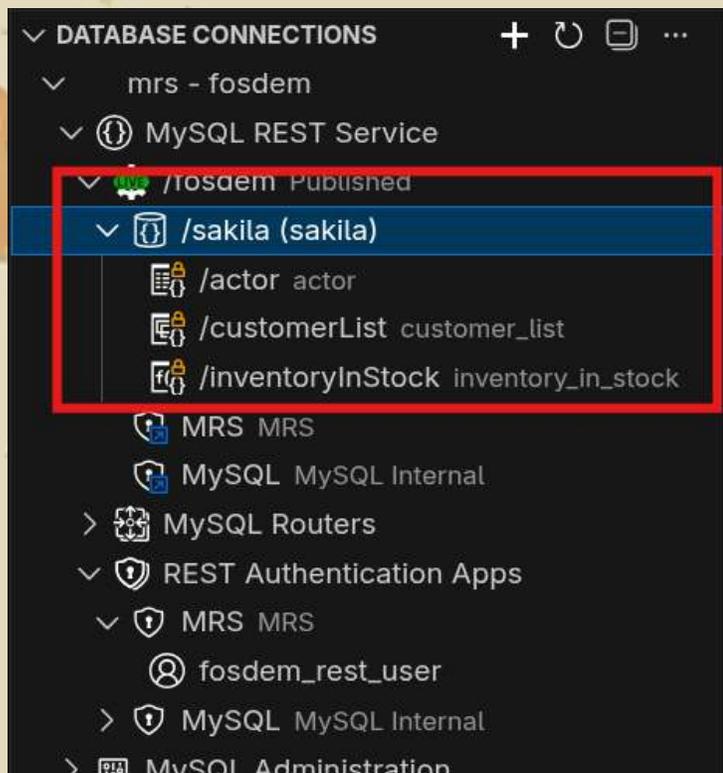
Parameters/Result Sets Settings Authorization Options

DB Object: inventory_in_stock Parameters SDK Language SQL Preview

```
FosdemSakilaInventoryInStockParams {  
  sakila.inventory_in_stock  
  ✓ pInventoryId → ○ p_inventory_id  
}
```

Cancel OK

Using the SDK



Don't forget to export the SDK at every changes in your service!

Using Routines (3)



Now create a Python program to call the function and display if the movie with a given id is in stock or not.

```
$ python fosdem_function.py
Movie id? 1
Movie in stock
Movie id? 6
Movie not in stock
Movie id?
```

Using Routines (4)

fosdem_function.py

```
from sdk.fosdem import *
my_service = Fosdem(verify_tls_cert=False)
async def main():
    await my_service.authenticate(
        username = "fosdem_rest_user",
        password = "Fosd3m2026!",
        app = "MRS",
    )
    movie_id = input("Movie id? ")
    records = await my_service.sakila.inventory_in_stock.call(p_inventory_id=int(movie_id))
    if records == 0:
        print("Movie not in stock")
    else:
        print("Movie in stock")

while True:
    asyncio.run(main())
```

Challenge for the Community Dinner ;)



Run the following query:

```
SQL > update film set language_id=5 where film_id = round(rand()*100);
```

Challenge for the Community Dinner ;)



Run the following query:

```
SQL > update film set language_id=5 where film_id = round(rand()*100);
```

Create an unauthenticated RESTful Endpoint that will return the list of films in a given languageId (5).



Share your ❤️ to MySQL

#mysql #MySQLCommunity



Join our slack channel!

bit.ly/mysql-slack

Questions ?

