

VP Softwaretechnologien WS2016/2017

SAP HANA

*High Performance **Analytic Appliance***

Dr. Schwaiger Roland

Vorstellung



roland.schwaiger@facet.at

Dr. Roland Schwaiger

Located

Bad Dürrenberg, Hallein, AT

Background

Mathematics (University Salzburg)

Computer Sciences (University Salzburg, Bowling Green State University)

Project & Process Management (SMBS – University of Salzburg Business School)

Profession

SAP Technical Consultant (Cert. SAP Development Consultant)

SAP Trainer

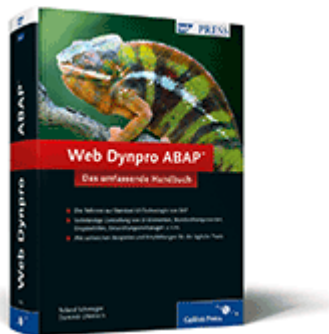
Project Coach (Cert. Scrum Master)

Software Architect

Software Developer (SAP AG, Walldorf, DE and Customer Development Projects)

Author (check out Amazon and/or www.citeseer.com)

Lecturer (University Salzburg, FH Salzburg)



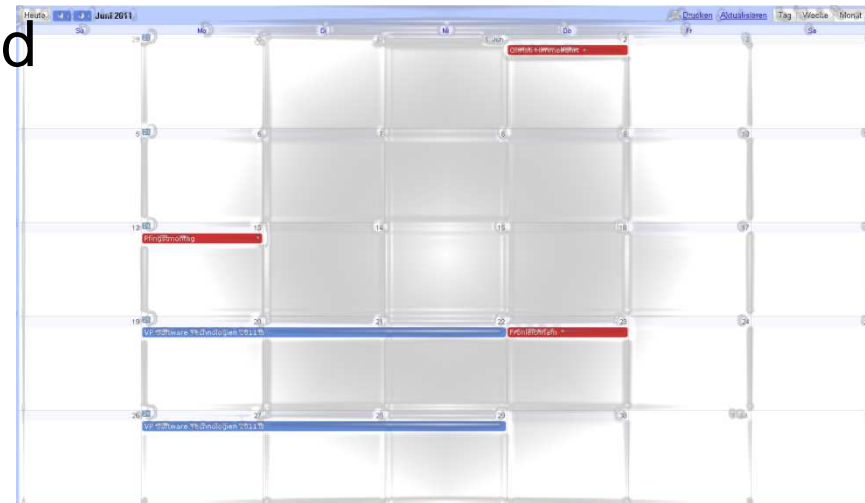
Inhalte/Organisation

1. Motivation
2. LV Überblick
3. Abschluss LV

Blockveranstaltung (www.facet.at)

Datum: 01.02 – 03.02

Zeit: 09:00-12:00 und
13:00-17:00



Motivation

Die drei Schwerpunkte, die wir behandeln werden sind:

1. **SAP HANA Entwicklung**
2. **oData (REST) Service Erstellung im SAP Backend**
3. **Web Anwendungen** mit Hilfe von SAP UI5

und das Schritt für Schritt in hoffentlich leicht verdaulicher Art und Weise.

VP Overview

First things first Logon, Get to know each other, DB Setup, User Setup, Package creation, Hello World, ...

Introduction to HANA Infrastructure, SAP HANA Architecture, Web IDE, Web Based Dev Tool, ...

Development Tools Eclipse, Web IDE, HANA ...

UIs with SAP UI5 UI5 Model, View, Controller, I18N, ...

SAP HANA Extended Application Services (XS) Packages, Schema, Roles,

The Persistence Model CDS,

oData Services

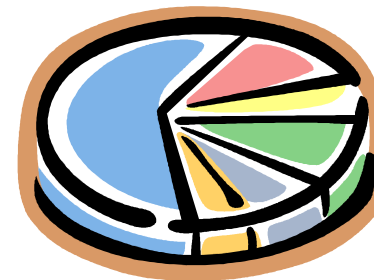
SQLScript

Server Side Java Script

Connectivity 4 HANA XS

? IoT

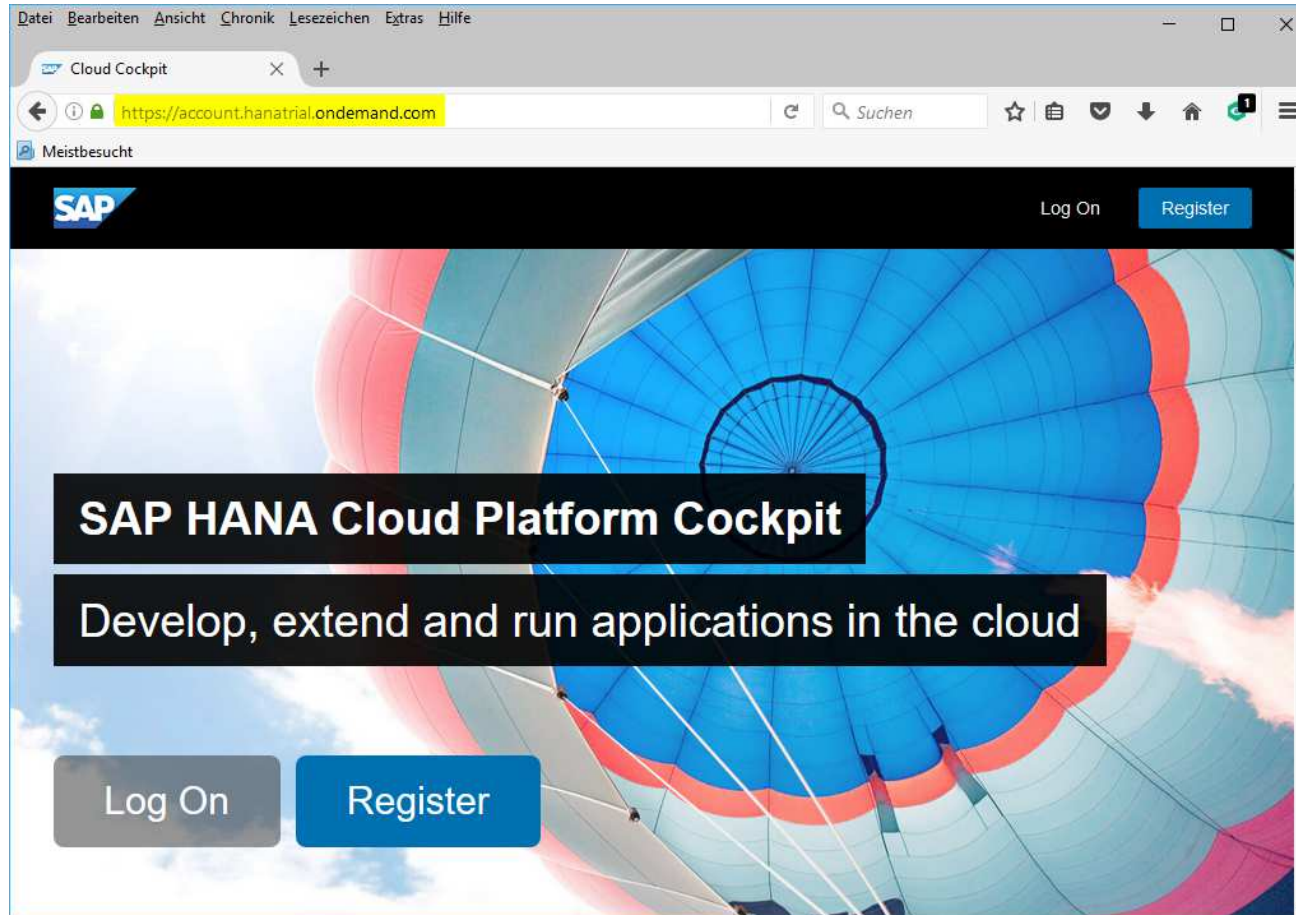
? Honeycomb



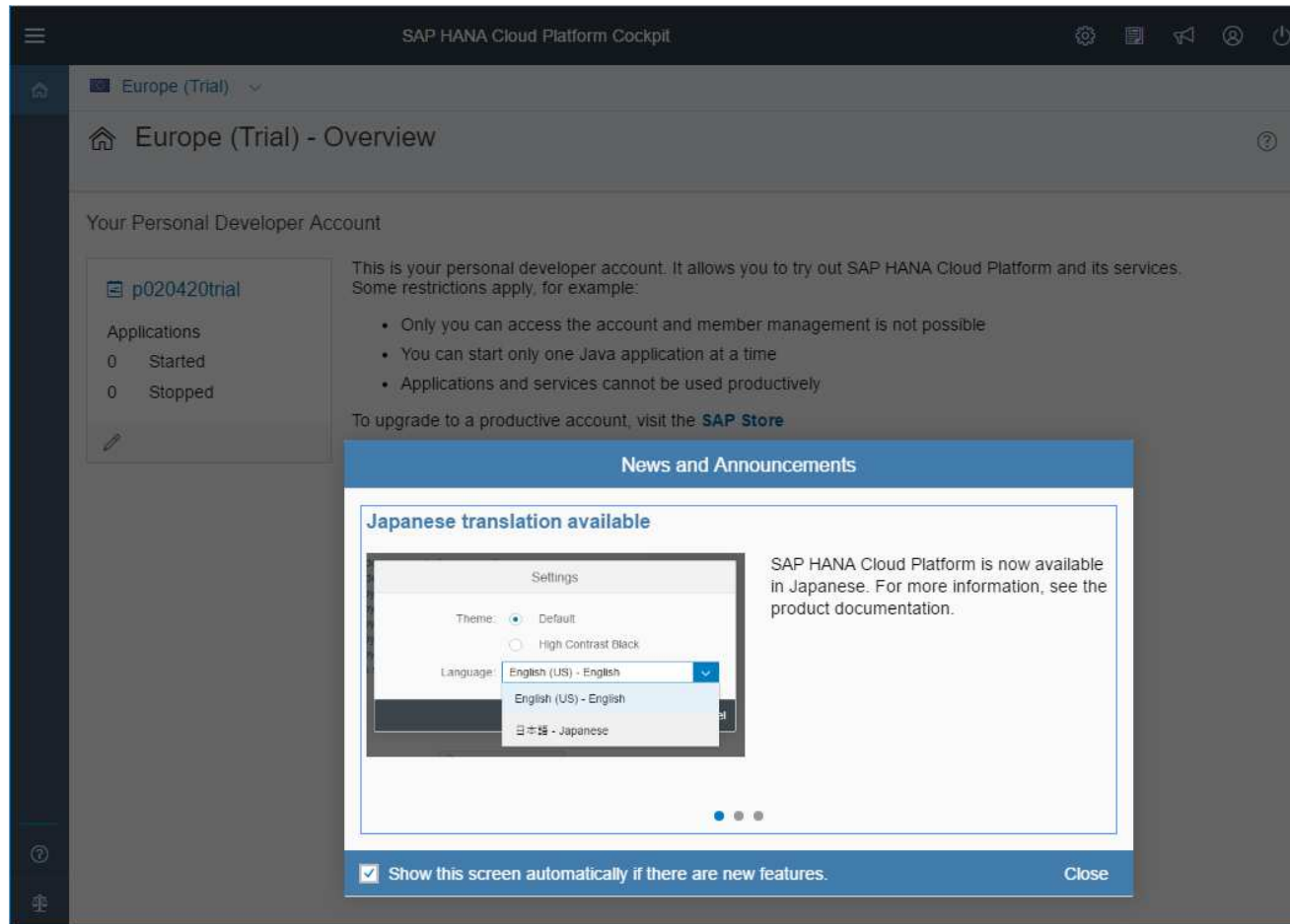
FIRST THINGS FIRST!

Get prepared for development

HANA Login



Entry Screen



Navigate to SAP HANA Cloud Platform Cockpit

The image displays two screenshots of the SAP HANA Cloud Platform Cockpit interface. The left screenshot shows the 'Overview' page for a 'Europe (Trial)' account. A yellow box highlights the account ID 'p020420trial' in the 'Your Personal Developer Account' section. A yellow arrow points from this box to the 'Applications' menu item in the right screenshot. The right screenshot shows the 'System Status' page for the same account, displaying health indicators for JAVA, HTML5, and DATABASE SYSTEMS.

System Status

Category	Overall Health	Details
JAVA	No applications yet	Get started here! or... Create an application Run sample applications
HTML5	OK	8 Applications 6 Started 2 Stopped
DATABASE SYSTEMS	N/A	0 Databases

Favorite Applications

You have not yet defined any favorite applications. You can add them from the [Java Applications list](#)

Account Information

SETUP MULTITENANT DATABASE CONTAINER (MDC)

- Create an MDC instance
- Create the SYSTEM user

Setup MDC

Create DB

SAP HANA Cloud Platform Cockpit

Europe (Trial) / p020420trial

p020420trial - Databases & Schemas

All: 2

New

DB/Schema ID

facetshared

ws201617

New Database/Schema

*Database ID: honeycomb8820038

Database System: HANA MDC (<trial>)

*SYSTEM User Password:

*Repeat Password:

Configure User for SHINE: **ON**

*SHINE User Name: SHINE

*SHINE User Password:

*Repeat password:

Parameters

Web Access: **ON**

DP Server: **ON**

Save Cancel

Setup MDC

Create DB - Steps

SAP HANA Cloud Platform Cockpit

Europe (Trial) / p1942292818trial / honeycomb8820038

honeycomb8820038 - Events

All: 3

Time	ID	Event	Description	User
31 Jan 2017, 15:22:49	1	Create	Step 2 of 3 finished	P1942292818
31 Jan 2017, 15:20:19	1	Create	Step 1 of 3 finished	P1942292818
31 Jan 2017, 15:18:37	1	Create	Database creation started	P1942292818

SAP HANA Cloud Platform Cockpit

Europe (Trial) / p1942292818trial / honeycomb8820038

honeycomb8820038 - Events

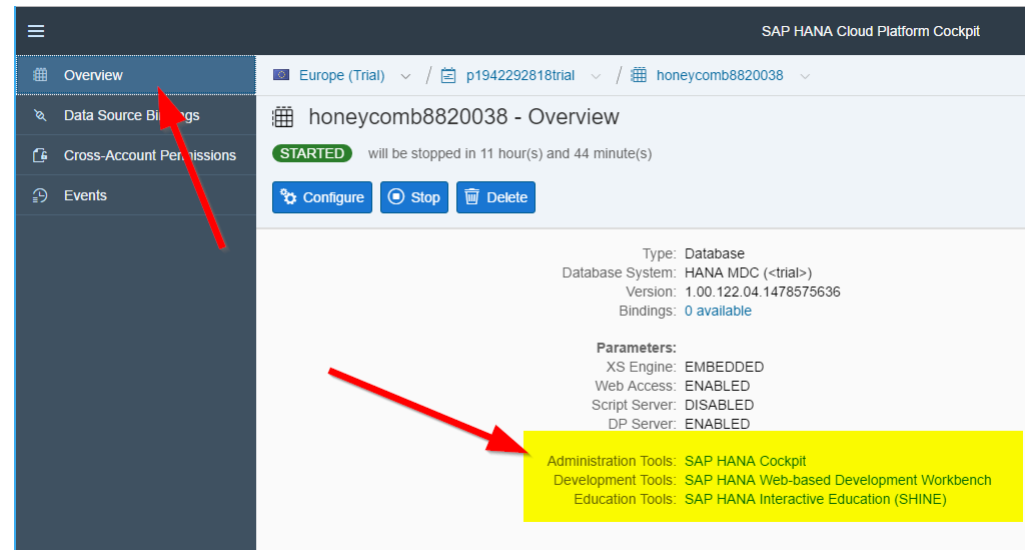
All: 16

Time	ID	Event	Description	User
31 Jan 2017, 15:27:55	1	Change	HTTPS access from Internet enabled	P1942292818
31 Jan 2017, 15:27:52	1	Change	Web dispatcher configuration added successfully	P1942292818
31 Jan 2017, 15:27:51	1	Start	Database started successfully	P1942292818
31 Jan 2017, 15:26:48	1	Start	Starting database	P1942292818
31 Jan 2017, 15:26:48	1	Change	scriptserver service is disabled	P1942292818
31 Jan 2017, 15:26:48	1	Change	dpserver service is enabled	P1942292818
31 Jan 2017, 15:26:48	1	Change	Security settings configured successfully	P1942292818
31 Jan 2017, 15:26:46	1	Stop	Database stopped successfully	P1942292818
31 Jan 2017, 15:26:20	1	Stop	Stopping database	P1942292818
31 Jan 2017, 15:26:20	1	Change	SHINE user is successfully created	P1942292818
31 Jan 2017, 15:26:20	1	Change	Statistics server checks are disabled	P1942292818
31 Jan 2017, 15:26:19	1	Create	Database created successfully	P1942292818
31 Jan 2017, 15:26:19	1	Create	Step 3 of 3 finished	P1942292818
31 Jan 2017, 15:22:49	1	Create	Step 2 of 3 finished	P1942292818
31 Jan 2017, 15:20:19	1	Create	Step 1 of 3 finished	P1942292818
31 Jan 2017, 15:18:37	1	Create	Database creation started	P1942292818

Setup MDC

Select DB

- Switch to Overview
- Or
- Select the newly created SAP HANA MDC database in the list of databases



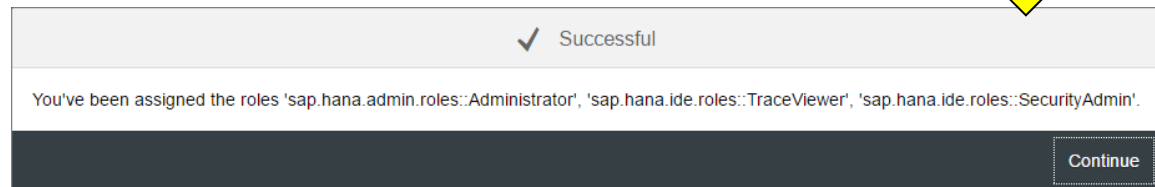
Setup MDC

Open SAP HANA Cockpit

- In the overview that is shown in the lower part of the screen, open the **SAP HANA Cockpit** link under **Administration Tools**.



- In the Enter Username field, enter **SYSTEM**, then enter the **password** you determined for the SYSTEM user in the Enter Password field.



SAP HANA Cockpit

SAP HANA Database Administration

The screenshot displays the SAP HANA Cockpit interface for database administration. The browser address bar shows the URL: <https://ws201617p020420trial.hanatrial.ondemand.com/sap/hana/uis/clients/ushell-app/shells/fiori/FioriLaunchpad.html?siteId=sap|hana|admin|cockpit|app|cockpit>. The interface features a teal header with the SAP logo and a 'SYSTEM' dropdown menu. The main content area is titled 'SAP HANA Database Administration' and contains several tiles:

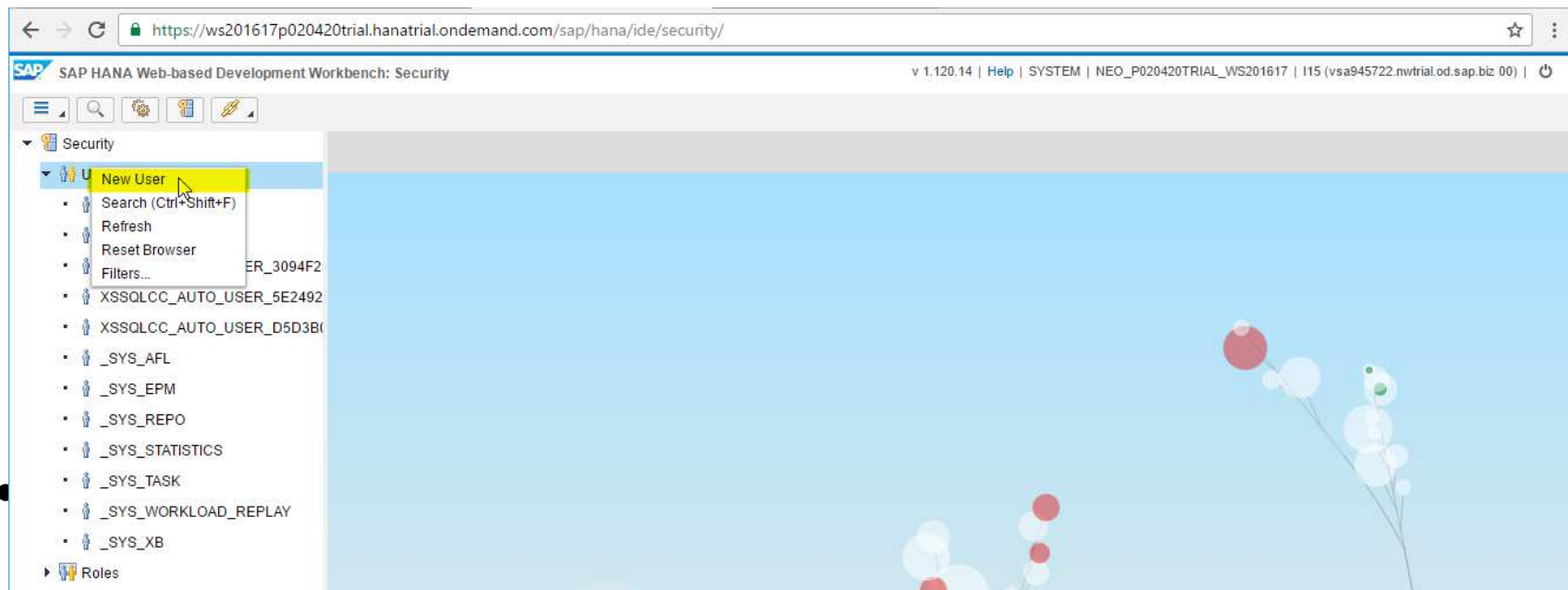
- Manage Services:** Overall Database Status - 115 - 1 Host. Shows 'Running with issues' (orange triangle), 'Services not running: 0', and 'Services running: 6'. A red alert icon indicates '1 related alert'.
- Alerts:** High Priority: 1, Medium Priority: 0.
- Used Memory:** 24.82 GB (bar chart), 230.22 GB (total). CPU Usage: 4% (line chart).
- Disk Usage:** 5 Disks (bar chart).
- User Tables:** High Priority: 0, Medium Priority: 0. Table Alerts.
- Monitor Statements:** Long-Running Statements: 0, Long-Running Blocking Situations: 0.
- Manage Workload Classes:** Active: 9, Blocked: 0.
- Service Restarts:** Restarted Services: 0.
- Number of Dumps:** Dumps: 0.
- General Information:** SAP HANA Version: 1.00.122.04.1478575636, Last Update: 14.12.16, 14:16, Platform: SUSE Linux Enterprise ...
- Configure Alerts:** (gear icon)
- Manage Roles and Users:** (highlighted with a red box, icon of a person with a lock)
- SAP HANA Cockpit for Offline Administration:** (document icon)
- SAP HANA Documentation Database Administration:** (document icon)

SYSTEM user

- Add roles:
 - sap.hana.xs.ide.roles::EditorDeveloper
 - sap.hana.xs.ide.roles::Developer
 - sap.hana.xs.lm.roles::Administrator
 - *sap.hana.xs.debugger::Debugger*
 - *sap.hana.xs.admin.roles::HTTPDestViewer*
 - *sap.hana.xs.admin.roles::HTTPDestAdministrator*
 - *sap.hana.xs.admin.roles::TrustStoreViewer*
 - *sap.hana.xs.admin.roles::TrustStoreAdministrator*

Create User

- Choose **Manage Roles and Users**.
- To create database users and assign them the required roles, expand the **Security** node.
- Open the context menu for the **Users** node and choose **New User**.



Create User

- On the **User** tab, provide a name for the new user. The user name always appears in upper case letters.
- In the **Authentication** section, make sure the **Password** checkbox is selected and enter a password. Note: The password must start with a letter and only contain uppercase and lowercase letters ('a' - 'z', 'A' - 'Z'), and numbers ('0' - '9').
- To create the user, choose **Save**. The new database user is displayed as a new node under the **Users** node.

Create User

The screenshot displays the SAP HANA Web-based Development Workbench Security interface. The main window is titled "New User" and contains the following fields and options:

- User Name*:** 8820038
- Restricted user:**
- Authentication:**
 - Password
 - SAML
 - SAP Logon Ticket
 - Kerberos
 - X509
 - SAP Assertion Ticket
- External ID*:** [Empty field]
- Valid From:** [hh:mm:ss]
- Valid Until:** [hh:mm:ss]
- Session Client:** [Empty field]

The "Granted Roles" section is visible at the bottom of the form. A yellow arrow points from the "New User" form to a smaller inset window showing a list of users in the "Users" folder. The user "8820038" is highlighted in blue in the list.

Grant Roles

- To assign your user the roles with the required permissions for working with SAP HANA Web-based Development Workbench, go to the Granted Roles section and choose the + (Add Role) button.
- Type **xs.ide** in the search field and select all roles in the result list.
- Choose Ok.

- The roles are added on the Granted Roles tab.
- Repeat the last two steps to assign the **CONTENT_ADMIN** role to the user.
- Save your changes.

- **Caution:** At this point, you are still logged on with the SYSTEM user. You can only use your new database user to work with **SAP HANA Web-based Development Workbench** by logging out from **SAP HANA Cockpit** first. Otherwise, you would automatically log in to the **SAP HANA Web-based Development Workbench** with the SYSTEM user instead of your new database user. Therefore, choose the **Logout** button before you continue to work with the **SAP HANA Web-based Development Workbench**, where you need to log on again with the new database user.

Grant Roles

The screenshot displays the SAP HANA Web-based Development Workbench: Security interface. The left sidebar shows a tree view of Security objects, including Users and Roles. The main area shows the configuration for user *8820038. The 'Granted Roles' tab is active, and a yellow arrow points to the '+' icon used to add a role. A 'Find Role' dialog box is open, showing a search for 'xs.ide' and a list of 5 matching roles: sap.hana.xs.ide.roles::CatalogDeveloper, sap.hana.xs.ide.roles::Developer, sap.hana.xs.ide.roles::EditorDeveloper, sap.hana.xs.ide.roles::SecurityAdmin, and sap.hana.xs.ide.roles::TraceViewer. The 'OK' button is highlighted.

SAP HANA Web-based Development Workbench: Security

Security

- Users
 - 8820038
 - SYS
 - SYSTEM
 - XSSQLCC_AUTO_USER_3094F2
 - XSSQLCC_AUTO_USER_5E2492
 - XSSQLCC_AUTO_USER_D5D3B0
 - _SYS_AFL
 - _SYS_EPM
 - _SYS_REPO
 - _SYS_STATISTICS
 - _SYS_TASK
 - _SYS_WORKLOAD_REPLAY
 - _SYS_XB
- Roles

User: 8820038

Authentication

- Password
- Password*: Confirm:
- Kerberos
- External ID*:

Valid From: Jan 3, 2017 11:05:59

Session Client:

Granted Roles

Role

- sap.hana.xs.ide.roles::TraceViewer
- sap.hana.xs.ide.roles::SecurityAdmin

Find Role

Type name to find a role:

xs.ide

5 item(s) matched

- sap.hana.xs.ide.roles::CatalogDeveloper
- sap.hana.xs.ide.roles::Developer
- sap.hana.xs.ide.roles::EditorDeveloper
- sap.hana.xs.ide.roles::SecurityAdmin
- sap.hana.xs.ide.roles::TraceViewer

OK Cancel

HANA Dev Workbench

- Open the SAP HANA Cloud Platform cockpit.
- Select the relevant database from the list and choose **SAP HANA Web-based Development Workbench** under Development Tools.
- Log on with your newly created user.
- **Note:** If you log on to the SAP HANA Web-based Development Workbench for the first time, you are prompted to change your initial password.

HANA Dev Workbench Call

The screenshot displays the SAP HANA Cloud Platform Cockpit interface. The top navigation bar shows the instance details: Europe (Trial), p020420trial, and ws201617. The left sidebar contains navigation options: Overview, Data Source Bindings, Cross-Account Permissions, and Events. The main content area shows the 'ws201617 - Overview' page. The instance status is 'STARTED' with a green indicator and a message 'will be stopped in 11 hour(s) and 0 minute(s)'. Below the status are three buttons: 'Configure', 'Stop', and 'Delete'. The instance details section lists the following information:

- Type: Database
- Database System: HANA MDC (<trial>)
- Version: 1.00.122.04.1478575636
- Bindings: 0 available

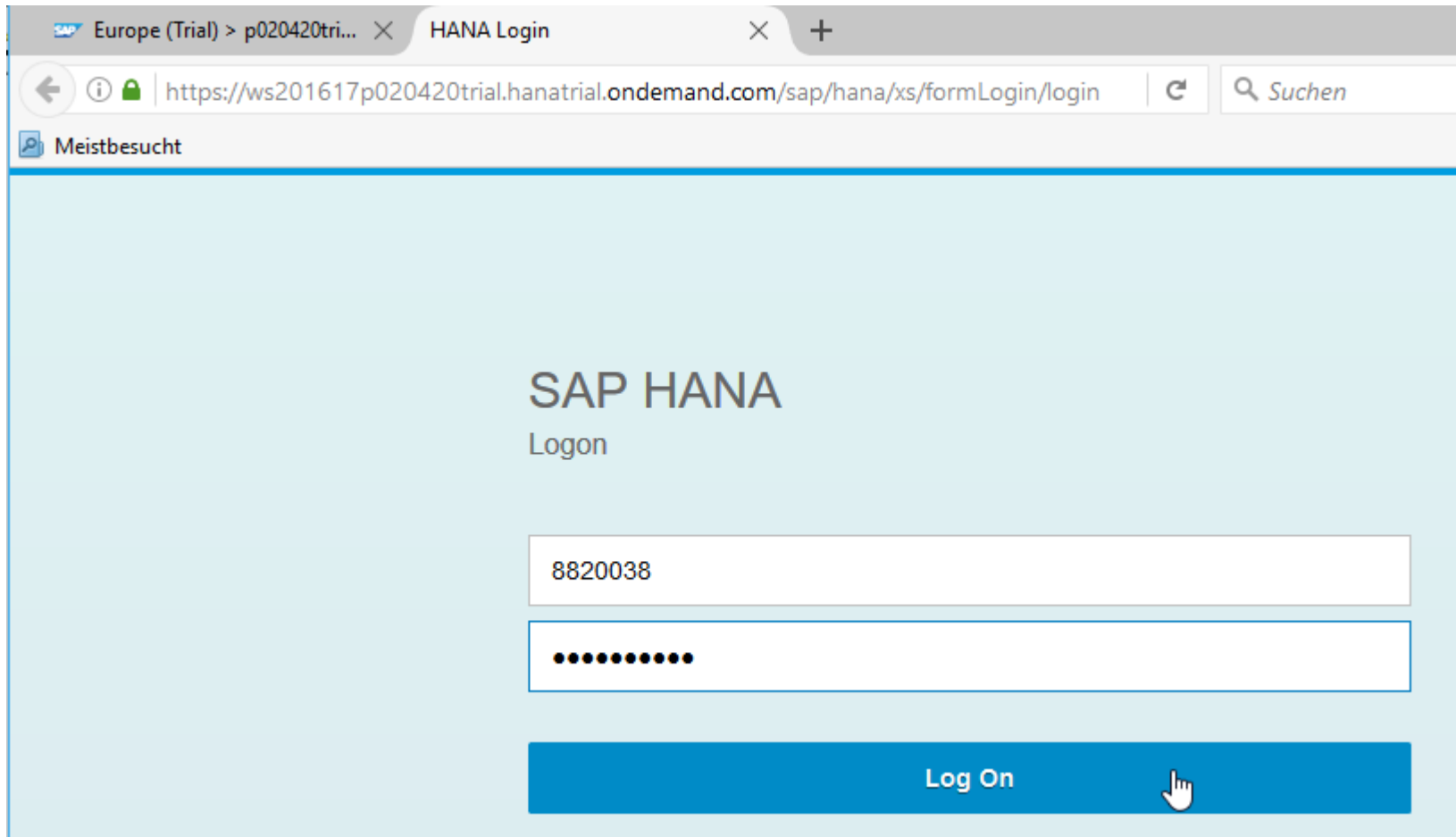
The 'Parameters' section lists the following settings:

- XS Engine: EMBEDDED
- Web Access: ENABLED
- Script Server: DISABLED
- DP Server: DISABLED

At the bottom, there are two links: 'Administration Tools: SAP HANA Cockpit' and 'Development Tools: SAP HANA Web-based Development Workbench', with the latter highlighted in yellow.

HANA Dev Workbench

Login with DB User



Europe (Trial) > p020420tri... HANA Login

https://ws201617p020420trial.hanatrial.ondemand.com/sap/hana/xs/formLogin/login

Meistbesucht

SAP HANA
Logon

8820038

.....

Log On

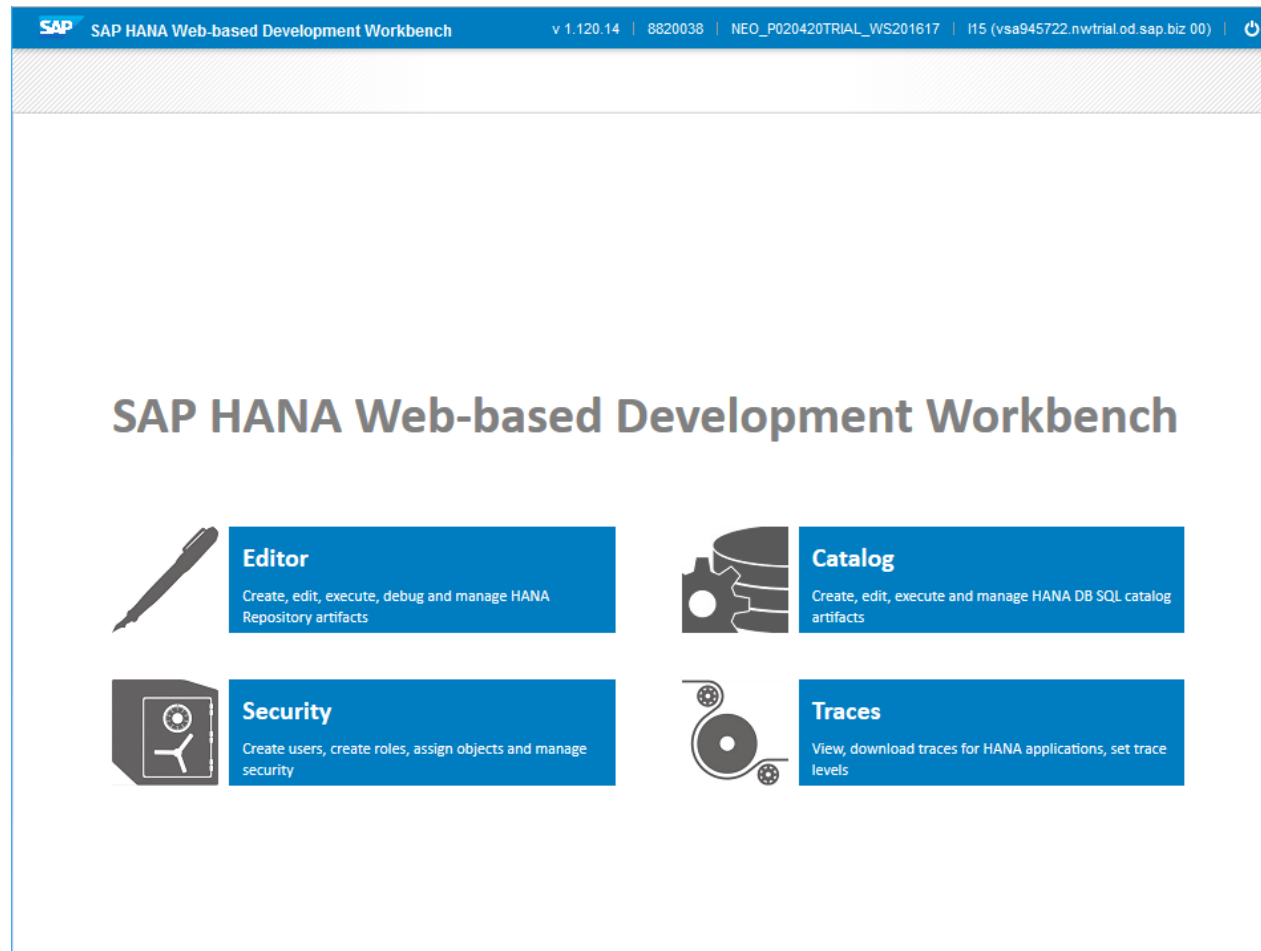
HANA Dev Workbench

Start

- Select the Editor.
- The editor is displayed. The header shows the details for your user and database. Hover over the entry for the SID to view the details.

HANA Dev Workbench

Start



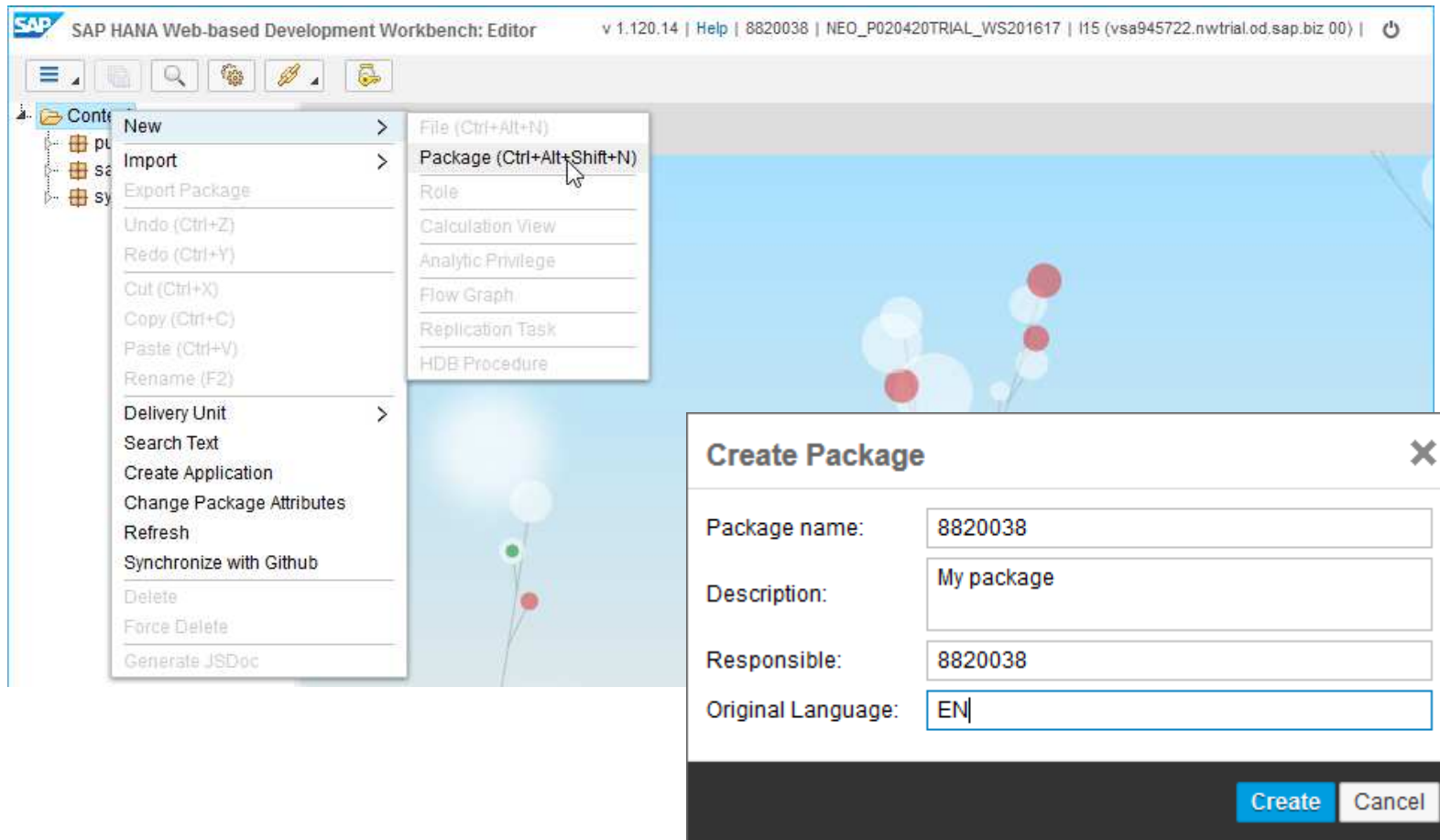
HANA Dev Workbench

Create New Package

- Create a new package by choosing **New Package** from the context menu for the **Content** folder.
- Enter a package name (! No Umlaute!)
- The package appears under the Content folder node.

HANA Dev Workbench

Create New Package



HANA Dev Workbench

Create New Application

- From the context menu for the new package node, choose **Create Application**.
- Select **HANA XS Hello World** as template and choose Create.
- Open the files under the new package hierarchy to view them in the editor.

- Only if you are using an SAP HANA MDC database: From the context menu for the new package node, choose **Activate All**.

HANA Dev Workbench

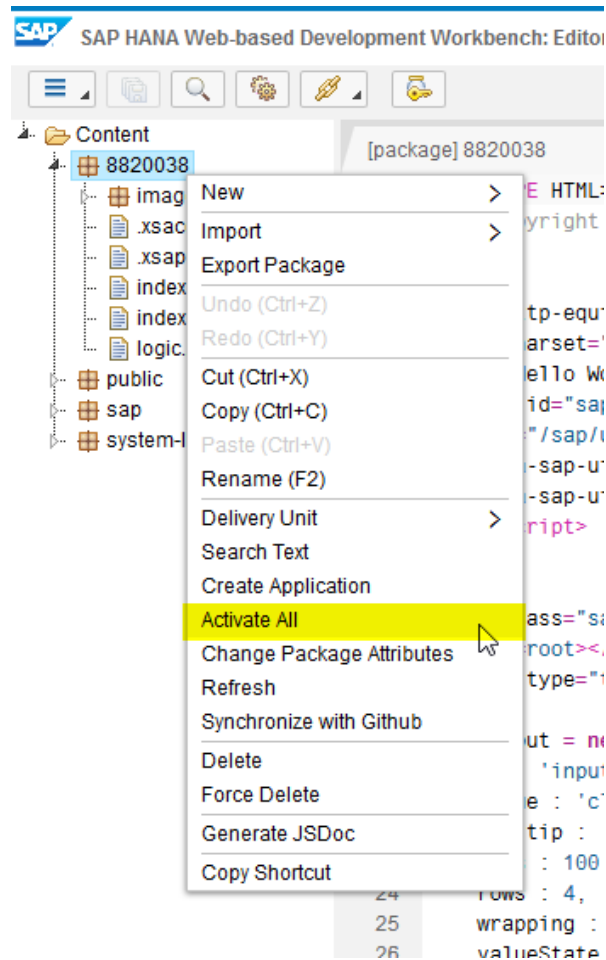
Create New Application

The image illustrates the process of creating a new application in the SAP HANA Web-based Development Workbench. It is divided into three main sections:

- Top Left:** A screenshot of the SAP HANA Web-based Development Workbench: Editor interface. The left-hand navigation pane shows a tree structure with folders like 'Content', '8820038', 'public', 'sap', and 'system-local'. A context menu is open over the '8820038' folder, with 'Create Application' highlighted. A yellow arrow points from this menu item to the 'Create Application' dialog.
- Top Right:** A screenshot of the 'Create Application' dialog box. The 'Template' dropdown is set to 'Application (with XSAccess and XSApp)'. The 'Package' dropdown is open, showing options: 'Empty application (with XSAccess and XSApp)', 'HANA XS Hello World', 'SAP UI5 Hello World', and 'Fiori Event Feedback'. A yellow arrow points from the 'HANA XS Hello World' option to the code editor below.
- Bottom:** A screenshot of the SAP HANA Web-based Development Workbench: Editor interface showing the newly created application. The left-hand navigation pane shows the folder structure, including 'indexUI5.html'. The main editor area displays the HTML code for 'indexUI5.html', which includes a DOCTYPE declaration, copyright information, and a title 'Hello World Demo with SAPUI5'. A yellow arrow points from the 'Create Application' dialog to this code editor.

HANA Dev Workbench

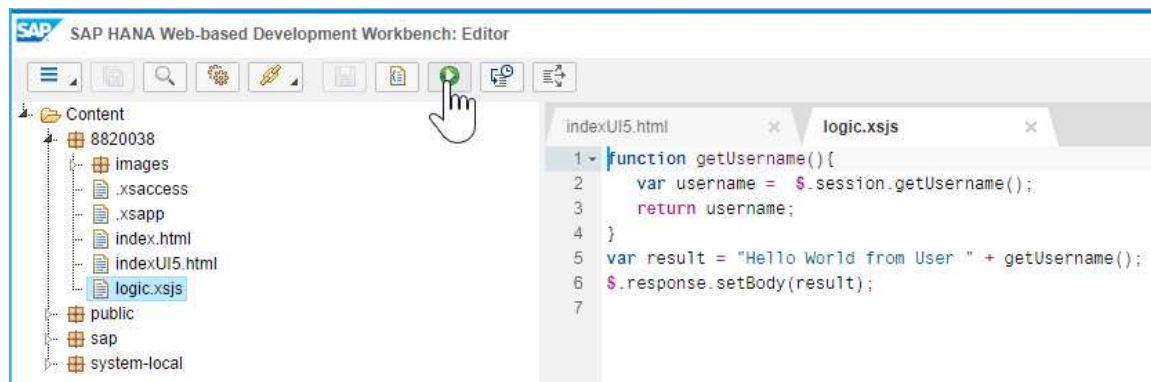
Activate All



HANA Dev Workbench

Test

- In the Editor of the SAP HANA Web-based Development Workbench, select the **logic.xsjs** file from the newly created package and choose **Run**.
- The program is deployed and displayed in the browser:
Hello World from User <Your User>.



- Note: If you have used an SAP HANA XS database for creating your SAP HANA XS application, you can also launch your application from the SAP HANA Cloud Platform cockpit.

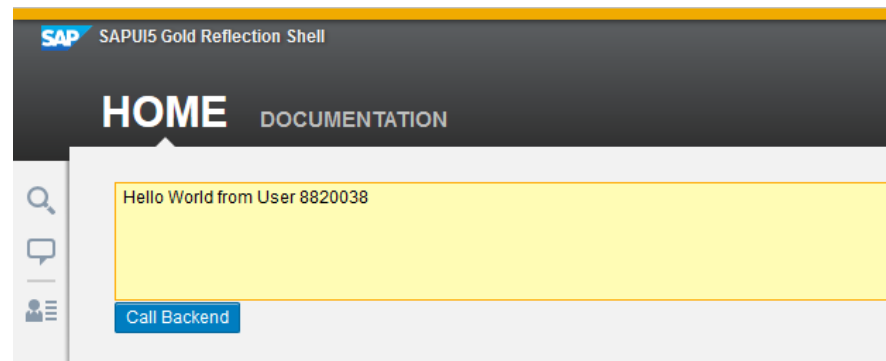
HANA Dev Workbench

Test Output

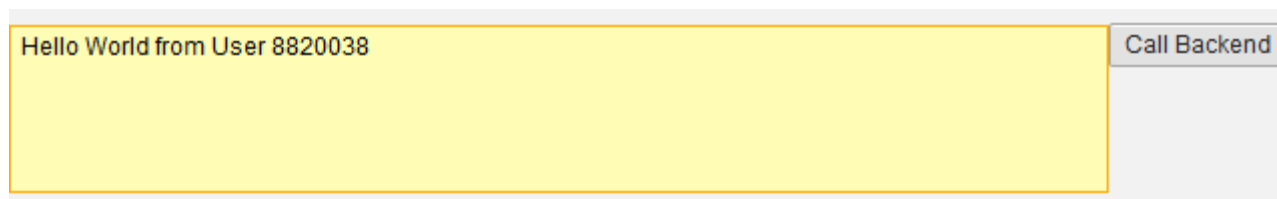
- logic.xsjs

Hello World from User 8820038

- indexUI5.html



- index.html



INTRODUCTION TO HANA

- Introduce SAP HANA
- Explain the system architecture and the impacts on development model
- Comparing the **XS Classic** and XS Advanced ecosystem

Introduction to HANA (New) HANA Paradigm

**„Do as much as you can in
the database to get the best
performance“**

Introduction to HANA

HANA Platform Overview

Open, standard interfaces

- Supporting all types of devices

Integrated Application Server Components

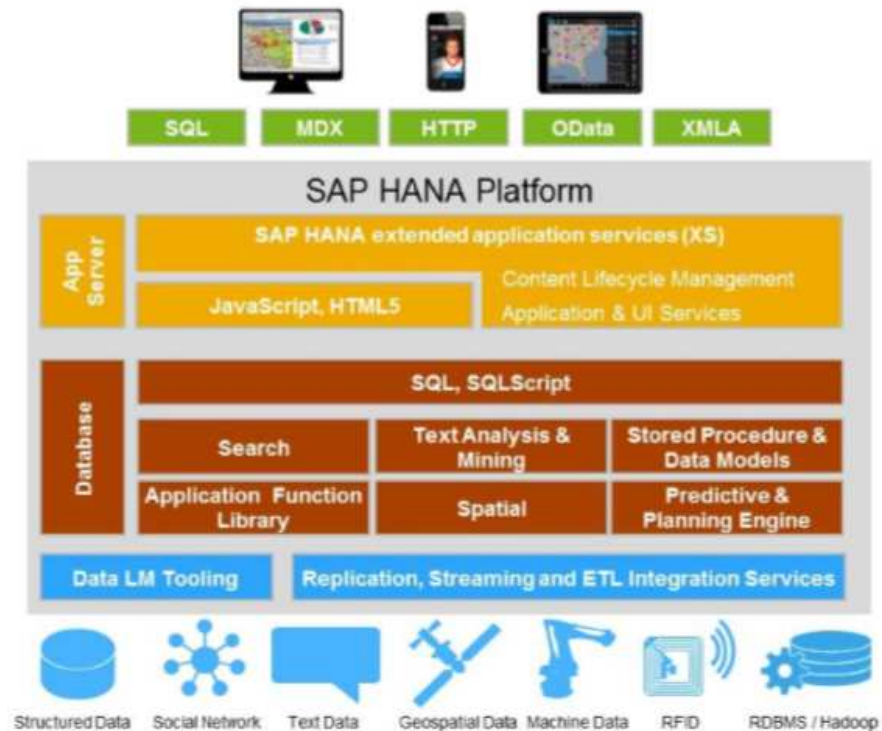
- Native application infrastructure

One DB for OLAP and OLTP Workloads

- With built-in functions for data-intensive processing

Data persistence and integration

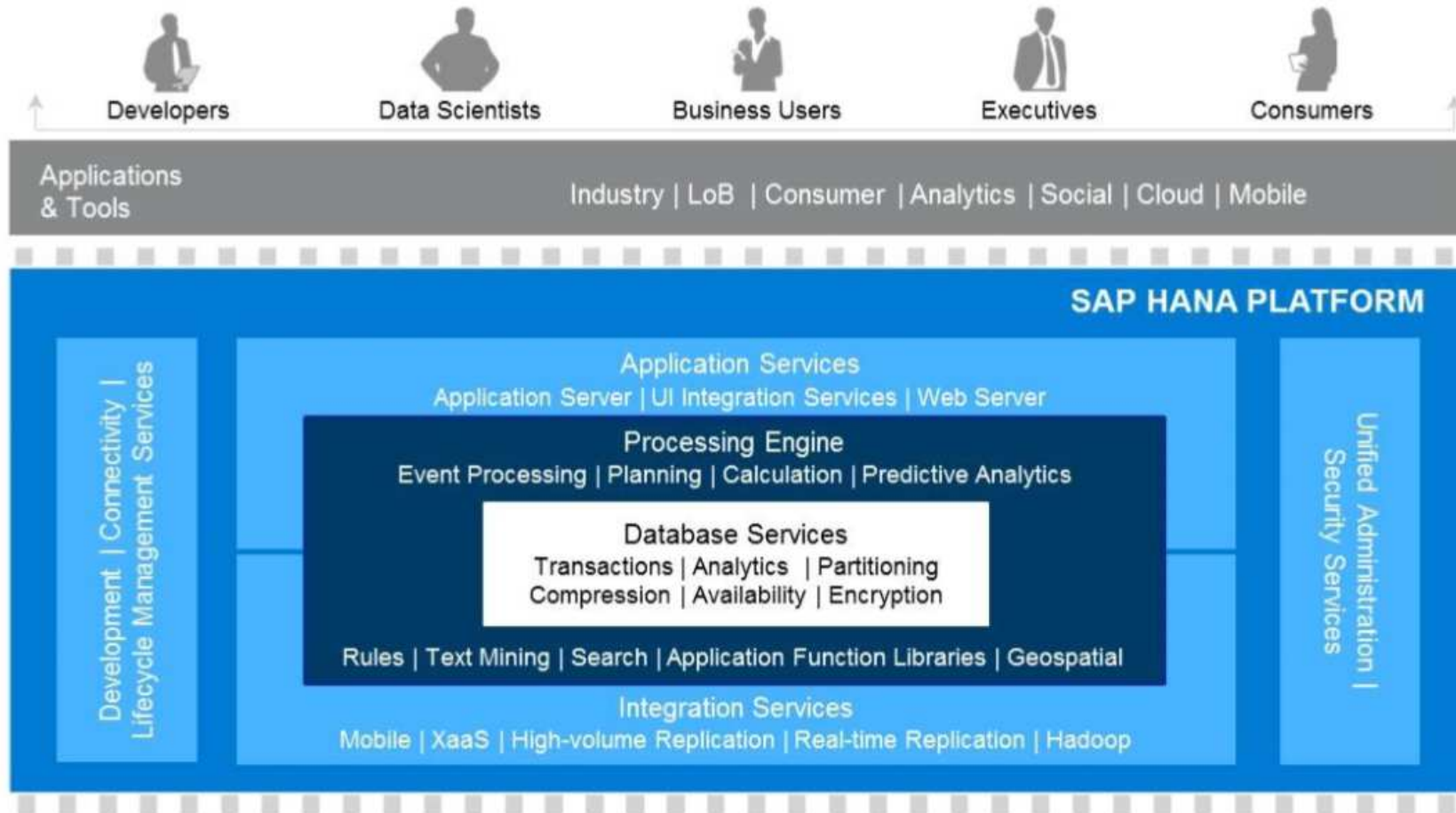
- Integrate any data from any source
- Ready for Big Data Scenarios



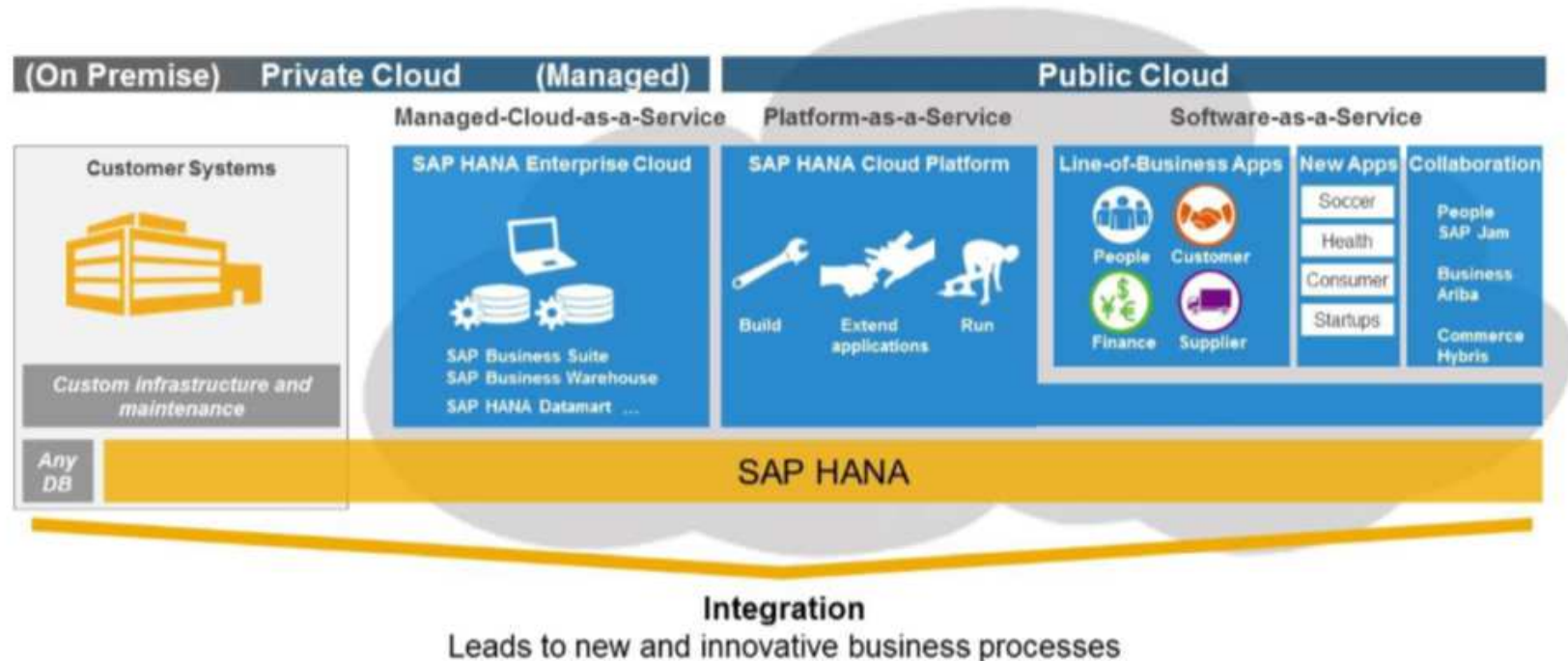
© SAP

Introduction to HANA

HANA Platform



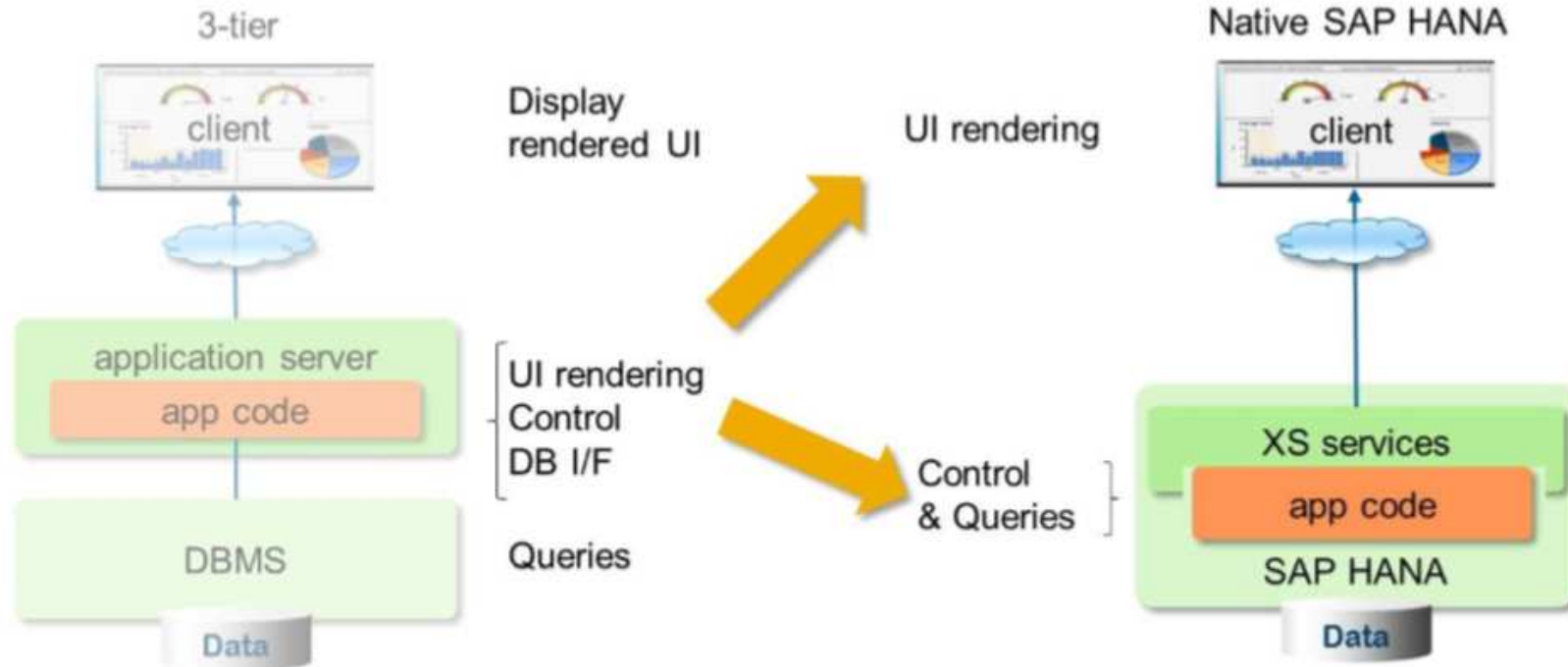
Introduction to HANA SAP Cloud



© SAP

Introduction to HANA

3-tier vs HANA Apps

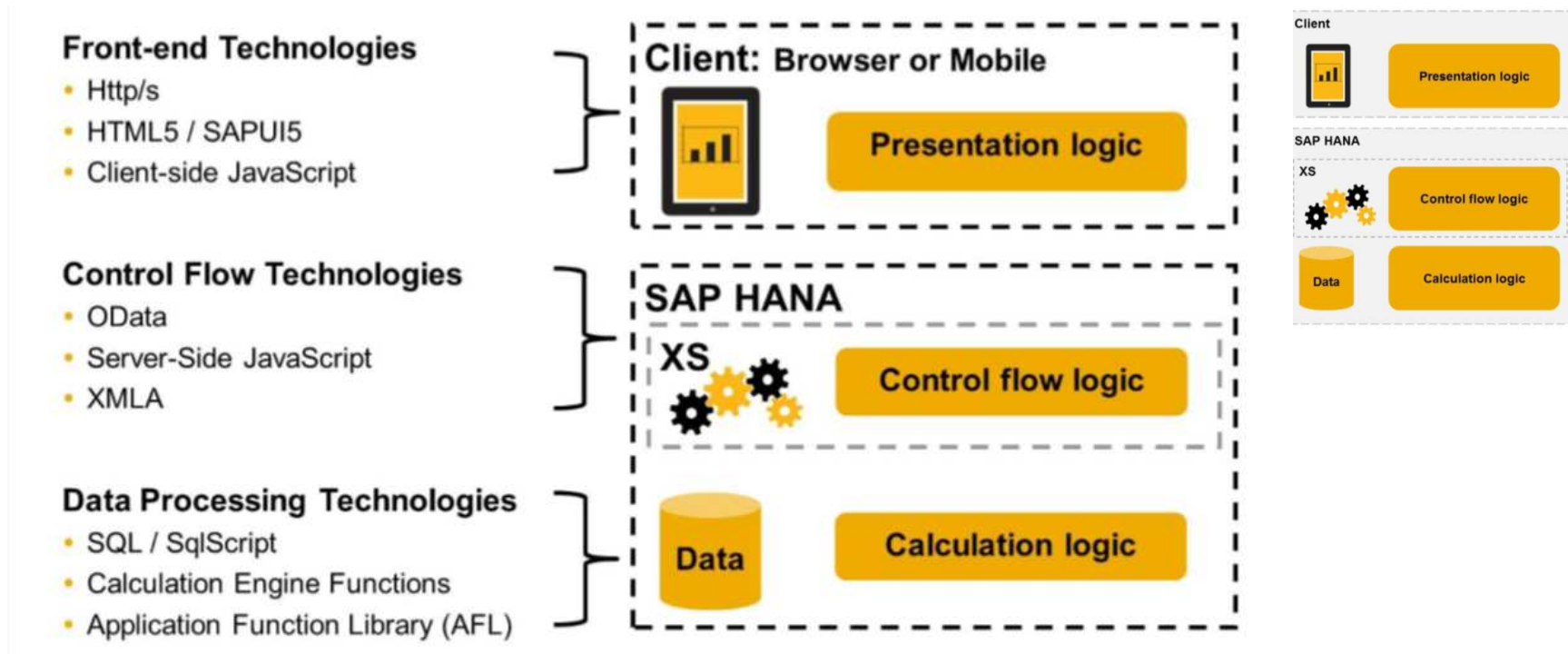


© SAP

Introduction to HANA

HANA Dev Model

SAP HANA Extended Application Services

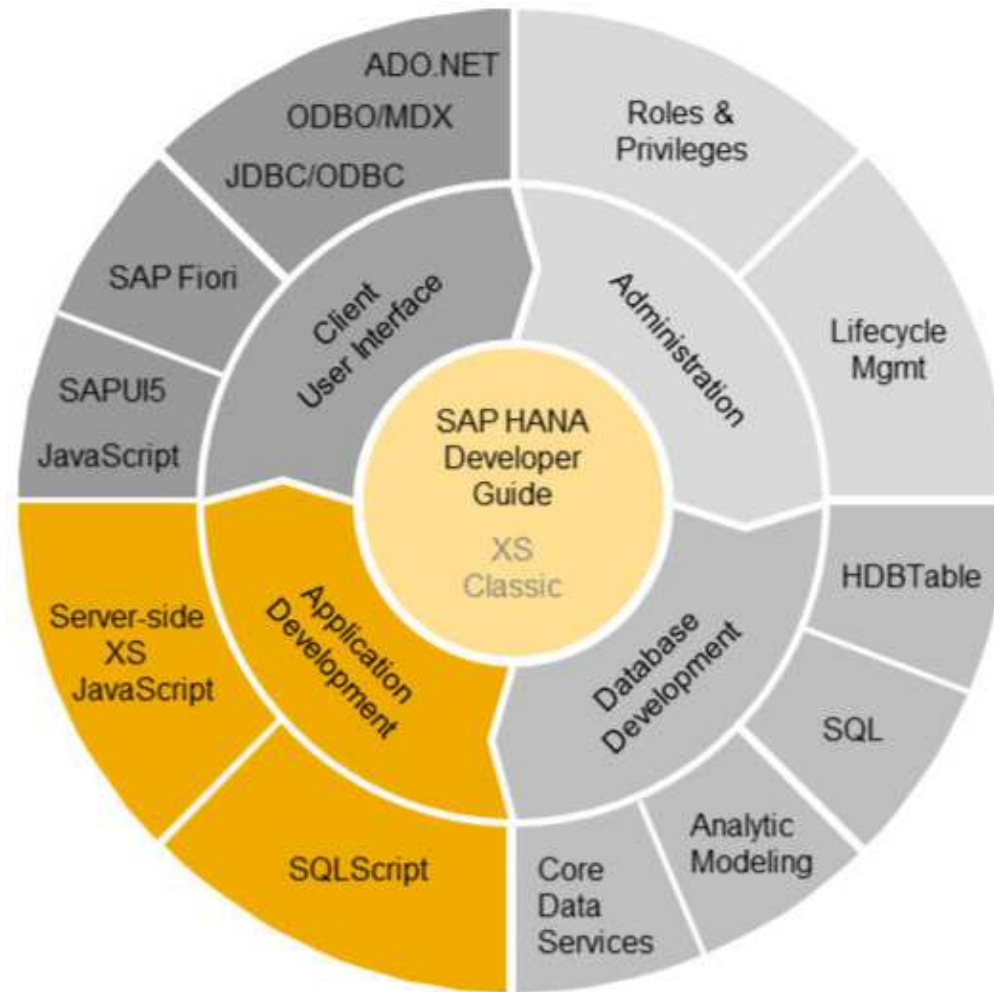


SAP HANA XS advanced model from SPS 11 on.

© SAP

Introduction to HANA

Dev in HANA XS Classical Model



DEVELOPMENT TOOLS

- Explain basic concepts of SAP HANA Studio
- Explain SAP HANA Web-based Development Workbench
- Explain XS Advanced Application Development Tools

Development Tools

SAP HANA Studio

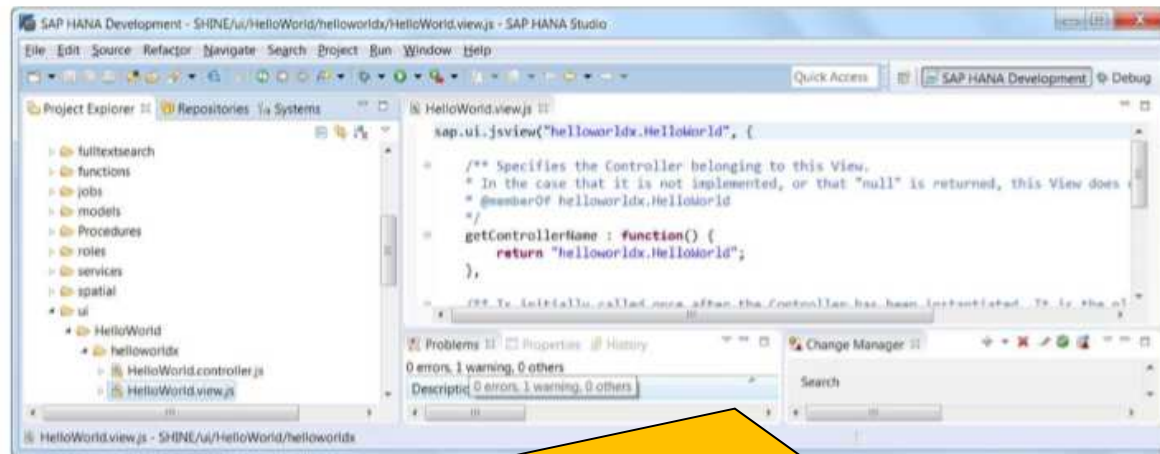
SAP HANA Studio

= *EclipseIDE for Java EE Developers + SAP Plugins*

Application Development in SAP HANA requires the use of an **Eclipse IDE**.

On top of Eclipse, the following SAP Plug-ins have to be installed:

- **SAP HANA Studio:** Used for the **server-side** development.
- **SAPUI5 Development Tools:** Used for the **client-side** development.



Update Site:

[https://tools.hana.ondemand.com/\[neon|mars\]](https://tools.hana.ondemand.com/[neon|mars])

© SAP

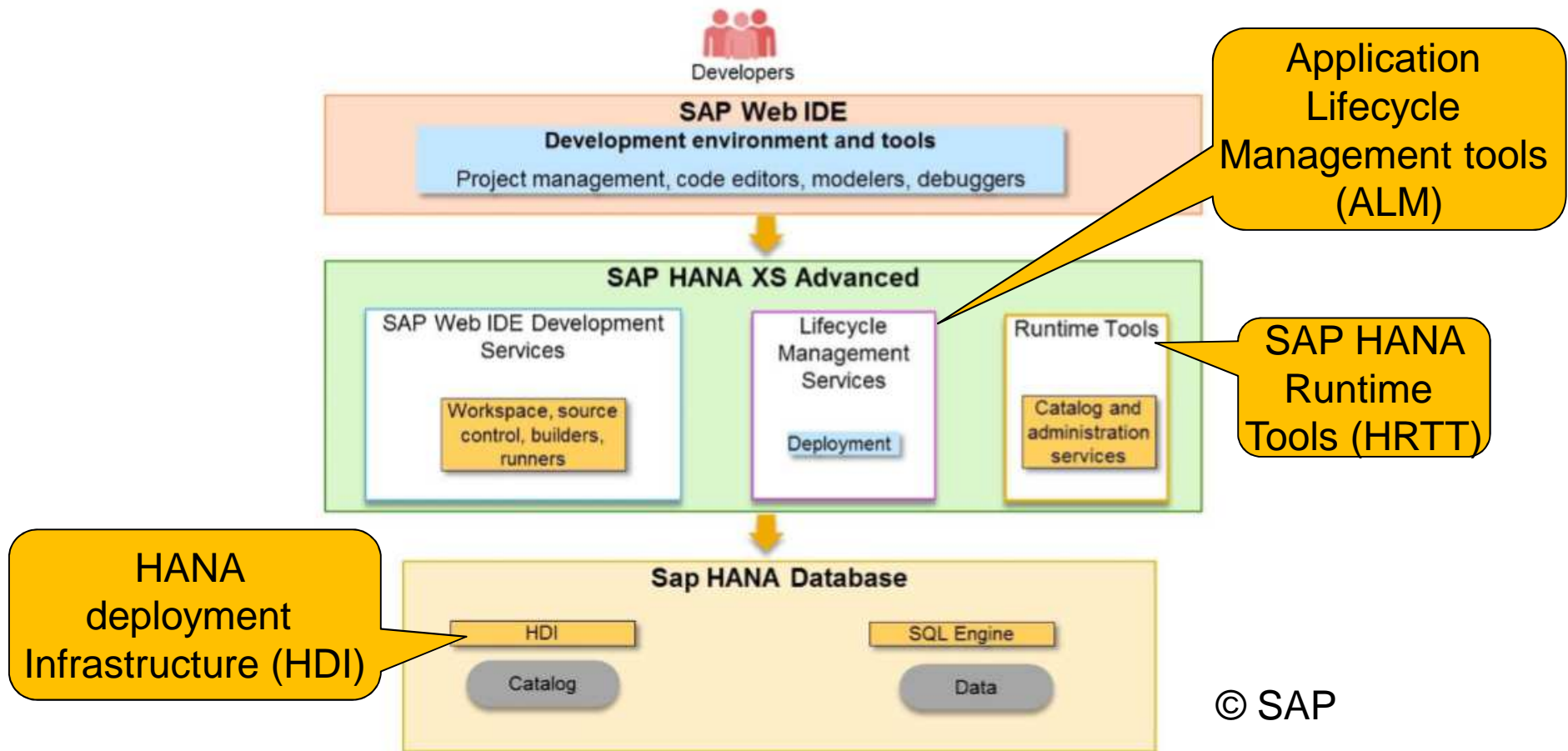
Development Tools

SAP HANA Dev Guide

http://help.sap.com/hana_platform

The screenshot displays the SAP Help Portal interface. At the top, the SAP logo and tagline 'The Best-Run Businesses Run SAP' are visible, along with a search bar and navigation links for 'Sitemap' and 'Global Search'. The main navigation menu includes categories like Analytics, Data Management, Human Capital Management, Supply Chain Management, Content and Collaboration, Enterprise Management, Product Lifecycle Mgmt, Technology Platform (highlighted), Customer Relationship Mgmt, Financial Management, Supplier Relationship Mgmt, and Additional Information. The breadcrumb trail shows the path: Technology Platform > SAP HANA Platform > SAP HANA Platform Core 2.0 SPS 00. The left sidebar lists various SAP products and services, with 'Development and Modeling' highlighted in yellow. The main content area features the title 'SAP HANA Platform (Core) 2.0' and the subtitle 'Support Package Stack 00 (Last Update: November 30, 2016, Revision 000)'. Below this, there is a list of links: 'What's New - Release Notes', 'Installation and Update', 'Security', 'System Administration', 'Development and Modeling' (highlighted), 'References', and 'Additional Information'. A section titled 'Available SAP Education Offerings' provides information about SAP courses, resources, and certifications, with a link to 'Curriculum SAP HANA'. At the bottom, there is a 'What's New - Release Notes' section with a link to 'What's New in the SAP HANA Platform 2.0 (Release Notes)' and a language selector for English.

XSA Application Development Tools



© SAP

Exercise

Web IDE

- Start SAP Web IDE for SAP HANA and Customize your code editor (That is how the Web IDE is going to look like in HANA Trial, once SP 11 is activated)
- Navigate to the SAP HANA Web-based Development Workbench

Solution WebIDE

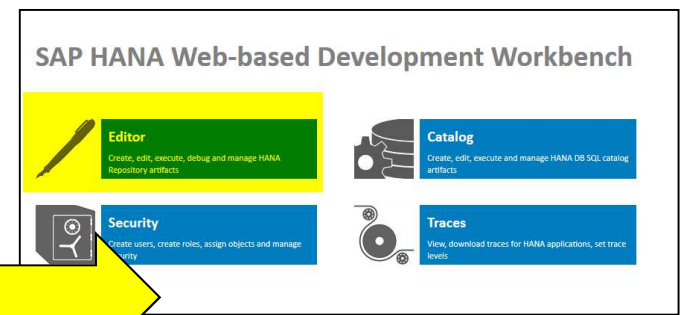
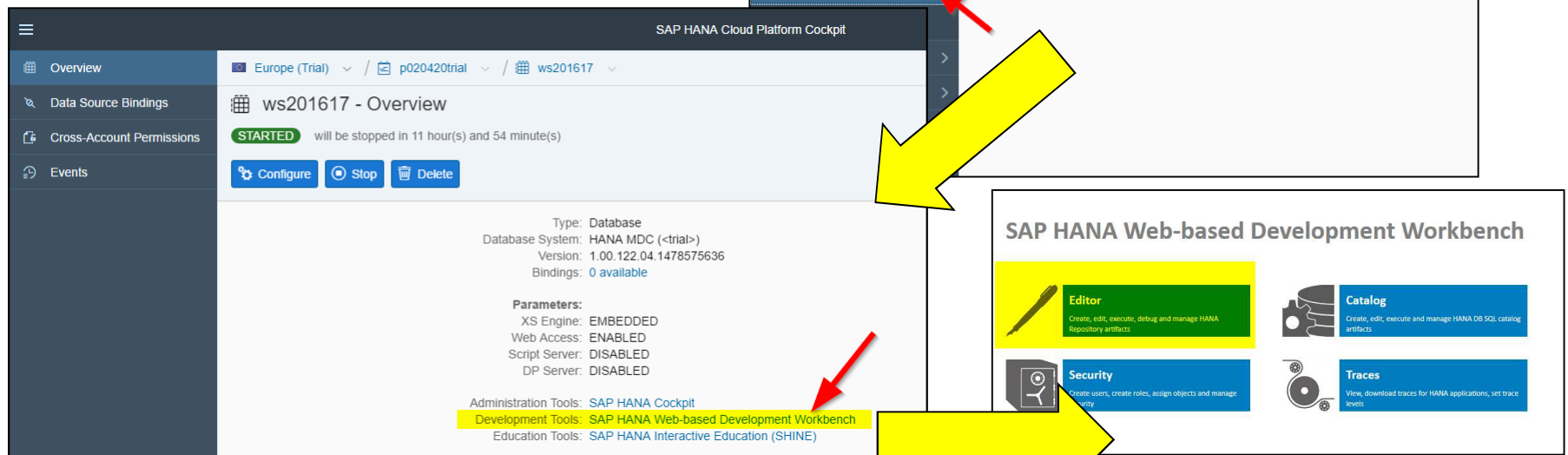
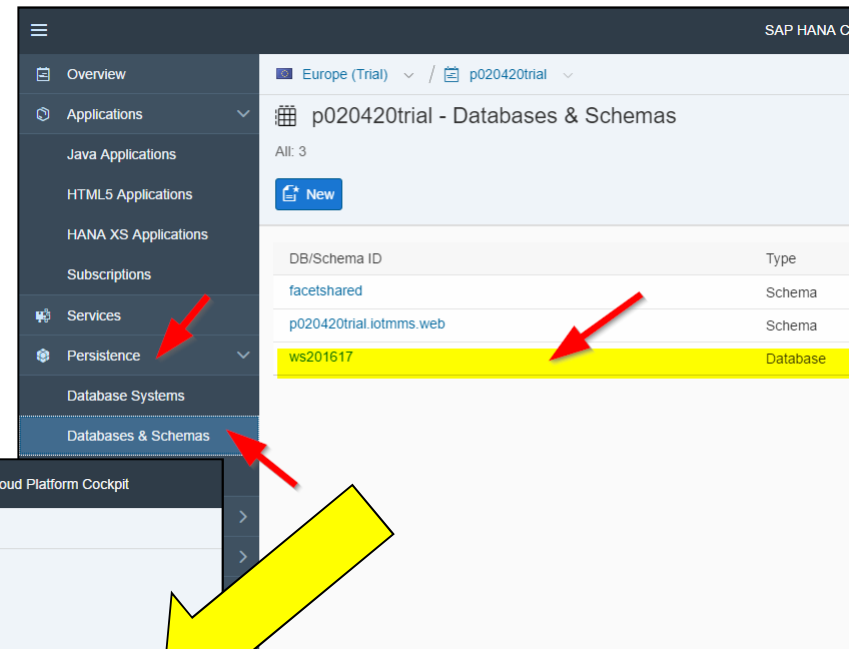
- Navigate to the Web IDE
- Customize your code editor by choosing a theme you like, changing the font size and enabling auto save.

The image shows the SAP HANA Cloud Platform Cockpit interface. On the left, a navigation menu lists various services. The main area displays a grid of service cards. The 'SAP Web IDE' card is highlighted in yellow. A yellow arrow points from this card to a detailed view of the 'SAP Web IDE - Overview' page. This page includes a 'Service Description', 'Documentation' (with an 'Open SAP Web IDE' button), and 'Service Configuration' sections. A second yellow arrow points from the 'Open SAP Web IDE' button to a code editor window showing a JavaScript file.

Solution

Web-based Development Workbench

- Navigate to Persistence -> Databases & Schemas
- Select your database
- On the next view select the Development Tools link
- Voila!



BUILDING UIS WITH SAPUI5

JavaScript

- JS is the basis e.g. for Node.js or SAP HANA XSJS
- It is a scripting language
- It is loosely typed, i.e. determined during runtime

Exercise:

Plain JS

1. Using Notepad, create a file named <matnr>-1.1.htm
2. In the file, copy the following content:

```
<script> var a = 2; var b = 3; var c = a+b; alert("c = "+c); </script>
```
3. Save the file.
4. Open the file using Chrome Developer Tools.
5. Set a breakpoint in the first line after the <script> tag. Execute the script and observe the execution stopping at the breakpoint.
6. Set 3 watch expressions to display the content of variables a, b, and c.
7. Execute the script step-by-step until it ends.

Function

Create an HTML file named <matnr>-1.2.htm containing a JavaScript program based on the following description:

1. Create a JavaScript function named Multiply with 2 input parameter. The function returns the sum of the two parameters.
2. Create 2 numeric variables, a and b, with values 4 and 5.
3. Display the result in an alert window.
4. Execute the multiplication of the variables using the Multiply function.

Object

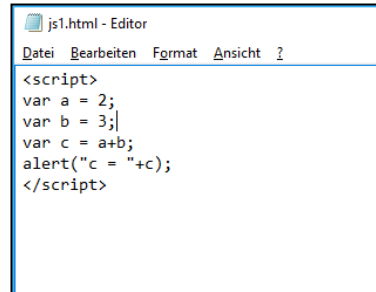
Create an HTML file named <matnr>-1.3.htm containing a JavaScript program based on the following description:

1. Create an object with the following 3 items: • firstName with value "John" • lastName with value "Smith" • a function named FullName, returning name and surname.
2. Execute the function and display the full name in an alert window.

Solution

Plain JS

```
<script>
  var a = 2;
  var b = 3;
  var c = a+b;
  alert("c = "+c);
</script>
```



js1.html - Editor

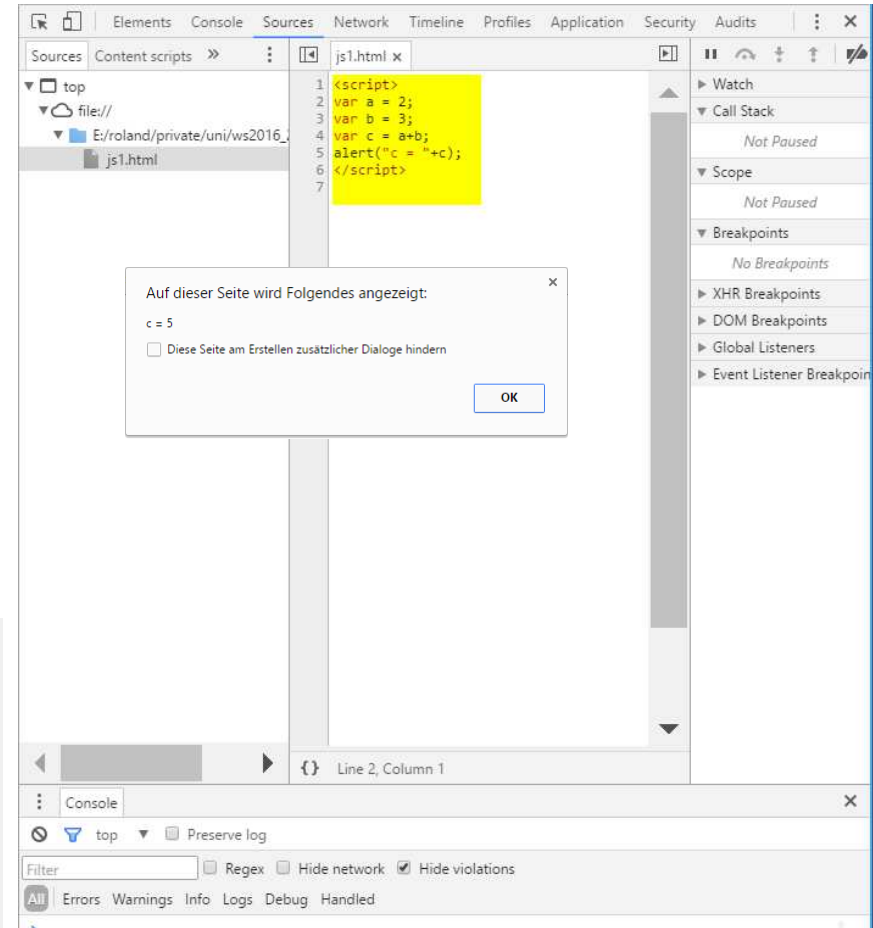
```
<script>
var a = 2;
var b = 3;
var c = a+b;
alert("c = "+c);
</script>
```

```
<script>
  function Multiply(x,y) { return x*y; }

  var a = 4, b = 5;
  alert("Result = " + Multiply(a, b));
</script>
```

```
<script>

  obj = {
    firstName: "John",
    lastName: "Smith",
    FullName: function(){
      return this.firstName + " " + this.lastName
    }
  };
  alert( obj.FullName() );
</script>
```



UI5

WebIDE

Create a SAPUI5 Project

Attention!

In the next steps we are using the **WEB IDE not** in the context of the **XS Applications** but in the context of cloud development of SAP UI5 development.

Lateron we are switching to XS applications development and we are going to use the **Web based development tools** for XSC Applications.

!!!2 different things!!!

UI5 WebIDE Create a SAPUI5 Project

The image shows the SAPUI5 WebIDE interface. On the left, the 'File' menu is open, displaying options such as 'New', 'Close All', 'Save', 'Import', 'Export', 'Archive', and 'Git'. The 'New' sub-menu is expanded, showing 'File', 'Folder', 'Project from Template', 'Project from Sample Application', 'Quick Start with Layout Editor', 'Extension Project', 'Extension', 'OPA Page', 'OPA Journey', 'QUnit Test', 'SAPUI5 View', 'Annotation File', 'OData Service', 'HTML5 Application', and 'HTML5 Application'. A yellow arrow points from the 'Project from Template' option to the 'New SAPUI5 Application' dialog box on the right.

The 'New SAPUI5 Application' dialog box is titled 'New SAPUI5 Application' and has a 'Template Selection' tab. It contains a search bar, a 'Category' dropdown set to 'SAP Fiori Application', a 'Sort By' dropdown set to 'Sort by recently used', and a 'SAPUI5 Version' dropdown set to 'SAPUI5 1.38'. Below these are four application templates: 'CRUD Master-Detail Application', 'SAP Fiori Full Screen Application', 'SAP Fiori Master-Detail Application', and 'SAPUI5 Application'. The 'SAPUI5 Application' template is highlighted. At the bottom, there are 'Previous' and 'Next' buttons.

UI5

WebIDE

Create a SAPUI5 Project

Template Selection **Basic Information** Template Customization Confirmation

New SAPUI5 Application

Basic Information

Project Name*

helloworld

App Descriptor Data

Namespace

at.facet.s8820038.helloworld

Enable as Mobile App

- Create fresh and engaging user experiences with mobile native capabilities.
Package, Build and Manage your SAP Fiori Apps as Hybrid/Cordova Apps when using Web IDE with Fiori Mobile service.

Previous

Next

Finish

UI5

WebIDE

Create a SAPUI5 Project

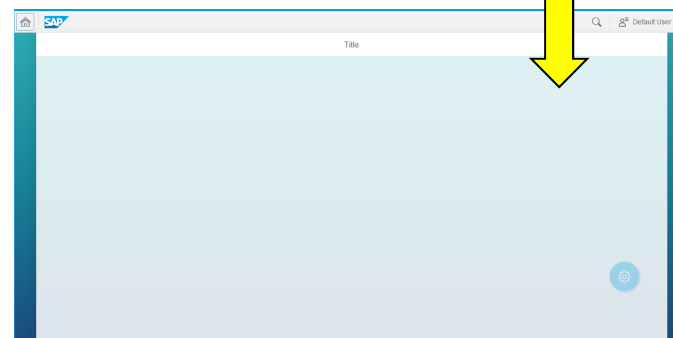
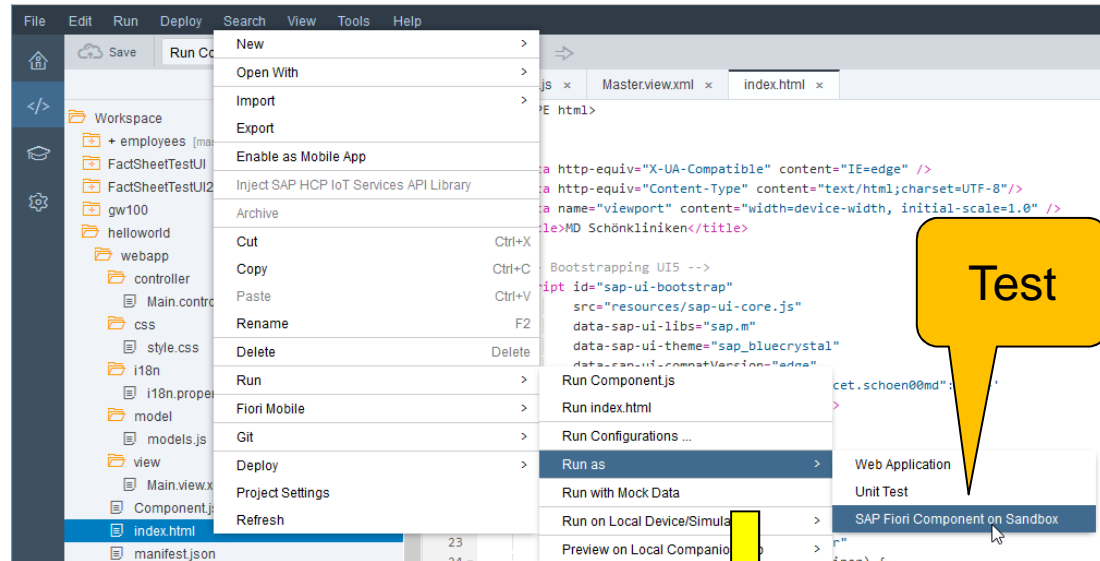
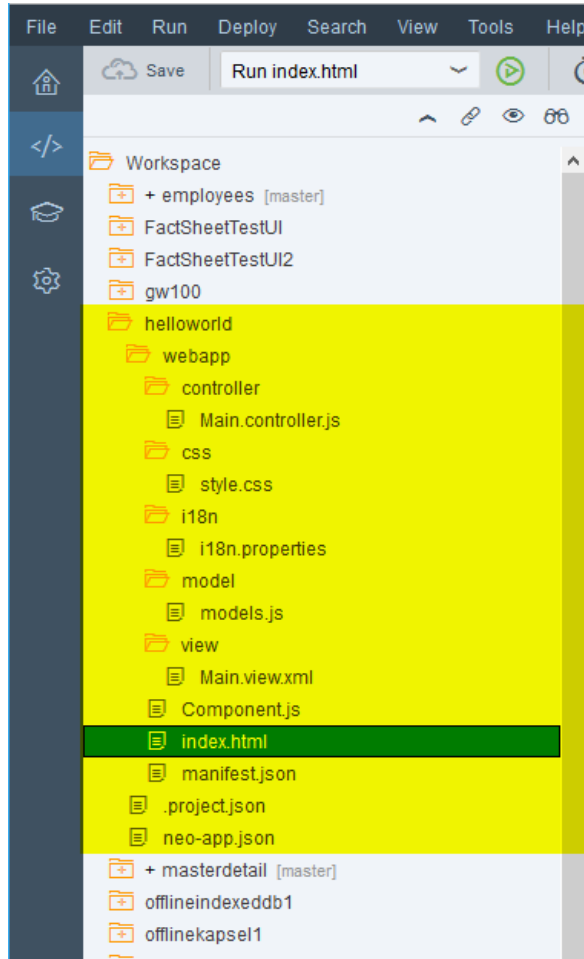
The screenshot shows the 'New SAPUI5 Application' wizard in the 'Template Customization' step. The wizard has four steps: 'Template Selection', 'Basic Information', 'Template Customization', and 'Confirmation'. The 'Template Customization' step is active. The title is 'New SAPUI5 Application' and the subtitle is 'Template Customization'. Under 'Initial View Details', there are two fields: 'View Type*' with a dropdown menu showing 'XML' and 'View Name' with a text input field containing 'Main'. At the bottom, there are three buttons: 'Previous', 'Next', and 'Finish'. A mouse cursor is pointing at the 'Next' button.

- Click Next and then Finish in the next screen

UI5

WebIDE

Create/Test a SAPUI5 Project



UI5

WebIDE

Create Label and Textfield

Entities (216)

lab

Chart

VizFrame 27

Display

Label 3

Process Flow 7

Map

Analytic Map 7

Geo Map 8

Entity

sap.m.Label

Application Component: CA-UI5-CTR Inherits from: Control Category: Display

Available Since: 1.10 Content Density: Compact, Cozy

ABOUT 3 SAMPLES 6 PROPERTIES 1 ASSOCIATIONS 1 METHODS

Name	Description
Label	Labels are helpful when you need to describe some other UI control.
Form - Single 3:5:4	With a form you can easily layout (a) lists of properties (b) input fields. The form automatically adapts to the screen size. Even though the form is not part of sap.m it can and shall be used for building mobile user interfaces. This is an example of a single-column 3:5:4 layout (label:input:emptyspace) using the Form control
Simple Form - Single 3:5:4	With a form you can easily layout (a) lists of properties (b) input fields. The form automatically adapts to the screen size. Even though the form is not part of sap.m it can and shall be used for building mobile user interfaces. This is an example of a single-column 3:5:4 layout (label:input:emptyspace) using the SimpleForm control

UI5

WebIDE

Create Label and Textfield

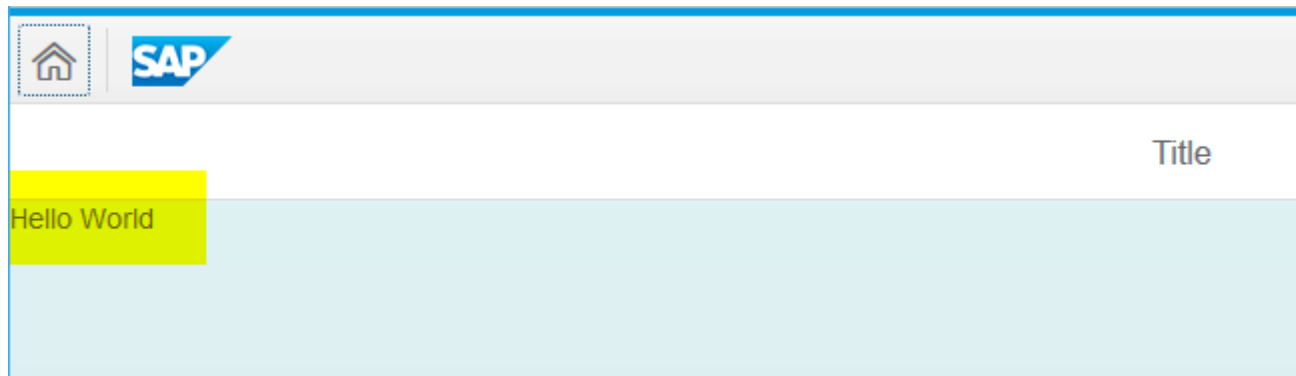
The screenshot illustrates the SAP WebIDE interface for creating a Label widget. The left pane, titled 'Entities (216)', lists various UI5 widgets: lab, Chart, VizFrame (27), and Display. The right pane, titled 'Sample: Label', shows a list of labels: Label a, Label b, Label c, and Label d. A yellow arrow points from the 'lab' entity in the left pane to the 'Code: Label' pane, which displays the XML code for a LabelGroup view.

```
01 <mvc:View>
02   xmlns:l="sap.ui.layout"
03   xmlns:mvc="sap.ui.core.mvc"
04   xmlns="sap.m">
05   <l:VerticalLayout
06     class="sapUiContentPadding"
07     width="100%">
08     <l:content>
09       <Label text="Label a" class="sapUiSmallMarginBottom" />
10       <Label text="Label b" class="sapUiSmallMarginBottom" />
11       <Label text="Label c" class="sapUiSmallMarginBottom" />
12       <Label text="Label d" class="sapUiSmallMarginBottom" design="Bold" />
13       <Label text="Label e" class="sapUiSmallMarginBottom" />
14     </l:content>
15   </l:VerticalLayout>
16 </mvc:View>
```

UI5

WebIDE

Create Label and Textfield



Exercise

WebIDE

SAPUI5

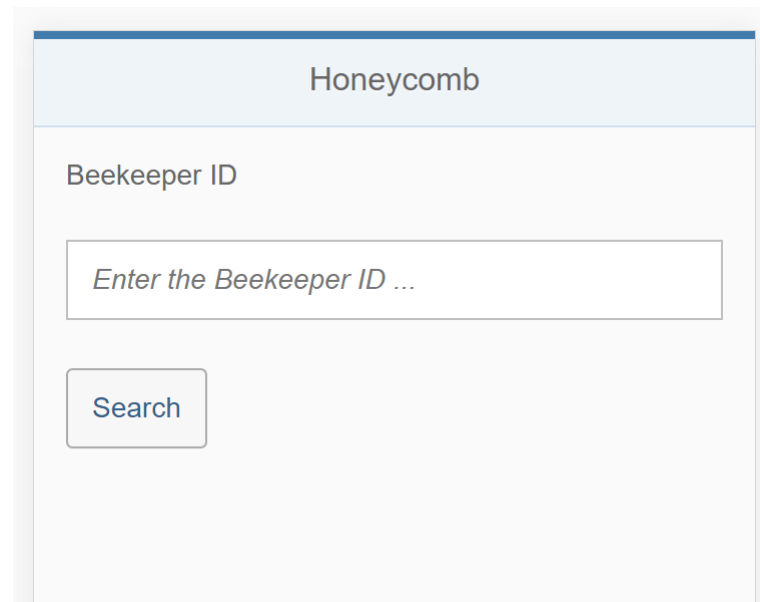
- Create a HelloWorld SAPUI5 Application with the WebIDE
- Insert a Label
- Insert an Inputfield
- Insert a Button

Solution

WebIDE

SAPUI5

- Just follow the previous steps



The screenshot displays a web interface titled "Honeycomb". Below the title, there is a label "Beekeeper ID" followed by a text input field containing the placeholder text "Enter the Beekeeper ID ...". Below the input field is a "Search" button.

Solution

WebIDE

SAUI5 View

```
<mvc:View controllerName="at.facet.hwHelloWorld.controller.Main"
  xmlns:html="http://www.w3.org/1999/xhtml" xmlns:mvc="sap.ui.core.mvc"
  xmlns:l="sap.ui.layout" displayBlock="true" xmlns="sap.m">
  <App>
    <pages>
      <Page title="{i18n>title}">
        <content>
          <l:VerticalLayout class="sapUiContentPadding"
            width="100%">
            <Label text="Beekeeper ID"
              class="sapUiSmallMarginBottom"/>
            <Input type="Text"
              class="sapUiSmallMarginBottom"
              placeholder="{i18n>placeholderBeekeeperID}"/>
            <Button text="Search"
              press="onPress"></Button>
          </l:VerticalLayout>
        </content>
      </Page>
    </pages>
  </App>
</mvc:View>
```

Solution

WebIDE

SAUI5 Controller

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function(Controller) {
    "use strict";

    return Controller.extend("at.facet.hwHelloWorld.controller.Main", {

        onPress: function(oEvent){
            debugger;
            var value =
this.getView().byId("beekeeperID").getValue();
            alert("Hello I am here" + value);
        }

    });
});
```


SAPUI5 Structure

`index.html` is the default starting point of the application. It contains 3 main areas:

- Bootstrap - Contains configuration including the libraries to be loaded, resource root location, and theme.
- Application – Initial construction of application including `placeAt` method to attach application into primary UI Area
- UI-Area – Contains primary UI anchor.

Best practice keeps minimal coding in `index.html`

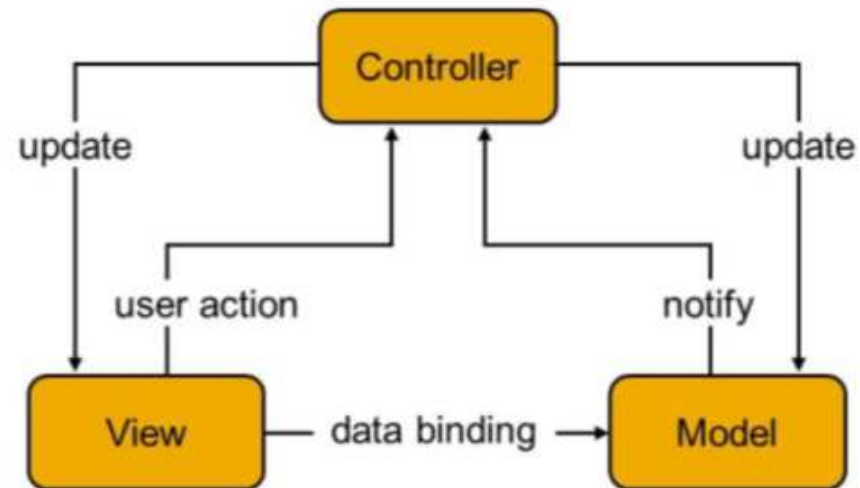


© SAP

SAPUI5 MVC

The Model View Controller (MVC) concept is used in SAPUI5 to separate the representation of information from the user interaction.

- Model – Manages application data
- View – Defines and renders the UI
- Controller – reacts to view events and user interaction by modifying the view and model



The separation has the following advantages:

- It provides better readability, maintainability, and extensibility.
- It allows you to change the view without touching the underlying business logic and to define several views of the same data.

Views and controllers usually form a 1:1 relationship (but standalone controllers and views are possible).

© SAP

SAPUI5 Model

A **model** holds the data, provides methods to fetch the data, and to set and update data. The model types are as follows:

- **JSON** – Client-side model. Intended for small datasets and can be used to bind controls to JavaScript object data.
- **XML** – Client-side model. Intended for small datasets. Does not support server based paging or loading of deltas.
- **Resource** – Handles resource bundles. Mainly for text translation.
- **oData** – Server-side model. Binds controls to oData services. Client only knows currently fetched rows and fields. Sorting and filtering only on the server.

Multiple models and types are possible within a view, and the models may be set at different levels such as at the application level or for an individual control.

The data binding is not dependent on the type of model used.

SAPUI5 View

The view is responsible for defining and rendering the UI. SAPUI5 supports the following predefined view types:

- **XML** – The user interface is defined in an XML file or string. (Supports a mix of XML and plain HTML.) Preferred approach for Fiori apps.
- **JSON** – The user interface is defined in a file or string in JSON format.
- **JS** – The user interface is constructed with JavaScript.
- **HTML** – The user interface is constructed with HTML.

Custom view types can be defined by extending base class `sap.ui.core.mvc.View`

```
<core:View xmlns:core="sap.ui.core"
  xmlns:mvc="sap.ui.core.mvc"
  xmlns="sap.m" controllerName="sap.sapx05.view.View1"
  xmlns:html="http://www.w3.org/1999/xhtml">
  <Page title="Exercise 5 - XML View">
    <content>
      <Label id="lblFirstName" text="First Name:" />
      <Input id="inpFirstName" />
      <Label id="lblLastName" text="Last Name:" />
      <Input id="inpLastName" />
      <Button id="btnSubmit" text="Submit" press="onBtnClick" />
    </content>
  </Page>
</core:View>
```

```
{
  "type": "sap.ui.core.mvc.JSONView",
  "controllerName": "sap.sapx05.view.View1",
  "content": [
    {
      "type": "sap.m.Page",
      "title": "Exercise 5 - JSON View",
      "content": [
        {
          "type": "sap.m.Label",
          "id": "lblFirstName",
          "text": "First Name:"
        },
        {
          "type": "sap.m.Input",
          "id": "inpFirstName",
          "text": ""
        },
        {
          "type": "sap.m.Label",
          "id": "lblLastName",
          "text": "Last Name:"
        },
        {
          "type": "sap.m.Input",
          "id": "inpLastName",
          "text": ""
        },
        {
          "type": "sap.m.Button",
          "id": "btnSubmit",
          "text": "Submit"
        }
      ]
    }
  ]
}
```

© SAP

SAPUI5 View Conventions

- View names are capitalized
- All views are stored in the view folder
- Names of XML views always end with *.view.xml
- The default XML namespace is sap.m
- Other XML namespaces use the last part of the SAP namespace as alias (for example, mvc for sap.ui.core.mvc)

SAPUI5 Controller

SAPUI5 uses the controller to separate the view logic from the model logic. It has the following lifecycle hooks:

- **onInit()** – Called when a view is instantiated and its controls (if available) have already been created; used to modify the view before it is displayed to bind event handlers and do other one-time initialization.
- **onExit()** – Called when the view is destroyed; used to free resources and finalize activities.
- **onAfterRendering()** – Called when the view has been rendered and, therefore, its HTML is part of the document; used to do post-rendering manipulations of the HTML. SAPUI5 controls get this hook after being rendered.
- **onBeforeRendering()** – Invoked before the controller view is re-rendered and **not** before the first rendering; use `onInit()` for invoking the hook before the first rendering.

A controller can also define additional methods that serve as event handlers or additional functionality offered by the controller.

SAPUI5 Controller Conventions

- Controller names are capitalized
- Controllers carry the same name as the related view (if there is a 1:1 relationship)
- Event handlers are prefixed with **on**
- Controller names always end with ***.controller.js**

SAPUI5

Common UI Controls

- Mobile:

<https://sapui5.hana.ondemand.com/explored.html>

- General (old):

<https://sapui5.hana.ondemand.com/#content/Controls/index.html>

SAPUI5

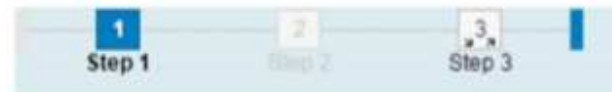
Control Samples

SAPUI5 provides different types of UI-controls:

- Simple controls
- Value holders
- Layouts
- Complex controls
- Dialogs
- UX3 controls



Emphasized Accept Reject



▼ Section 1

CheckBox1 0Label 1 0

CheckBox1 1Label 1 1

CheckBox1 2Label 1 2

> Section 2

> Section 3

> Section 4

Table Example

Button in the Toolbar

Button in the Extension Area

Last Name	First Name	Chec...	Web Site	Image	Gender	Rating
Ander	Conny	<input type="checkbox"/>	www.sap.com		male	★★★★★
Bar	Sandy	<input checked="" type="checkbox"/>	www.sap.com		male	★★★★☆
Burr	Tim	<input checked="" type="checkbox"/>	www.sap.com		male	★★★★☆
Case	Justin	<input type="checkbox"/>	www.sap.com		male	★★★★☆
Corrie	Al	<input checked="" type="checkbox"/>	www.sap.com		male	★★★★★
Dewitt	Kensia	<input type="checkbox"/>	www.sap.com		female	★★★★☆
Early	Brighton	<input checked="" type="checkbox"/>	www.sap.com		male	★★★★☆

© SAP

SAPUI5

Control in XML View

```
<Control propertyX="value" ...  
    propertyY="{ subpropX:'value' subpropY:'value' ...}"  
    propertyZ="{ subpropX:{ sspX:'val' sspY:'val' ...} ...}"  
>  
<aggregation1></aggregation1>  
<aggregation2></aggregation2>  
...  
</Control>
```

- **Control** – Name of the control.
- **propertyX Y Z** – Property for the specific control. See Documentation. Could also be an event.
- **subpropX Y sspX Y** – Sub-properties in JSON string format. Can be nested.
- **aggregation1 2** – Additional groups of data, such as like nested controls, layout data, and so on.

SAPUI5

sap.m.Input

Selected Properties:

- **type** – HTML type for input tag. Supported by browsers natively (Text, Number, Email, Phone, ...)
- **placeholder** – Text to display in field before data entry.
- **description** – Description of field data to be displayed after field.
- **fieldWidth** – Width when using description

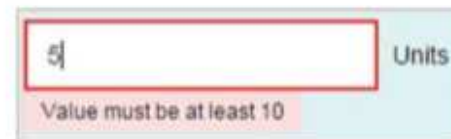
Events:

- **liveChange** – triggered as data is entered. (Assumed in the controller.)



Enter Number... Units

```
<Input type="Number"
  placeholder="Enter Number..."
  fieldWidth="25%"
  description="Units"
  liveChange="onChange">
</Input>
```



5 Units
Value must be at least 10

```
onChange      : function(oEvent) {
  var input = oEvent.getSource();
  if (input.getValue() < 10) {
    input.setValueState(sap.ui.core.ValueState.Error);
    input.setValueStateText("Value must be at least 10");
  } else {
    input.setValueState(sap.ui.core.ValueState.None);
  }
}
```

© SAP

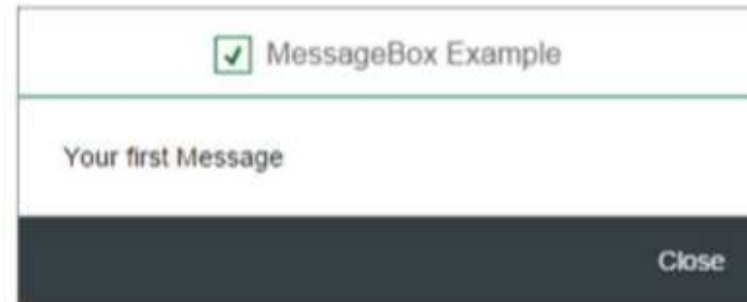
SAPUI5

sap.m.MessageBox

Specialization of sap.m.Dialog with configured texts and buttons.

Additional Properties:

- **title** – Displayed on top of the pop-up.
- **icon** – Icon for message type. Supported values: Error, Information, Success, None, Question, Success, Warning.
- **actions** – Button to be added to message. Supported Values: Abort, Cancel, Close, Delete, Ignore, No, Ok, Retry, Yes.
- **onClose** – Method triggered when user presses a button.



```
var message = "Your first Message";
sap.m.MessageBox.show(message, {
  icon: sap.m.MessageBox.Icon.SUCCESS,
  title: "MessageBox Example",
  actions: sap.m.MessageBox.Action.CLOSE,
  verticalScrolling: false,
  horizontalScrolling: false
});
```

SAPUI5

sap.ui.table.Table

The Table control provides a set of sophisticated and comfort functions for table design.

Additional Properties/Aggregations:

- **rows** – Includes binding information for the data.
- **columns** – Defines each column and can contain a data template based on the bound data.
- **sortProperty** – At the column level supports user selected sorting based on the configured data.
- Many other options. Also see **sap.m.Table** for selected functionality optimized for mobile devices.

ID	First Name	Last Name
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock

```
<t:Table id="oDataTable" rows="{path: '/Employees'}">
  <t:columns>
    <t:Column sortProperty="EmployeeID">
      <m:Label text="ID"/>
      <t:template>
        <m:Text text="{EmployeeID}"/>
      </t:template>
    </t:Column>
    <t:Column sortProperty="FirstName">
      <m:Label text="First Name"/>
      <t:template>
        <m:Text text="{FirstName}"/>
      </t:template>
    </t:Column>
    <t:Column sortProperty="LastName">
      <m:Label text="Last Name"/>
      <t:template>
        <m:Text text="{LastName}"/>
      </t:template>
    </t:Column>
  </t:columns>
</t:Table>
```

© SAP

SAPUI5

Layout

- **sap.m.FlexBox** – Flexible box layout
 - **sap.ui.layout.FitFlex** – A fixed part for multiple controls and an adaptable part for one control
 - **sap.ui.layout.Grid** – 12 column flow layout
 - **sap.ui.layout.HorizontalLayout**
 - **sap.ui.layout.ResponsiveFlowLayout** – Controls blown up to fit one line
 - **sap.ui.layout.VerticalLayout**
- **sap.ui.commons.layout** – layouts focused on desktop devices:
 - **AbsoluteLayout**
 - **BorderLayout**
 - **MatrixLayout**
 - **HorizontalLayout** – deprecated, use **sap.ui.layout** version.
 - **ResponsiveFlowLayout** – deprecated, use **sap.ui.layout** version
 - **VerticalLayout** – deprecated, use **sap.ui.layout** version

© SAP

SAPUI5

Example sap.m.FlexBox

Selected Properties:

- **alignItems** – Options include: Baseline, Center, End, Inherit, Start, and Stretch.
- **justifyContent** – Options include: Center, End, Inherit, SpaceAround, SpaceBetween, and Start.
- **height / width** – CSS sizes, such as "1px" or "2em" or "50%".

Aggregations:

- **items** – Contains any controls.

```
<FlexBox alignItems="Center" justifyContent="SpaceAround">  
  <items>  
    <Label id="lblLastName" text="Last Name:" />  
    <Input id="inpLastName" />  
  </items>  
</FlexBox>
```



In this example, the controls are centered on the line and the space is evenly distributed between the controls.

SAPUI5

i18n

- Create an `i18n.properties.[de|en|...]` for translateable texts

```
// set i18n model on view
```

```
    var i18nModel = new ResourceModel({  
        bundleName: 8820038.i18n.i18n  
    });
```

```
    this.getView().setModel(i18nModel, i18n);
```


i18n

Conventions

- The resource model for internationalization is called the i18n model.
- The default filename is i18n.properties.
- Resource bundle keys are written in (lower) camelCase.
- Resource bundle values can contain parameters like {0}, {1}, {2}, ...
- Never concatenate strings that are translated, always use placeholders.
- Use Unicode escape sequences for special characters.

Exercise

WebIDE

SAPUI5

- Internationalize the App
- React to the Button press in the controller
- Alert the value of the Inputfield
- Send a MessageBox
- Insert a sap.m.Table
- Insert a Json model
- Set the Json model data in the table

Solution WebIDE SAPUI5

Honeycomb

Beekeeper ID

Search

Measures

BeekeeperID	Date	Weight
24 Imker Lois	01.02.2017	13 g

Solution

WebIDE

SAUI5 View - Table

```
<Table id="idMeasureTable" inset="false" items="{ path: '/MeasureCollection', sorter: { path: 'beekeeperID' } }">
```

```

    <Title text="Measures" level="H2"/>
    <Text text="BeekeeperID"/>
    <Text text="Date"/>
    hAlign="Right">
    <Text text="Weight"/>
    <cells>
        <ObjectIdentifier title="{beekeeper_id}" text="{beekeeper_name}"/>
        <!-- <Text text="{measure_date}"/> -->
        <!-- <Text text="{path: 'measure_date',
            type : 'sap.ui.model.type.Date',
            formatOptions: {pattern: 'dd/MM/yyyy'}
        }"/> -->
        <Text text="{path: 'measure_date', formatter: '.formatDate'}"/>
        <ObjectNumber number="{measure_weight}" unit="{measure_unit}"/>
    </cells>
</ColumnListItem>
</Table>
</items>
</Table>
```

Solution

WebIDE

SAUI5 Controller

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/m/MessageBox"
], function(Controller, MessageBox) {
    "use strict";
    return Controller.extend("at.facet.hwHelloWorld.controller.Main", {
        // Init the object
        onInit: function() {
            // set explored app's demo model on this sample
            var oModel = new sap.ui.model.json.JSONModel("model/measures.json");
            this.getView().setModel(oModel);
        },
        // React on the button
        onPress: function(oEvent) {
            debugger;
            var value = this.getView().byId("beekeeperID").getValue();
            //alert("Hello I am here " + value);
            //sap.m.MessageBox.show("Hello I am here " + value,
            // Falls sap.m.MessageBox in function Parameters definiert
            MessageBox.show("Hello I am here " + value, {
                icon: sap.m.MessageBox.Icon.SUCCESS,
                title: "Hello Beekeeper",
                actions: sap.m.MessageBox.Action.CLOSE,
                verticalScrolling: false,
                horizontalScrolling: false
            });
        },
        formatDate: function(date) {
            // Deklaration laden
            jQuery.sap.require("sap.ui.core.format.DateFormat");
            // Format definieren und Objekt instanzieren
            var oDateFormat = sap.ui.core.format.DateFormat.getDateInstance({
                pattern: "dd.MM.YYYY"
            });
            // Formatierungsergebnis
            return oDateFormat.format(new Date(date));
        }
    });
});
```

Solution

WebIDE

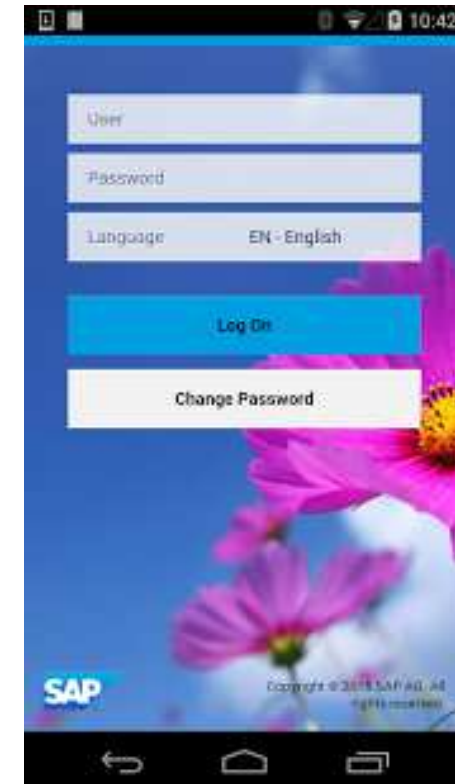
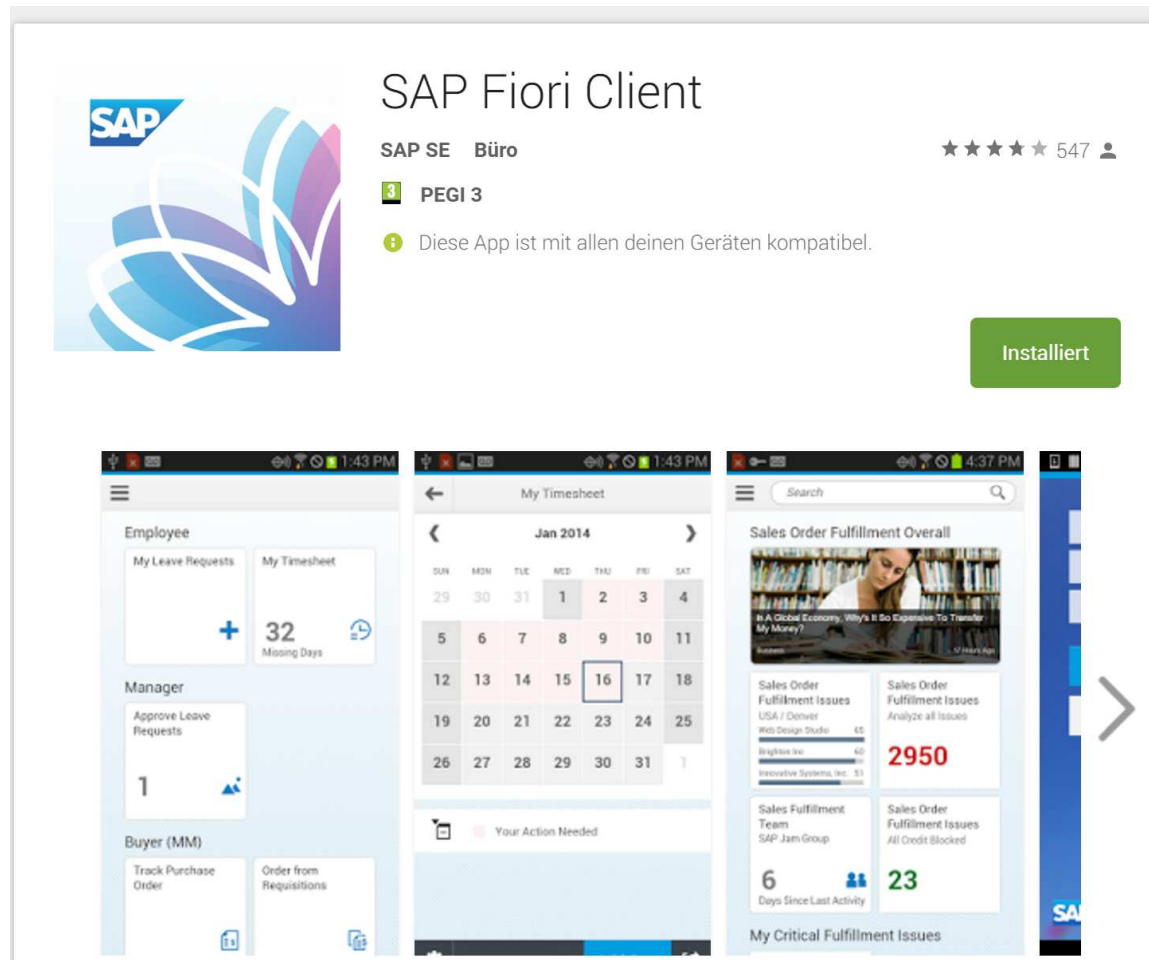
SAUI5 model/measures.json

```
{
  "MeasureCollection": [{
    "beekeeper_id": "24",
    "beekeeper_name": "Imker Lois",
    "measure_date": "2017-02-01",
    "measure_weight": "13",
    "measure_unit": "g"
  }
]
```

SAPUI5

Check the App in your Mobile

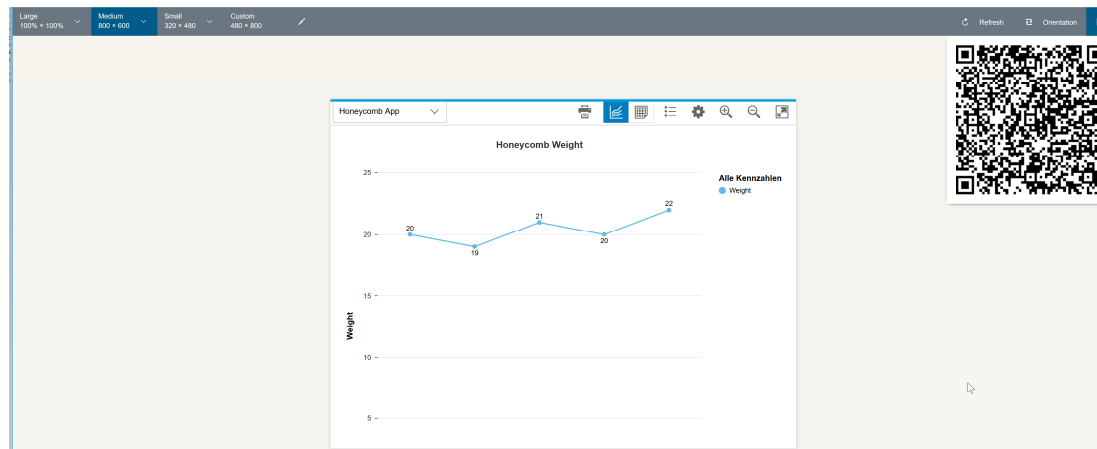
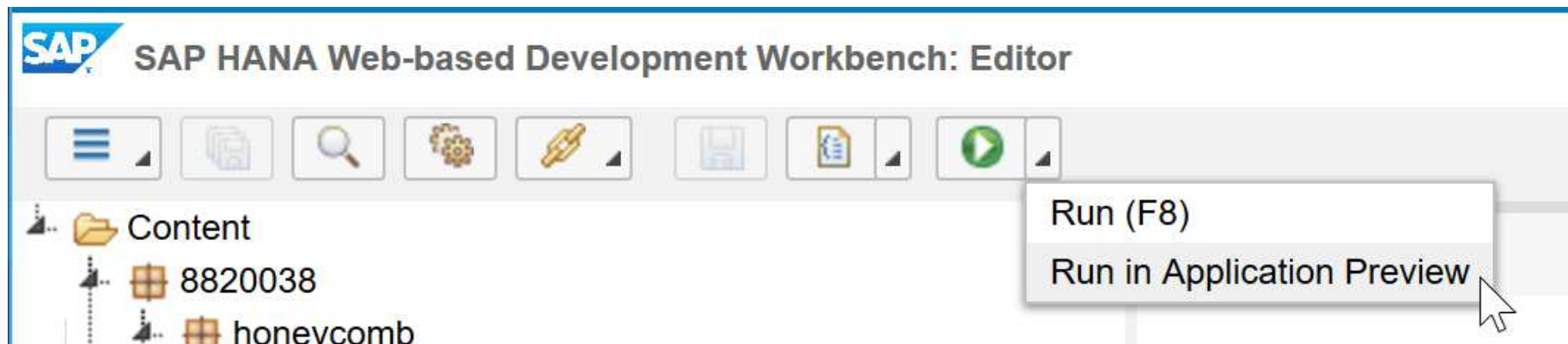
Install the SAP Fiori Client, e.g. on Android



SAPUI5

Check the App in your Mobile XSC Apps

Precondition: Installed SAP Fiori Client on your smartphone



SAPUI5

Check the App in your Mobile WebIDE

The image illustrates the process of running an SAPUI5 application in a mobile environment. It shows three main components:

- Left Panel:** A file explorer showing 'index.html' with a context menu open. The 'Run' option is selected, leading to a sub-menu where 'Run Configurations ...' is highlighted.
- Right Panel:** The 'Run Configurations for HelloWorld' dialog box. The 'General' tab is active, showing 'Run index.html' as the application name and '/webapp/index.html' as the file name. Under 'Preview Mode', the 'With Frame' option is selected. A 'Save and Run' button is visible at the bottom right.
- Bottom Panel:** A mobile device view showing a QR code. A yellow arrow points from the QR code to the text 'Scan with your Fiori Client'.

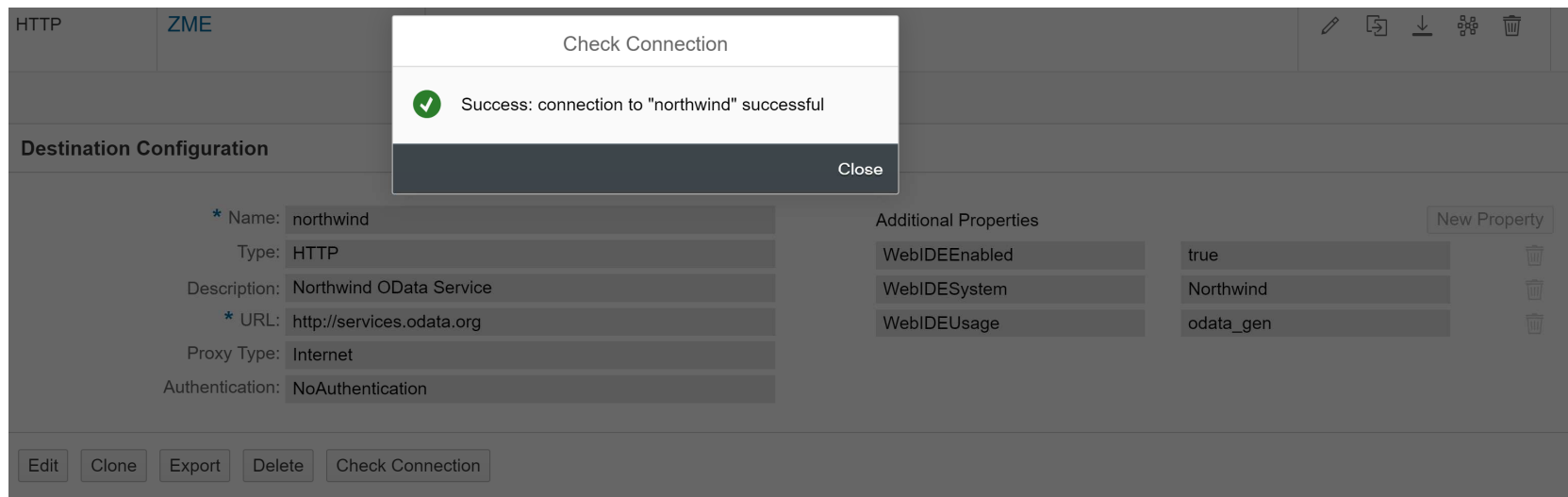
A large yellow arrow points from the 'Run Configurations' dialog to the mobile device view, indicating the flow of the process.

Automatic UI5 Generation

Destinations

Northwind

- Connectivity -> Destination
- New Destination



Automatic UI5 Generation Template

Template Selection Basic Information Data Connection Template Customization Confirmation

New SAP Fiori Master Master-Detail Application

Template Selection

Search Category Sort E

All categories

SAPUI5 Application

SAP Fiori Master Master-Detail Application

SAPUI5 M Kapsel Offlin

Overview Page Application

SAP Web IDE Feature

SAP Web

An SAP Fiori application which displays data from an OData service using the

Template Selection Basic Information Data Connection Template Customization Confirmation

New SAP Fiori Master Master-Detail Application

Data Connection

Service: northwind.svc selected.

Choose a service from one of the sources listed below.

Sources

Service Catalog
Workspace
File System
Service URL

Choose a system to connect to the required service

Northwind OData Service

Service

- northwind.svc
 - > Categories
 - > CustomerDemographics
 - > Customers
 - > Employees
 - > Order_Details
 - > Orders
 - > Products
 - > Regions
 - > Shippers
 - > Suppliers
 - > Territories
 - > Alphabetical_list_of_products

Automatic UI5 Generation Template

Template Selection Basic Information Data Connection **Template Customization** Confirmation

New SAP Fiori Master-Detail Application

Template Customization

Status Attribute: RequiredDate

Attribute: ShipPostalCode

Detail Section

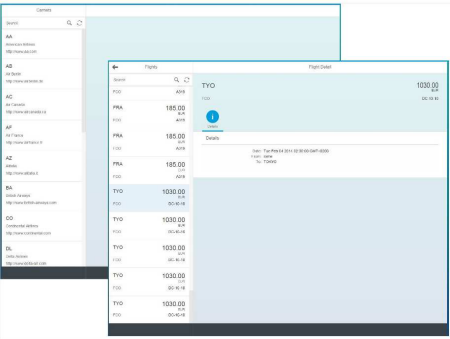
Title: Ship Information

Attribute 1: ShipName

Attribute 2: ShipRegion

Attribute 3: ShipVia

Previous Next Finish



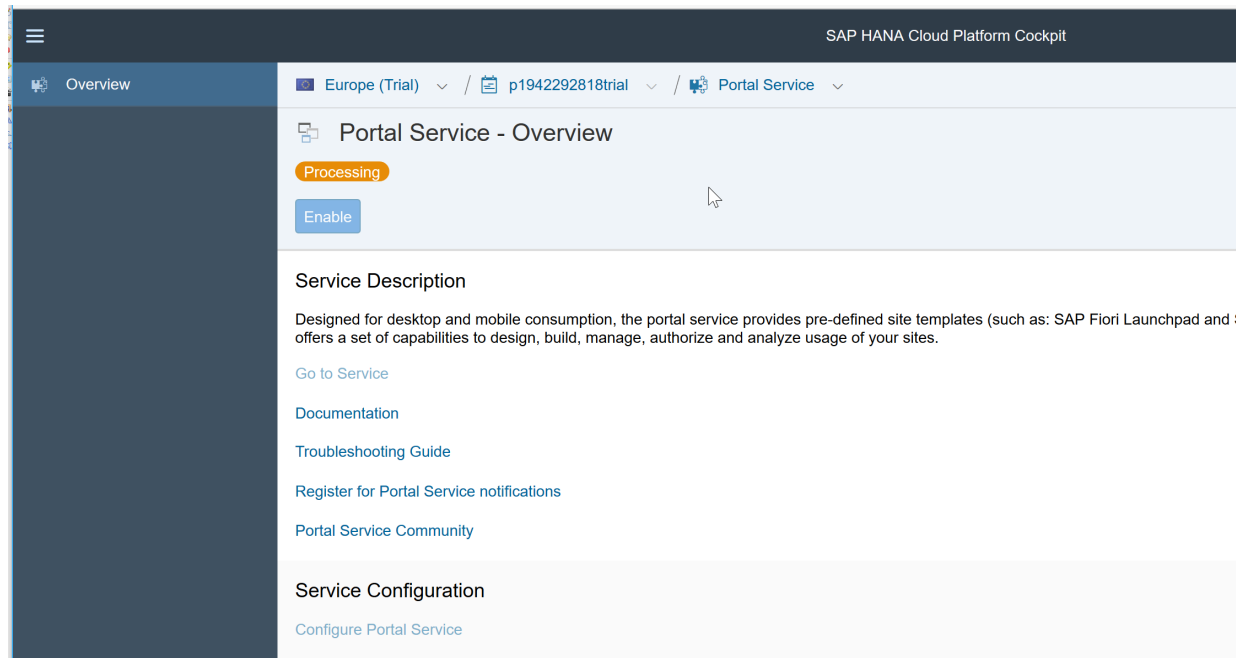
The screenshot shows a SAP Fiori Master-Detail application. The master view displays a list of items with columns for Item Code, Item Name, and Item Price. The detail view shows a form for the selected item, including fields for Item Code, Item Name, and Item Price. The application is running on a mobile device, as indicated by the navigation bar at the bottom.

Launchpad HANA

The screenshot displays the SAP HANA Cloud Platform Cockpit interface. The top navigation bar includes a hamburger menu, the title "SAP HANA Cloud Platform Cockpit", and utility icons for settings, help, notifications, and power. The left sidebar contains a navigation menu with categories: Overview, Applications (Java, HTML5, HANA XS), Subscriptions, Services (selected), Persistence (Database Systems, Databases & Schemas, Service Requests), Connectivity, Security (Trust, Authorizations), OAuth, Repositories, Resource Consumption, Useful Links, and Legal Information. The main content area shows the status of various services for the "Europe (Trial)" region and instance "p1942292818trial".

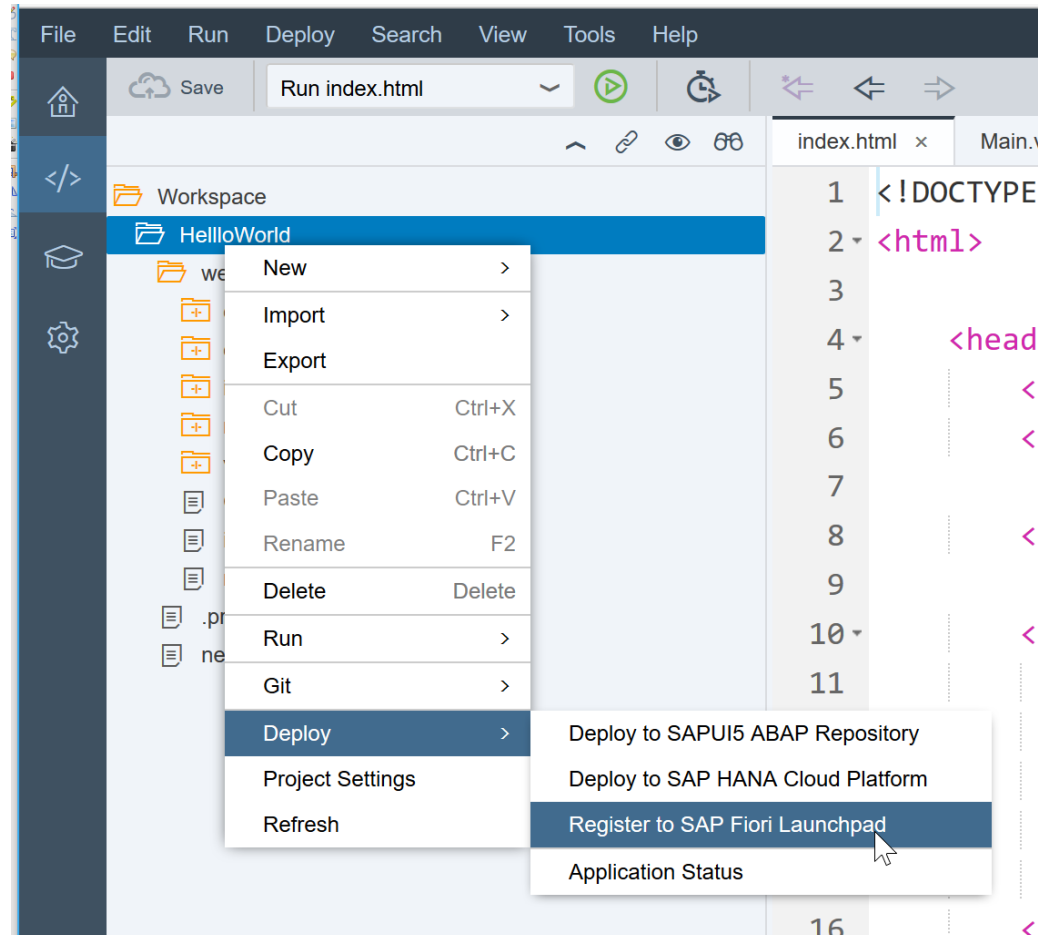
Category	Service Name	Status	Description
Applications	Mobile App Management	Enabled	Manage and secure mobile apps and devices.
	Mobile App Development	Not enabled	Build and run new mobile apps or extend existing on-premise or cloud solutions with a mobile user experience.
	Mobile App Monitoring	Not enabled	Optimize, build, manage and monitor SAP Fiori apps on mobile devices.
Runtime & Containers	HCP, Starter Edition for Clo... BETA	Enabled	Provides access to the HCP Cloud Foundry services to develop, deploy and run applications.
	Authorization Management	Enabled	Platform API for managing roles and their assignments to users.
Security	Identity Provisioning Service	Not enabled	Automates provisioning and de-provisioning of identities and authorizations for cloud applications.
	Keystore Service	Enabled	Secure repository for cryptographic keys and certificates.
	OAuth 2.0 Service	Enabled	Protect applications and APIs with OAuth 2.0.
	UI Theme Designer	Not enabled	Create custom themes to apply your corporate branding to your SAP Fiori launchpad (Portal Service) and SAPUI5 applications.
User Experience	BUILD	Enabled	Create user-friendly interactive prototypes, based on end-user feedback studies, without writing any code.
	Feedback Service BETA	Not enabled	Collect and analyze user feedback for your applications.
	Forms by Adobe	Not enabled	Generates print and interactive forms using Adobe Document Services.
	Portal Service	Not enabled	Easily create free-style and GAP Fiori launchpad style business sites for employees, customers, and partners.

Launchpad HANA



Launchpad HANA

Launchpad HANA



Launchpad HANA

Deploy Application to SAP HANA Cloud Platform

A build will be triggered for the project. The build results will be deployed.

Application Details

Deploy a new application Update an existing application

Account *

Project Name

Application Name *

Version Management

Version * Activate

```
app: new sap.ui.core.ComponentContainer({
```

Launchpad HANA

General Information

Tile Configuration

Assignment

Confirmation

Register to SAP Fiori Launchpad

General Information

Provider Account *

trial (fipportal)

Application Name * [?](#)

at.facet.hwHelloWorld.HelloWorld

Description

Honeycomb

Intent

Semantic Object	Action
HelloWorld	Display

[More information on intent properties.](#)

Previous

Next

Launchpad HANA

General Information

Tile Configuration

Assignment

Confirmation

Register to SAP Fiori Launchpad

Tile Configuration

Type *

Static

Title *

Honeycomb

Subtitle

The only app for beekeepers - sweet!

Icon

sap-icon://bed

Browse

Honeycomb

The only app for
beekeepers - sweet!



Previous

Next

Launchpad HANA

General Information > Tile Configuration > **Assignment** > Confirmation

Register to SAP Fiori Launchpad

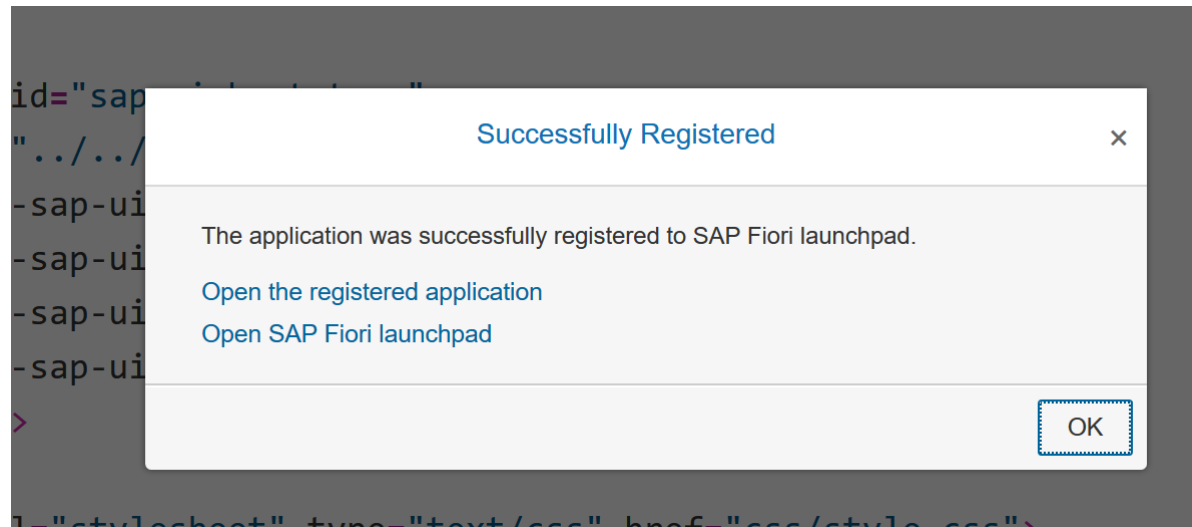
Assignment

Site *

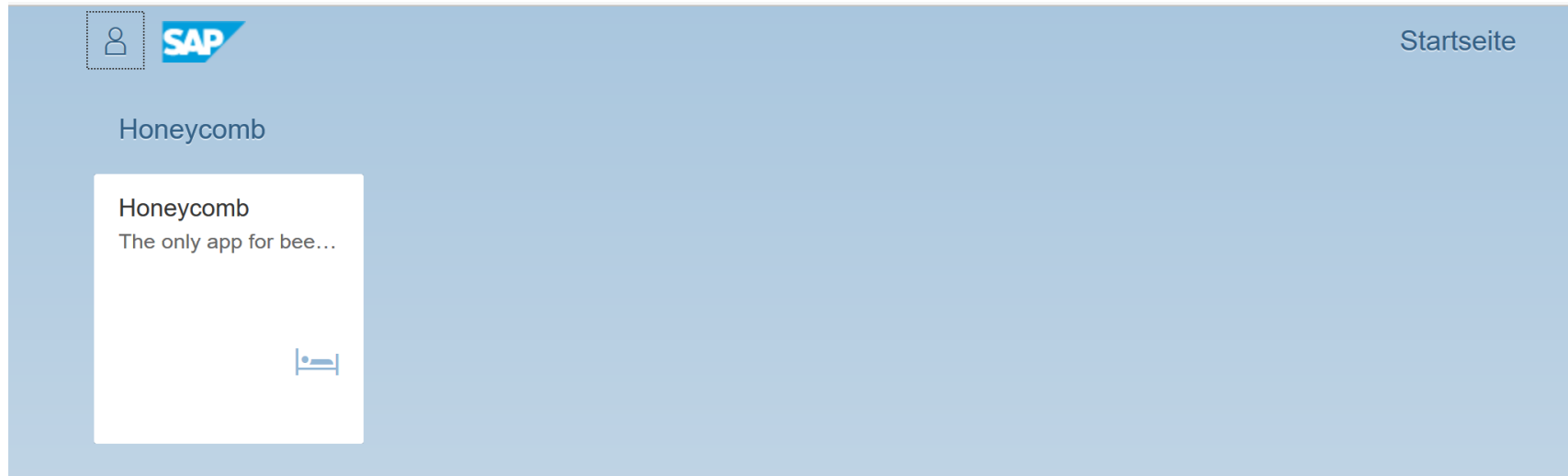
Catalog *

Group *

Launchpad HANA



Launchpad HANA



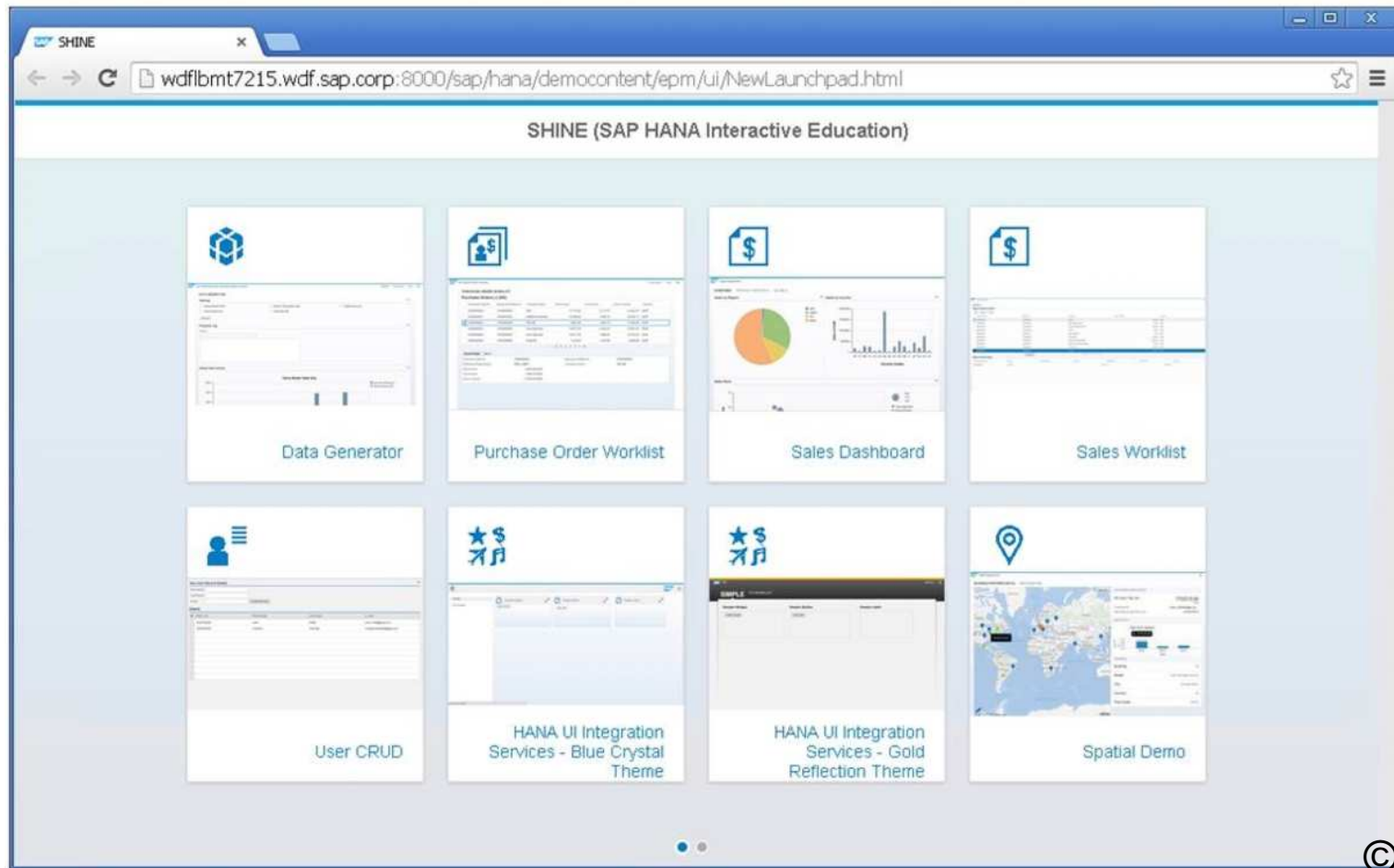
Launchpad HANA

The screenshot displays the Fiori Configuration Cockpit interface. The left sidebar contains navigation options: Startseite, Einstellungen, Inhaltsverwaltung (expanded), Apps, Kataloge, Gruppen, Rollen, Services und Tools, App-Ressourcen, and Verwendungsanalyse. The main content area shows the configuration for 'FLP-8820038' under 'Website-Einstellungen'. The 'Allgemein' section includes: Beschreibung: A site template that uses SAP Fiori design elements. This template is provided by SAP. Status: Offline (indicated by a red dot). ID: d909dc76-9c70-4519-b5aa-2412f56707e7. Typ: SAP Fiori Launchpad. Zugriffsebene: Privat. URL: /sites?siteId=d909dc76-9c70-4519-b5aa-2412f56707e7. Alias: Kein Alias definiert. The 'Systemeinstellungen' section includes: SAPUI5-Version: Innovation. Abmeldeseite: Standardabmeldeseite. SAP-Jam-Integration: Nein. Zugriffsart: Extern. The 'Benutzereinstellungen' section includes: Gruppen ausblenden: Nein. Suchen: Nein. Auswahl des Motivs: Ja. Benutzerpersonalisierung: Ja. Drucktaste Abmelden: Ja. Hintergrundformen: Nein. SAP Easy Access Menü: Nein.

THE PROJECT IN SAP HANA XS CLASSIC AND ADVANCED

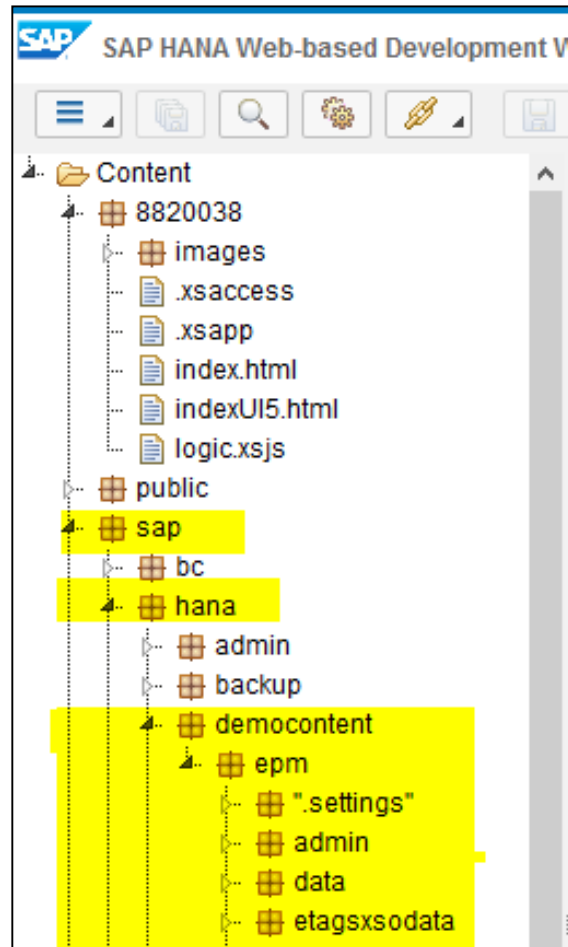
- Give an overview about the SHINE application
- Comparing the XS Classic and XS Advanced development process
- Create a XS Classic Project
- XS Advanced Multi Target Application
- Create a XS Advanced Project

SHINE Demo App



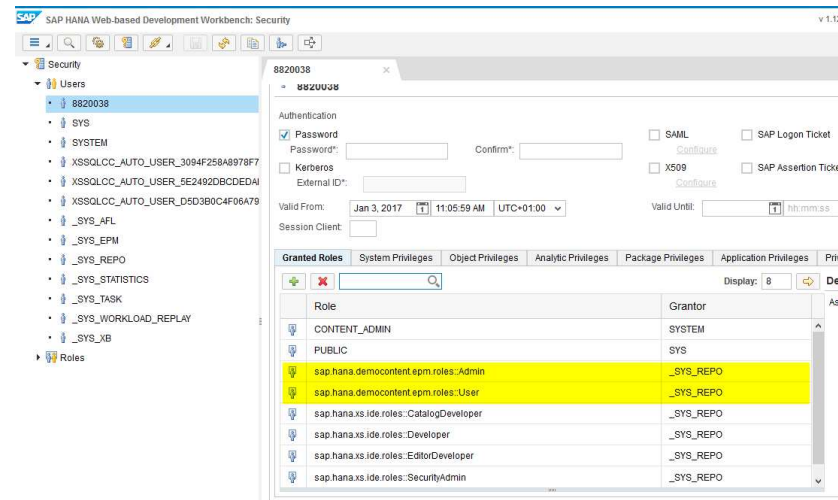
SHINE: SAP HANA

Web-based Dev Workbench: Editor



SHINE: Auth Preparation

- To login to SHINE Launchpad, the user should have either **sap.hana.demococontent.epm.roles:Admin** or **sap.hana.demococontent.epm.roles:User**



Stop after 12 hours



The following SAP HANA database will be stopped on Tuesday, 03-January-2017 at 21:43 UTC

Database: **[REDACTED]**
Account: **[REDACTED]**
Landscape: **hanatrial.ondemand.com**

You can manage your database from the [Cloud Cockpit](#).

If you need to run your database and applications continuously, [upgrade your account](#).

This is an automatic email. Please do not reply.

SAP HANA Cloud Platform Operations

[Contact SAP](#) | [Copyright/Trademark](#) | [Privacy](#) | [Impressum](#)

SAP SE, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany

Pflichtangaben/Mandatory Disclosure Statements: <http://www.sap.com/company/legal/impressum.epx>
Diese E-Mail kann Betriebs- oder Geschäftsgeheimnisse oder sonstige vertrauliche Informationen enthalten. Sollten Sie diese E-Mail irrtümlich erhalten haben, ist Ihnen eine Kenntnisnahme des Inhalts, eine Vervielfältigung oder Weitergabe der E-Mail ausdrücklich untersagt. Bitte benachrichtigen Sie uns und vernichten Sie die empfangene E-Mail. Vielen Dank.

This e-mail may contain trade secrets or privileged, undisclosed, or otherwise confidential information. If you have received this e-mail in error, you are hereby notified that any review, copying, or distribution of it is strictly prohibited. Please inform us immediately and destroy the original transmittal. Thank you for your cooperation.

XSC Project

Initially we create a proper "space" on the server for you to develop.

To do this, we use user `<MATNR>` (with security administration authorizations).

1. Create a **package** in the repository to store development artifacts (that is, JavaScript sources).
2. Create a **schema** in the catalog to store database objects (that is, database tables).
3. Create a security **Role** with authorization to develop in the package and in the schema
4. Assign the role to your user.
5. Generate and run a test application.

Then we create the project in `SAP HANA Web-based Development Workbench`

6. Create an XS Project in `SAP HANA Web-based Development Workbench`

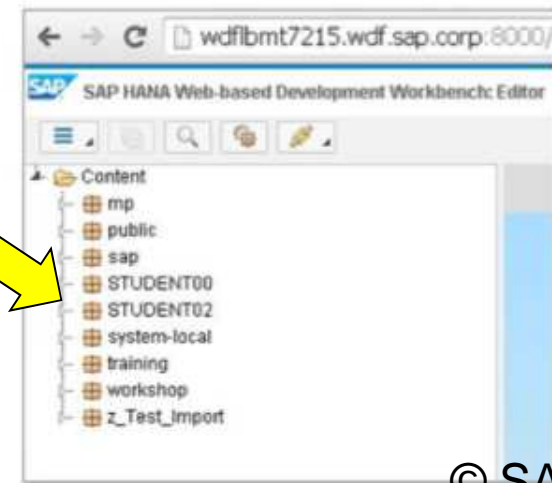
XSC Project

All of your design-time development artifacts will be stored in a package within the SAP HANA Repository.

To create the package, execute following actions:

- Using a web browser, open the **SAP HANA Web-based Development Workbench editor**

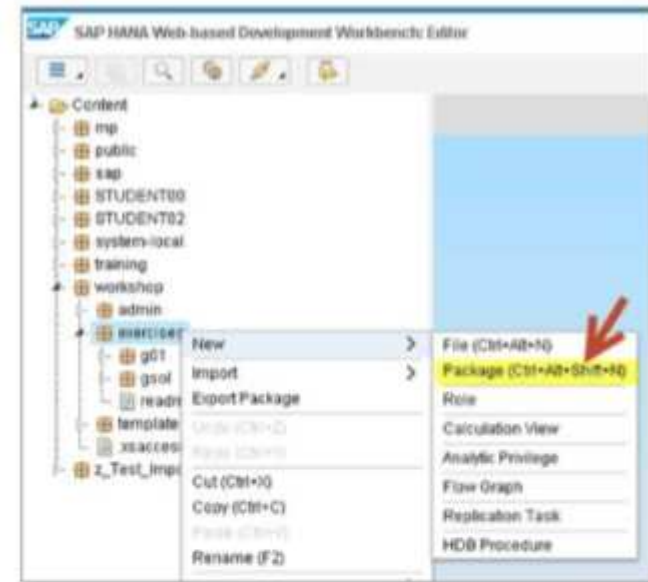
SAP HANA Web-based Development Workbench



XSC Project

Create Package

- Switch to the Editor Hierarchy
- Use the Context Menu -> New -> Package menu entry
- Provide the data for the package
- Use the button Create to insert the package



Exercise

XSC Project

Create Package

- Create packages right below the Content node and call it <MATNR>, given the following hierarchy:
- Content
 - playground
 - honeycomb

Solution
XSC Project
Create Package

- Just follow the steps explained above.

XSC Project

Schema

- **Relational databases** contain a **catalog** that describes the various **elements** in the system. The catalog divides the database into **sub-databases** known as **schema**. A database schema enables you to logically group together objects such as **tables**, **views**, and **stored procedures**. Without a defined schema, you cannot write to the catalog.
- **SAP HANA Extended Application Services (SAP HANA XS)** enables you to create a database schema as a transportable design-time file in the repository. Repository files can be read by applications that you develop.

XSC Project Create Schema

Repository
File
.hdbschema



Schema
created in the
catalog

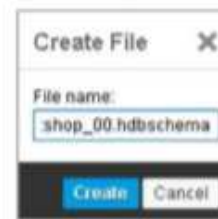
```
HANA_WORKSHOP_SOL.hdbschema 23
H00 (TRAINER) wdfibmt7215.wdf.sap.corp 00
schema_name = "HANA_WORKSHOP_SOL";
```



XSC Project

Create Schema

- Navigate package hierarchy to *workshop.exercises.g<group number>* (for example, g00).
- Press the right mouse button and choose *New* → *File*.
- Name the file *hana_workshop_<group number>.hdbschema* (for example, *hana_workshop_00.hdbschema*).
- Choose *Insert Snippet* button from the bar.
- Replace the default *MY_SCHEMA_NAME* with *hana_workshop_<group number>* (for example, *hana_workshop_00*).
- Save the file.
- The schema is automatically created in the Catalog.



© SAP

Exercise

XSC Project

Create Schema

- Create a schema file and call it `<MATNR>.hdbschema`.
- The name of the schema is `<MATNR>`

Solution

XSC Project

Create Schema

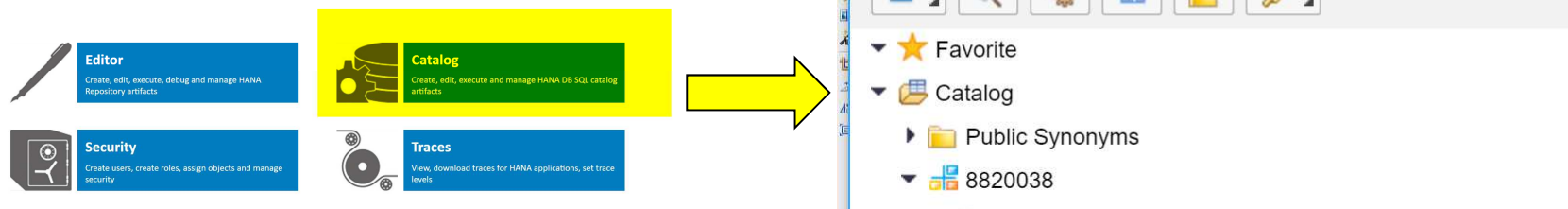
- Just follow the steps explained above.



The screenshot shows the SAP HANA Web-based Development Workbench: Editor interface. The left sidebar displays a project structure with folders like '8820038', 'honeycomb', 'honeycomb_ui', 'honeycomb.hdbschema', and 'playground'. The main editor area shows a file named 'honeycomb.hdbschema' with the content `schema_name="8820038";`. A 'Create File' dialog box is open, showing the file name 'honeycomb.hdbschema' and buttons for 'Create' and 'Cancel'.

- `schema_name=„<MATNR>“;`

SAP HANA Web-based Development Workbench



The diagram illustrates the SAP HANA Web-based Development Workbench interface. On the left, there are four main components: 'Editor' (Create, edit, execute, debug and manage HANA Repository artifacts), 'Security' (Create users, create roles, assign objects and manage security), 'Catalog' (Create, edit, execute and manage HANA DB SQL catalog artifacts), and 'Traces' (View, download traces for HANA applications, set trace levels). A yellow arrow points from the 'Catalog' component to the right, where a screenshot of the 'SAP HANA Web-based Development Workbench: Catalog' interface is shown. The Catalog view displays a tree structure with 'Favorite', 'Catalog', 'Public Synonyms', and '8820038'.

XSC Project

Roles

- SAP HANA uses roles to **control access** to the Web-based tool that enable you to maintain important parts of the application-development environment, for example, security and authentication methods.
- When using the Web-based tools provided by SAP HANA XS, the availability of features, screens, tabs, and UI controls (for example, Add, Edit, or Save, or Delete buttons) is based on privileges. **For the sake of convenience**, the specific privileges required to use the features provided with a particular tool have been **collected into a selection of specific roles**, which you assign to the user who wants to use a tool.
- For example,
 - a user assigned the **role sap.hana.xs.admin.roles::HTTPDestViewer** can display HTTP destinations but not change them in any way;
 - a user assigned the role **sap.hana.xs.admin.roles::SQLCCAdministrator** can view SQL connection configurations and modify them, too.

XSC Project

Roles

Privileges Types	Description
System Privileges	It controls normal system activity.
	System Privileges are mainly used for –
	Creating and Deleting Schema in SAP HANA Database
	Managing user and role in SAP HANA Database
	Monitoring and tracing of SAP HANA database
	Performing data backups
	Managing license
	Managing version
	Managing Audit
	Importing and Exporting content
	Maintaining Delivery Units
Object Privileges	Object Privileges are SQL privileges that are used to give authorization to read and modify database objects. To access database objects user needs object privileges on database objects or on the schema in which database object exists. Object privileges can be granted to catalog objects (table, view, etc.) or non-catalog objects (development objects). Object Privileges are as below –
	CREATE ANY
	UPDATE, INSERT, SELECT, DELETE, DROP, ALTER, EXECUTE
	INDEX, TRIGGER, DEBUG, REFERENCES

XSC Project

Roles

Privileges Types	Description
Analytic Privileges	Analytic Privileges are used to allow read access on data of SAP HANA Information model (attribute view, Analytic View, calculation View).
	This privilege is evaluated during query processing.
	Analytic Privileges grants different user access on different part of data in the
	Same information view based on user role.
	Analytic Privileges are used in SAP HANA database to provide row level data
	Control for individual users to see the data is in the same view.
Package Privileges	Package Privileges are used to provide authorization for actions on individual packages in SAP HANA Repository.
Application Privileges	Application Privileges are required in In SAP HANA Extended Application Services (SAP HANA XS) for access application.
	Application privileges are granted and revoked through the procedures GRANT_APPLICATION_PRIVILEGE and REVOKE_APPLICATION_PRIVILEGE procedure in the _SYS_REPO schema.
Privileges on User	It is an SQL Privileges, which can grant by the user on own user.
	ATTACH DEBUGGER is the only privilege that can be granted to a user.

XSC Project

Create Dev Role

- Create a new role in the honeycomb folder: developer.hdbrole
- Use the authorizations in the next slide

The screenshot shows the SAP HANA Web-based Development Workbench: Editor interface. The left sidebar displays a content tree with folders like '8820038', 'honeycomb', 'honeycomb_ui', 'playground', 'public', 'sap', 'bc', 'hana', 'admin', 'backup', and 'democontent'. The 'honeycomb' folder is expanded, showing 'developer.hdbrole' and 'honeycomb.hdbschema'. The main area shows the role configuration for '8820038.honeycomb::developer'. The 'Granted Roles' tab is active, displaying a table of roles granted to this role.

Role	Origin
sap.hana.xs.ide.roles::EditorDeveloper	Design Time
sap.hana.xs.debugger::Debugger	Design Time

For Example

XSC Project

Auth for Dev Role 1

Granted Roles

- `sap.hana.xs.ide.roles::EditorDeveloper`
- `sap.hana.xs.debugger::Debugger`

System Privileges

- `repo.export`
- `repo.import`
- `repo.maintain_delivery_units`
- `repo.modify_change`
- `repo.modify_foreign_contribution`
- `repo.modify_own_contribution`
- `repo.work_in_foreign_workspace`

Exercise

XSC Project

Execute the following actions:

1. Create a Repository package in `<MATNR>` to store your development artifacts.
2. Create a schema to store database objects.
3. Create and run the test HelloWorld application from our Project.

Solution

XSC Project

Execute the following actions:

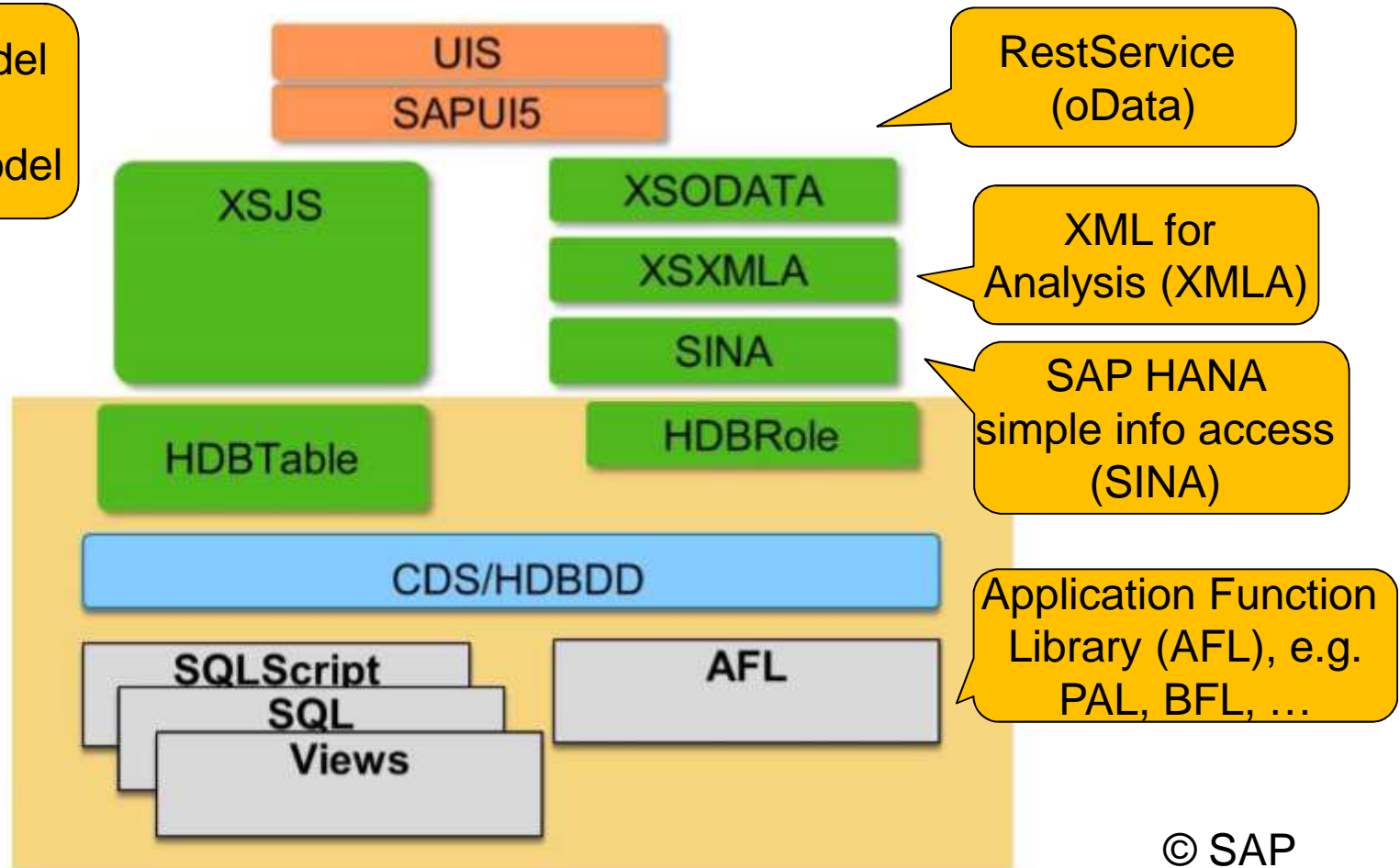
1. Create a Repository package in `<MATNR>` to store your development artifacts.
2. Create a schema to store database objects.
3. Create and run the test HelloWorld application from our Project.

THE PERSISTENCE MODEL

- Explain basic concepts of Core Data Services

Persistence Model

Persistence Model
vs
Consumption Model



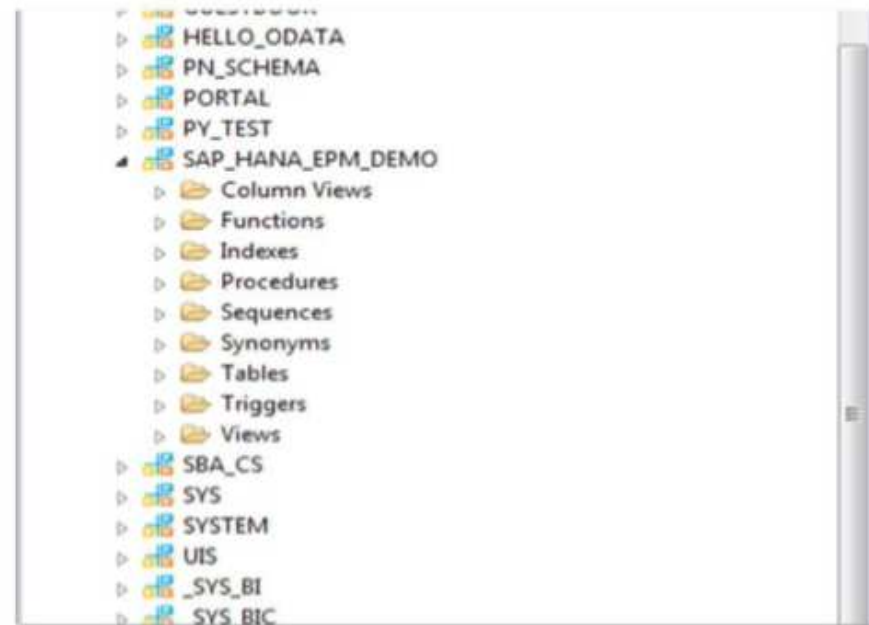
© SAP

Persistence Model

Data Persistence in SAP HANA

Data persistence objects

- Schemas
- Tables
- SQL views
- Sequences
- Procedures
-



Persistence Model

Create

To create objects:

At design time: the developer creates a descriptive representation of the DB object in the Repository.

On activation: the corresponding runtime object is created.

```
namespace workshop.exercises.gsol.data;  
@Schema: 'HANA_WORKSHOP_SQL'  
context User {  
    @Catalog.tableType: #COLUMN  
    Entity Details {  
        key PERS_NO: String(10);  
        FIRSTNAME: String(40);  
        LASTNAME: String(40);  
        E_MAIL: String(255);  
    };  
};
```



Table Name:				
workshop.exercises.gsol.data::User.Details				
Columns Indexes Further Properties Runtime Information				
	Name	SQL Data Type	Di...	Column Str
1	PERS_NO	NVARCHAR	10	STRING
2	FIRSTNAME	NVARCHAR	40	STRING
3	LASTNAME	NVARCHAR	40	STRING
4	E_MAIL	NVARCHAR	255	STRING

Persistence Model

Core Data Service

artifact Type	CDS	HDBTable (old)
Schema	.hdbschema	.hdbschema
Table	.hdbdd	.hdbtable
Table Type	.hdbdd	.hdbstructure
View	.hdbdd	.hdbview
Association	.hdbdd	-
Sequence	.hdbsequence	.hdbsequence
Structured Types	.hdbdd	
Data import	.hdbti	.hdbti

© SAP

Persistence Model

DB Schema for HANA

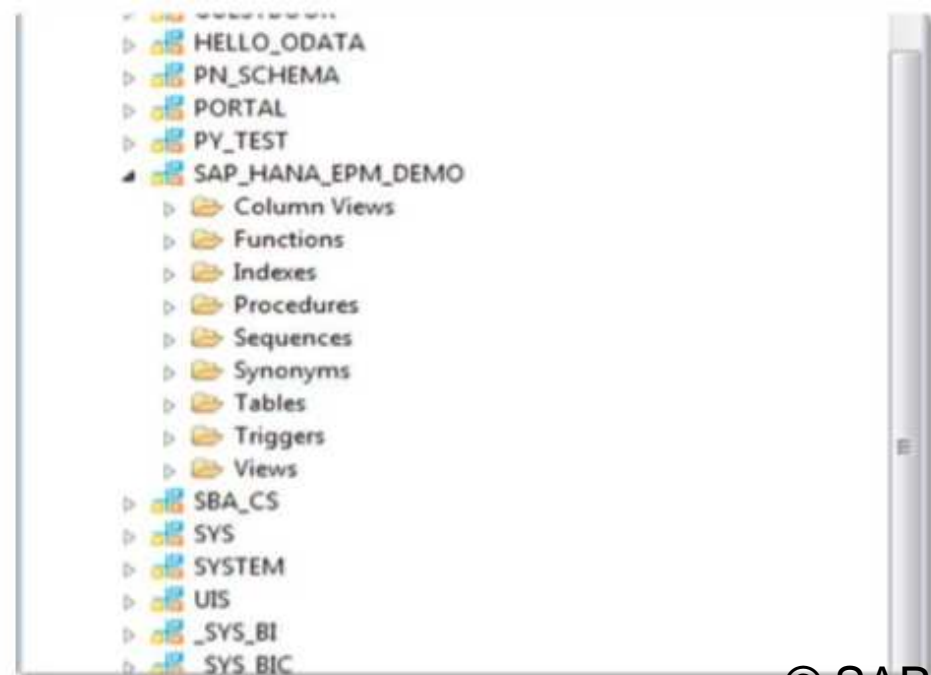
A schema defines the container that holds database objects such as tables, views, and stored procedures.

Database object

Mandatory

Contains logical groups:

- Tables
- Views
- Procedures
- Sequences
-



© SAP

Persistence Model

Creating a Schema

Repository
File
.hdbschema

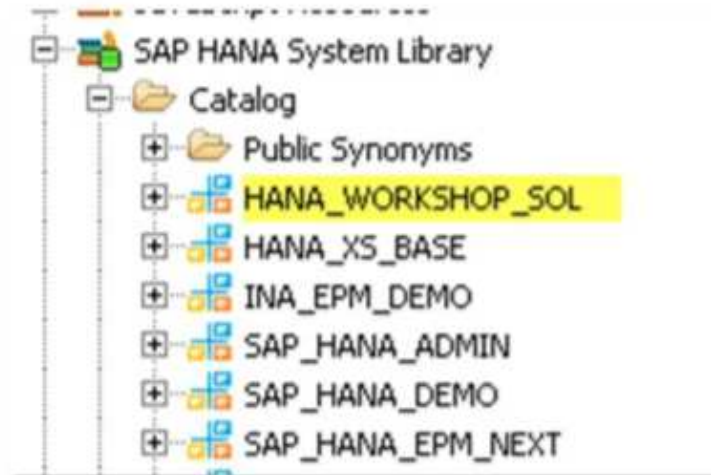
Activate

Schema
created in the
catalog

```
HANA_WORKSHOP_SOL.hdbschema
```

H00 (TRAINER) wdfbmt7215.wdf.sap.corp 00

```
schema_name = "HANA_WORKSHOP_SOL";
```



© SAP

Persistence Model

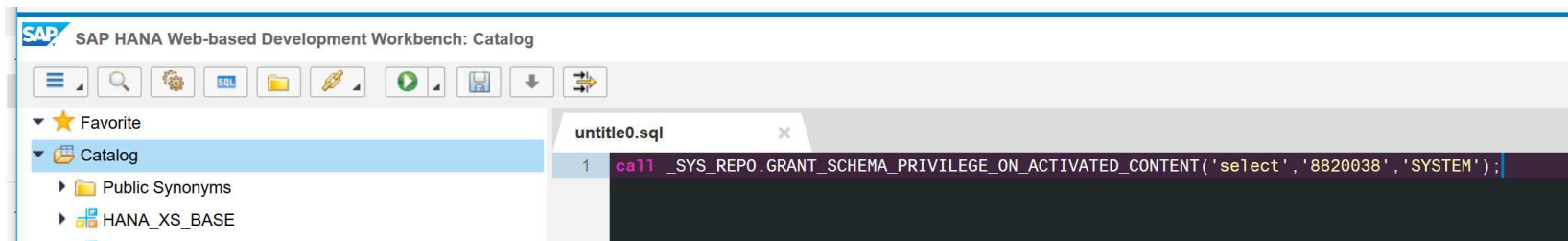
Authorizing Schema

- In order to see the schema in the Catalog you have to authorize the viewing user.
- Open the SQL console in the catalog and provide the command:

call

```
_SYS_REPO.GRANT_SCHEMA_PRIVILEGE_ON_ACTIVATED_CONTENT('select','<Schema>','<User>');
```

- Execute the command
- Refresh the catalog through the use of the context menu on the Catalog folder



Persistence Model

HDBTable

Database object

<package.path>::<TableName>

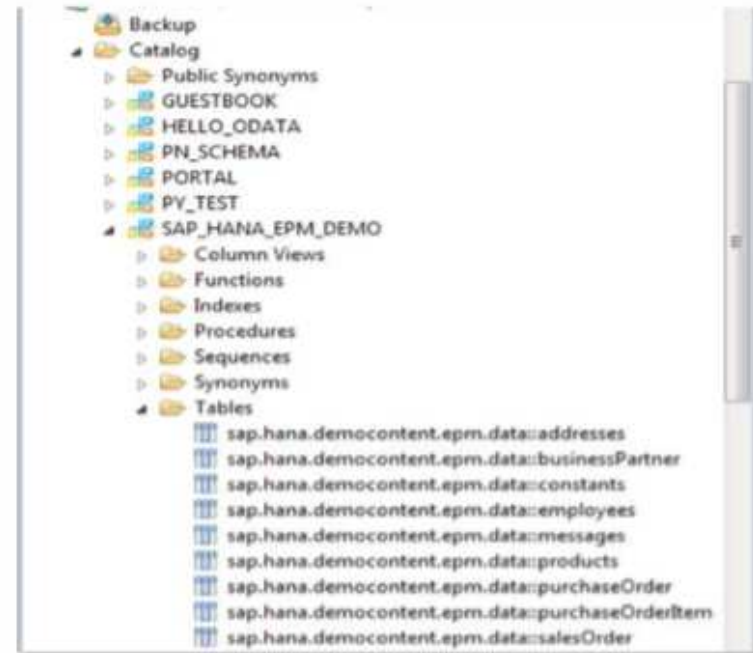
A set of data elements

Type:

- Column
- Row

Metadata:

- Constraints on table
- Constraints on values in particular columns...



Persistence Model

Creating a Table with HDBTable

Repository
File .hdbtable



Table created
in the catalog

```

table.schemaName = "HANA_WORKSHOP_SOL";
table.tableType = COLUMNSTORE;
table.columns =
[
  (name = "MY_COL1";  sqlType = VARCHAR;  length = 200; defau
  (name = "MY_COL2";  sqlType = INTEGER;  defaultValue = "1";
  (name = "MY_COL3";  sqlType = NVARCHAR; length = 200; defau
  (name = "MY_COL4";  sqlType = DECIMAL;  precision = 4; scal
  (name = "MY_COL5";  sqlType = DATE; nullable = false; defau
  (name = "MY_COL6";  sqlType = TIME; nullable = false; defau
  (name = "MY_COL7";  sqlType = TIMESTAMP; nullable = false;
];
table.primaryKey.pkcolumns = ["MY_COL1", "MY_COL2"];
    
```

Table Name:
/workshop.exercises.qsol.data::table

	Name	SQL Data Type	Di...	Column Store Data Type
1	MY_COL1	VARCHAR	200	STRING
2	MY_COL2	INTEGER		INT
3	MY_COL3	NVARCHAR	200	STRING
4	MY_COL4	DECIMAL	4,3	FIXED
5	MY_COL5	DATE		DAYDATE
6	MY_COL6	TIME		SECONDTIME
7	MY_COL7	TIMESTAMP		LONGDATE

© SAP

Persistence Model

Creating a Sequence

Repository
File
.hdbsequence

```
schema= "HANA_WORKSHOP_SOL";  
start_with= 1000000100;  
maxvalue= 1999999999;  
nomaxvalue=false;  
minvalue= 1;  
nominvalue=true;  
cycles= false;  
depends_on_table= "workshop.exercises.gsol.data::User.Details";
```

Activate

Sequence
created in the
catalog

Sequence Name:	workshop.exercises.gsol.data:userSeqId	Schema:	HANA_WORKSHOP_SOL
Start with:	1000000100		
Increment By:	1		
Minimum Value:	1		
Maximum Value:	1999999999		
Reset Query:			
<input type="checkbox"/> Cycle			

© SAP

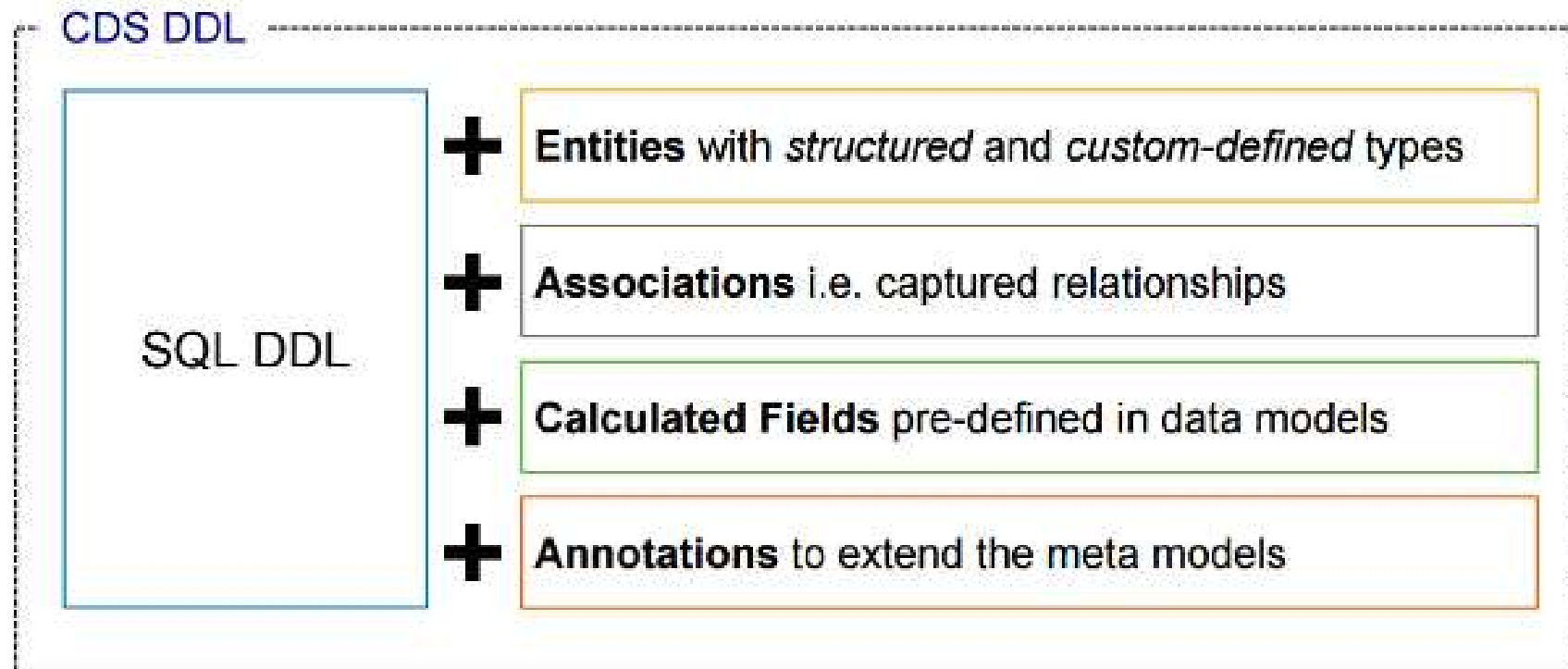
Persistence Model

Core Data Services

- To take advantage of SAP HANA for application development, SAP introduced a new **data modeling infrastructure** known as **core data services**.
- With CDS, data models are defined and **consumed** on the **database** rather than on the application server.
- CDS also offers capabilities beyond the traditional data modeling tools, including support for **conceptual modeling** and **relationship definitions, built-in functions, and extensions**.

Persistence Model

CDS Dev Model

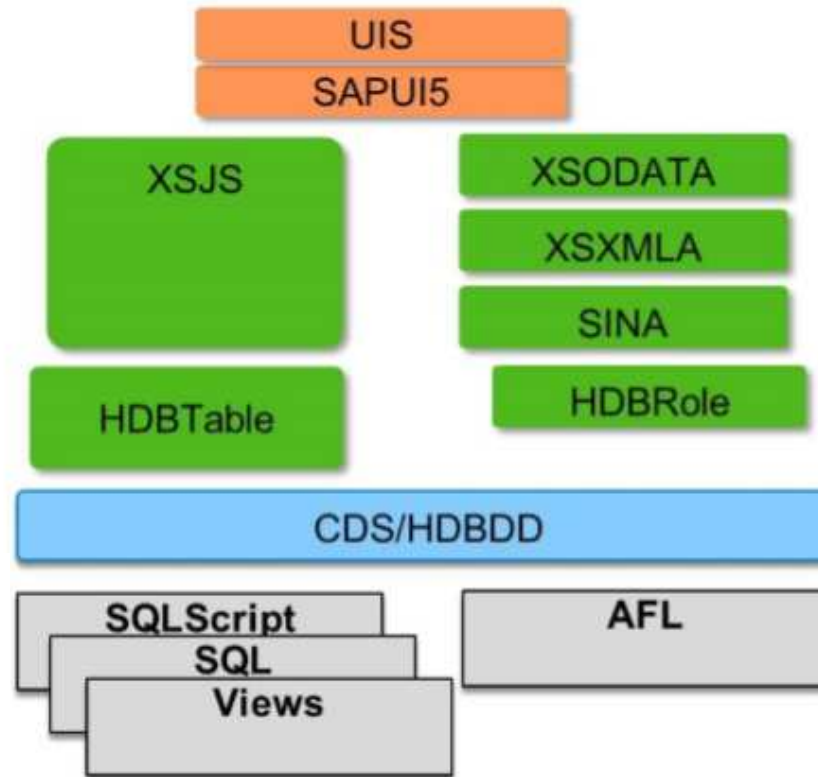
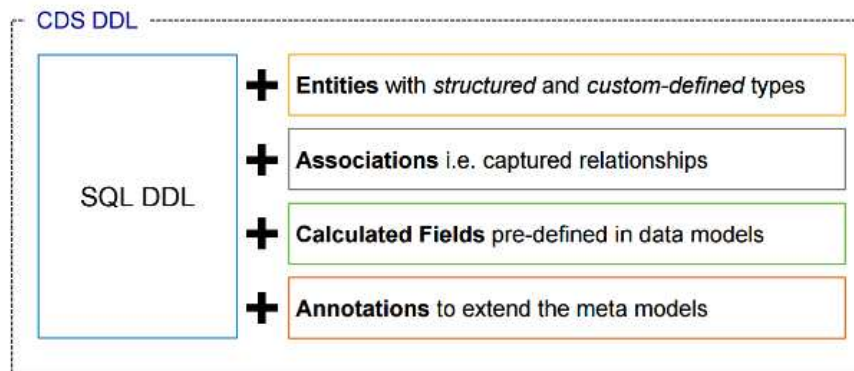


Persistence Model

CDS Dev Model

CDS – Core Data Services

- Data Definition Language
- Query Language
- Expression Language



Persistence Model

CDS Dev Artifacts

- New development artifact extension – hdbdd (Data Dictionary)
- Define the schema to use, reusable types, and multiple entities (tables), and views all within one source file
- Multiple catalog objects are generated upon activation

```
EmployeeModel.hdbdd
namespace demo.cds.CoreDataServicesDemo;

@Schema: 'SAPUIA'
context EmployeeModel {

    type Name : String(80);

    type FullName {
        firstName : Name;
        middleName : Name;
        lastName : Name;
    };

    type Street {
        street : String(80);
        number : String(8);
    };

    entity Address {
        key id : Integer;
        street : EmployeeModel.Street;
        city : String(40);
        zip : String(16);
    };
};
```

Persistence Model

CDS Associations

- Associations define relationships between Entities (tables)
- Not yet integrated into the database/SQL but can be referenced when defining views within the same CDS object

```
entity Address {
  owner : Association to Employee;
  key streeAddress : String(77);
  key zipCode : String(11);
  city : String(44);
  kind : String(10);
};

entity OrganizationUnit {
  key name : String(111);
  costcenter : String(44);
  manager : Association [0..1] to Employee;
  parent : Association to OrganizationUnit;
};

entity Employee {
  key ID : String(36); //UUID;
  name : String(77);
  salary : Amount; // Amount is a struct
  addresses : Association [0..*] to Address;
  orgunit : Association to OrganizationUnit;
}
```

Persistence Model

CDS Views

- CDS syntax expands to include the definition of views
- Views can use the defined associations between entities

```
define view EmployeesView as SELECT from Employee {  
  /*key*/ ID, name,  
  orgunit.name as orgunitname  
};
```

```
define view OrgUnitSalaries //with parameters  
as SELECT from Employee {  
  orgunit.name as orgunit,  
  SUM (salary) as totalSalaries  
} group by orgunit;
```

Persistence Model

Table Import

File .hdbti



File .csv



DB Table
filled with
data

```
import =  
[(  
  cdstable = "workshop.exercises.gsol.data::User.Details";  
  file = "workshop.exercises.gsol.data.loads:UserDetails.csv";  
  header = false;  
)];
```

```
1000000000, Akira, Kurosawa, akira.kurosawa@unknown.com  
1000000001, Federico, Fellini, federico.fellini@unknown.com  
1000000002, Igmarr, Bergman, igmar.bergman@unknown.com  
1000000003, John, Ford, john.ford@unknown.com
```

Raw Data | Distinct values | Analysis

Filter pattern 4 rows retrieved - 21 ms

PERS_NO	FIRSTNAME	LASTNAME	E_MAIL
1000000000	Akira	Kurosawa	akira.kurosa...
1000000001	Federico	Fellini	federico.felli...
1000000002	Igmarr	Bergman	igmar.bergm...
1000000003	John	Ford	john.ford@u...

Excercise

Create CDS Tables in XSC

1. In the exercises-<MATNR> project, create a new package **honeycomb_data** and within create a data definition file named **honeycomb.hdbdd**.
2. In the file, within the honeycomb context, define an entity named **Measures**, with the columns defined on the next slide.
3. Activate the file.
4. Check that the DB table was correctly created in the SB schema.

Excercise

Column Details

- BEEKEEPERID: String(10); -> OK
- HONEYCOMBID:String(4); -> OK
- MEASUREDATE: LocalDate; -> Format yyyy-mm-dd, e.g. 2017-01-07
- MEASURETIME: LocalTime; -> Format hh:mm:ss, e.g. 09:05:00
- WEIGHT: Decimal(5,2); -> e.g. 33.00
- WEIGHTUNIT: String(2); -> Empty or unit, e.g. kg (SI-Writing)
- TEMPERATURE: Decimal(5,2); -> e.g. 27.50
- TEMPERATUREUNIT: String(2); -> Empty or ([C|F], e.g. C
- BEEKEEPERNAME: String(40); -> Optional

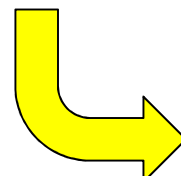
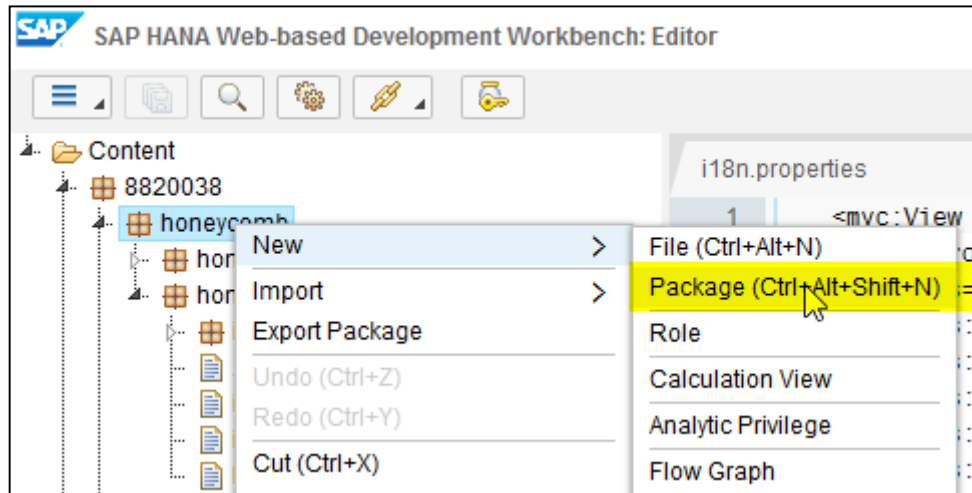
Excercise

Hints

- Numbers have to be quoted in the namespace
- CDS Primitive Types:
https://help.sap.com/saphelp_hanaplatform/helpdata/en/cf/394efd3fb4400f9c09d10315028515/content.htm

Solution

Honeycomb data package



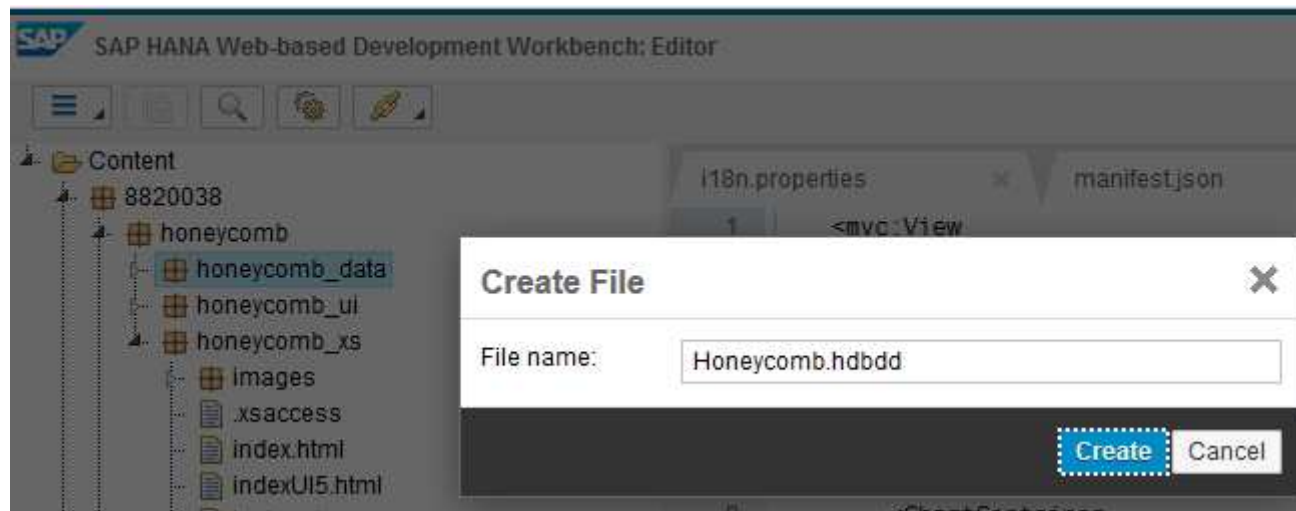
The 'Create Package' dialog box is shown with the following fields:

- Package name:
- Description:
- Responsible:
- Original Language:

Buttons:

Solution

Honeycomb CDS file



Solution

Honeycomb CDS file

```
namespace 8820038.honeycomb.honeycomb_data;  
@Schema: '8820038'  
context Honeycomb {  
  @Catalog.tableType: #COLUMN  
  Entity Measures {  
    KEY ID: Integer;  
    BEEKEEPERID: String(10);  
    HONEYCOMBID: String(4);  
    MEASUREDATE: LocalDate;  
    MEASURETIME: LocalTime;  
    WEIGHT: Decimal(5,2);  
    WEIGHTUNIT: String(2);  
    TEMPERATURE: Decimal(5,2);  
    TEMPERATUREUNIT: String(2);  
    BEEKEEPERNAME: String(10);  
  };  
};
```

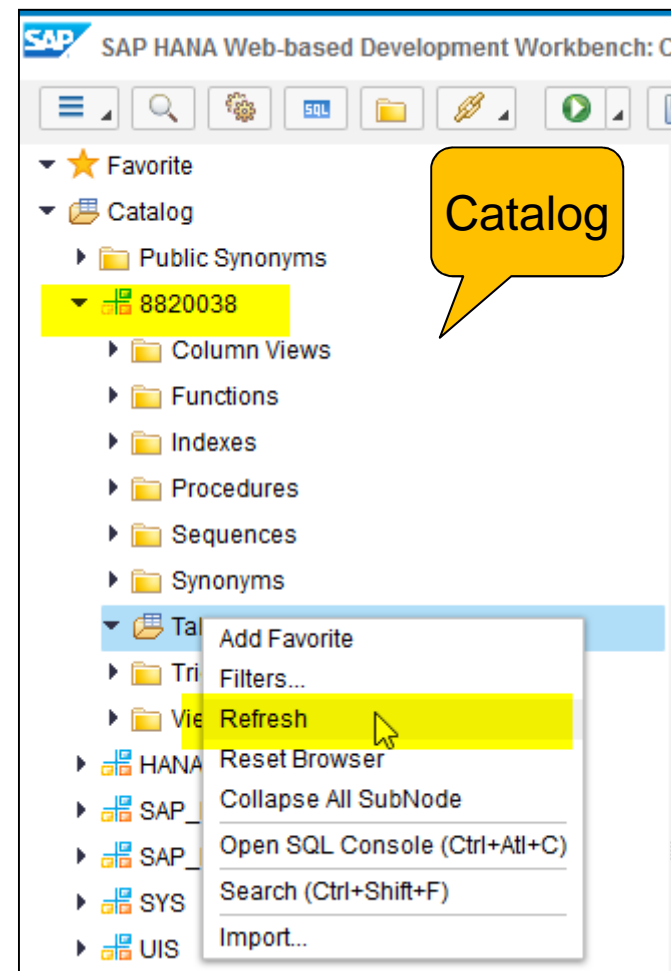
Path in the editor.
!Note the quoted number!

The schema from the catalog view

Context.Entity
A means to cluster the tables

Solution

CDS Activation 1



Solution

CDS Activation 2

SAP HANA Web-based Development Workbench: Catalog v 1.120.14 | Help | 8820038 | NEO_P020420TRIAL_WS201617 | I15 (vsa945722.nwtrial.od.sap.biz:00) |

8820038.honeycomb... x

Table Name: 8820038.honeycomb.honeycomb_data:Honeycomb.Measures Schema: 8820038 Type: COLUMN Open Content...

	Name	SQL Data Type	Dim	Column Store D...	Key	Not Null	Default	Comment
1	ID	INTEGER		INT	(X1)	X		
2	BEEKEEPERID	NVARCHAR	10	STRING				
3	HONEYCOMBID	NVARCHAR	4	STRING				
4	MEASUREDATE	DATE		DAYDATE				
5	MEASURETIME	TIME		SECONDTIME				
6	WEIGHT	DECIMAL	5,2	FIXED				
7	WEIGHTUNIT	NVARCHAR	2	STRING				
8	TEMPERATURE	DECIMAL	5,2	FIXED				
9	TEMPERATUREUNIT	NVARCHAR	2	STRING				
10	BEEKEEPERNAME	NVARCHAR	10	STRING				



Solution

Import Data

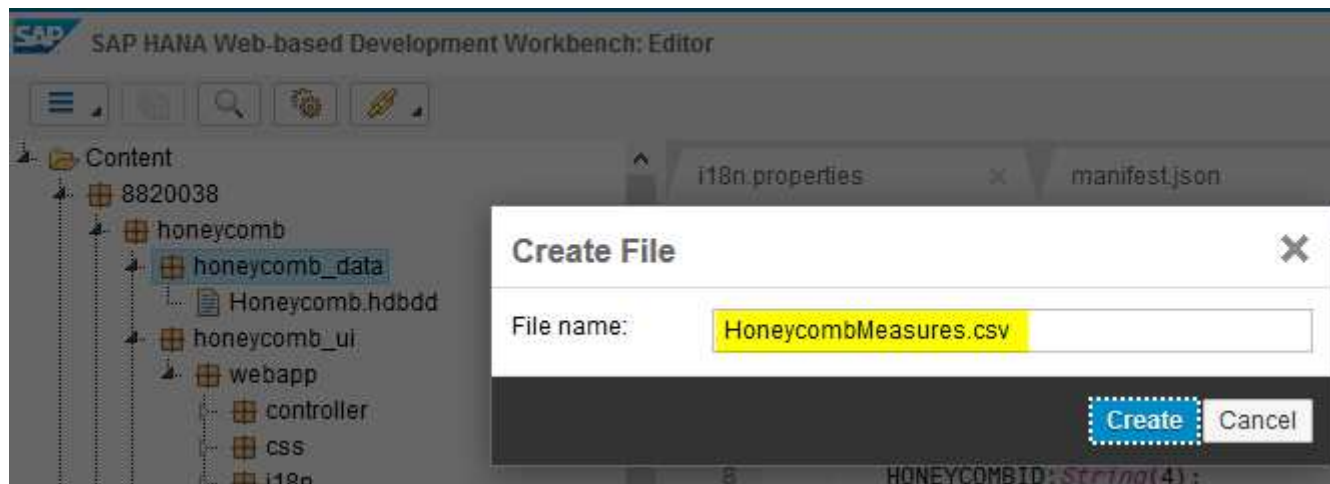
- Create a new file **honeycomb_data/HoneycombMeasures.csv** containing the lines defined in the next slide.
- Execute a table import of the file in the **Honeycomb.Measures** table that you created in the previous exercise. Define the import definition **Honeycomb.hdbti**

Solution

Data File

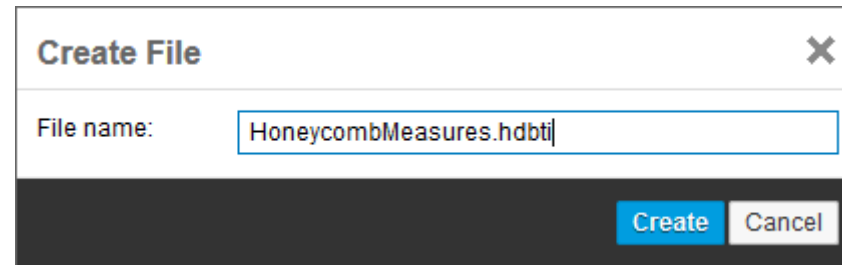
1,1,0001,2017-01-01,10:00:00,30.0,kg,27.00,C,lois
2,1,0001,2017-01-02,10:00:00,31.0,kg,27.00,C,lois
3,1,0001,2017-01-03,10:00:00,33.0,kg,27.00,C,lois
4,1,0001,2017-01-04,10:00:00,37.0,kg,27.00,C,lois
5,1,0001,2017-01-05,10:00:00,29.0,kg,27.00,C,lois
6,1,0001,2017-01-06,10:00:00,35.0,kg,27.00,C,lois

Solution Data File



Solution

Import File



Create File

File name: HoneycombMeasures.hdbti

Create Cancel

```
import =  
{  
  cdstable = 8820038.honeycomb.honeycomb_data::Honeycomb.Measures;  
  file = 8820038.honeycomb.honeycomb_data:HoneycombMeasures.csv;  
  header = false;  
};
```

Save File = Activation

Editor Console

```
[09:41:59] File /8820038/honeycomb/honeycomb_data/Honeycomb.hdbti saved & activated successfully.  
[09:44:15] File /8820038/honeycomb/honeycomb_data/HoneycombMeasures.csv saved & activated successfully.  
[09:44:39] File /8820038/honeycomb/honeycomb_data/Honeycomb.hdbti saved & activated successfully.
```

Solution

Import File

SAP HANA Web-based Development Workbench: Catalog

v 1.120.14 | Help | 8820038 | NEO_P020420TRIAL_WS201617 | H15 (vsa945722.nwtrial.od.sap.biz 00) |

Now editing: 8820038/8820038.honeycomb.honeycomb_data::Honeycomb.Measures

ID	BEEKEEPERID	HONEYCOMBID	MEASUREDATE	MEASURETIME	WEIGHT	WEIGHTUNIT
1	1	0001	01.01.2017	10:00:00	30	kg
2	1	0001	02.01.2017	10:00:00	31	kg
3	1	0001	03.01.2017	10:00:00	33	kg
4	1	0001	04.01.2017	10:00:00	37	kg
5	1	0001	05.01.2017	10:00:00	29	kg
6	1	0001	06.01.2017	10:00:00	35	kg

Catalog
Show Content

Notes

- Once the database tables are activated again the order of the columns may change in the table randomly. Thus you have to check the order and possibly change it in the csv file.
- Use Integer for foreign keys! String does not work.

ODATA SERVICES

- Explain basic concepts of Odata in SAP HANA
- OData Services on XSC

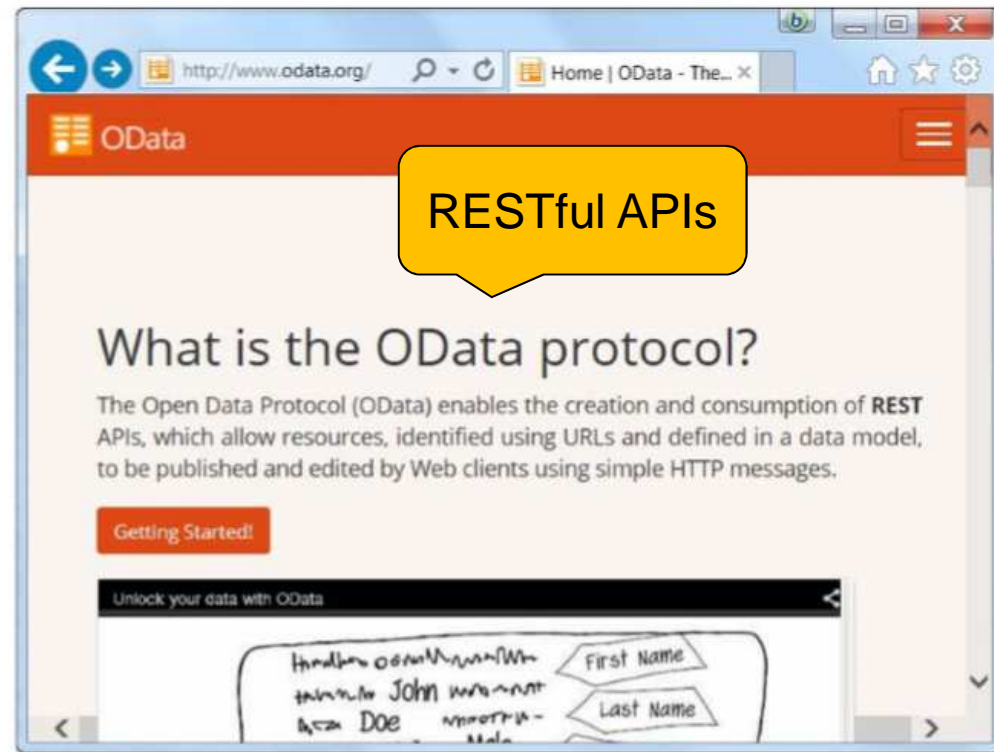
oData

Intro

The Open Data Protocol (OData) enables the creation and consumption of REST APIs, which allow resources to be published and edited by Web clients using simple HTTP messages.

It is an open standard, by the OASIS consortium.

Detailed reference information at <http://www.odata.org/>



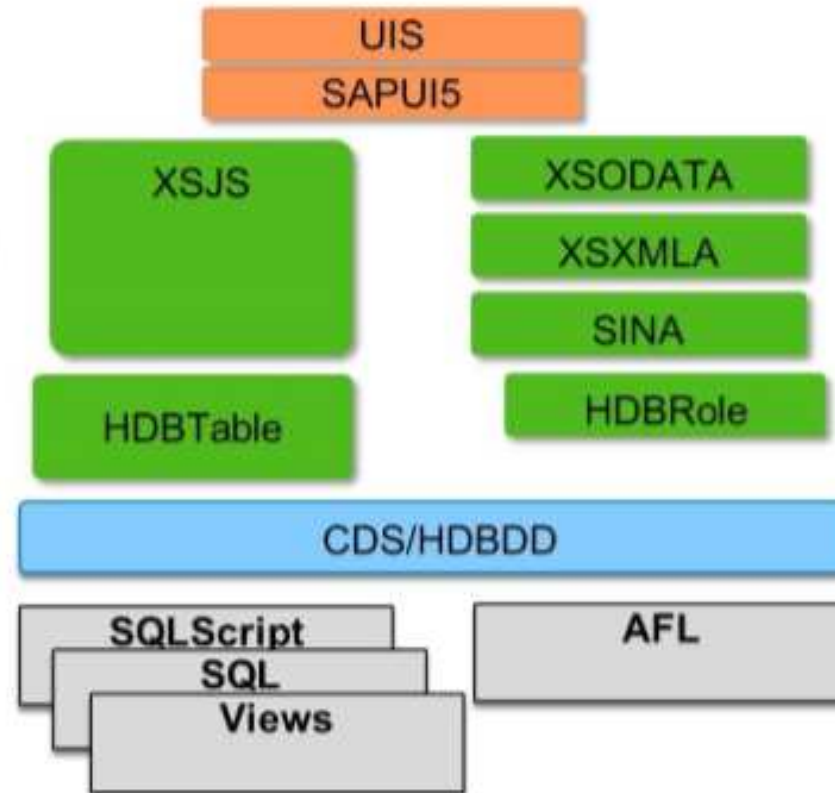
© SAP

oData

oData in SAP HANA

XSODATA

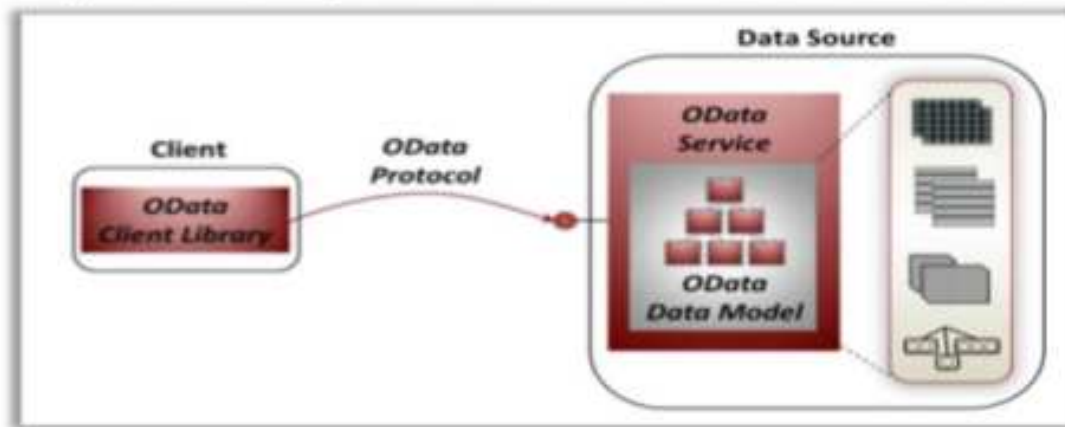
- Low coding OData REST service generation framework
- Based on existing tables and views
- Create/Update/Deletion operations support added in SPS 06



© SAP

oData Basics

- **oData Data Model**
Organize/describe data with Entity Data Model (EDM)
- **oData Protocol**
REST-based create, read, update, and delete + oData-defined query language
- **Client Libraries**
Pre-built libraries to request oData and display results
- **oData Services**
Exposes an end point that allows access to data in the SAP HANA database

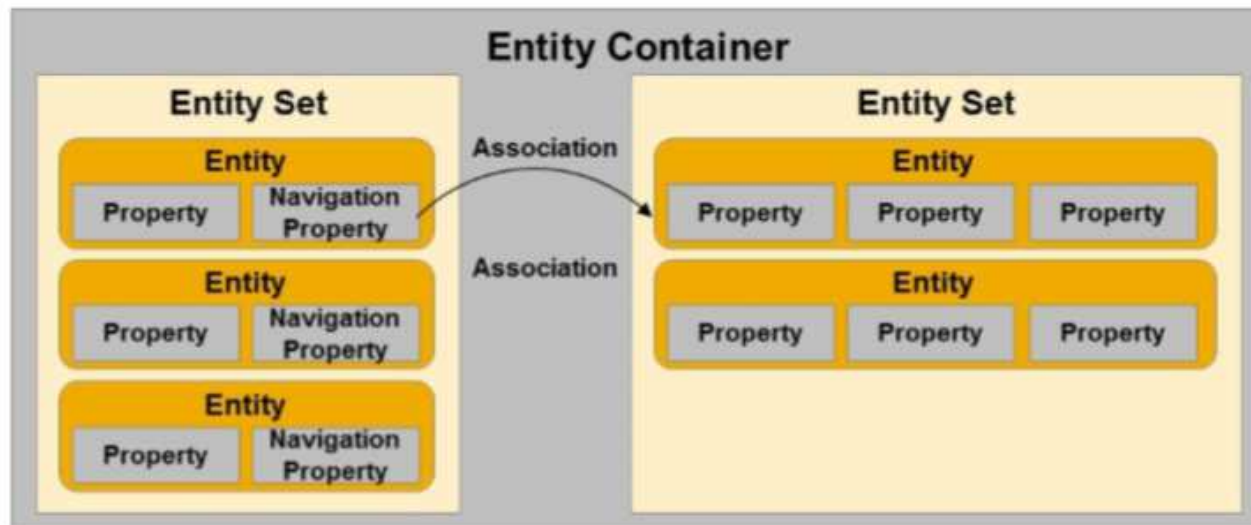


© SAP

oData

Funcs in SAP HANA

- **Aggregation**
Use functions to aggregate columns
- **Associations**
Express relationships between entities
- **Key Specification**
EntityType denotes a set of properties forming a unique key
- **Parameter Entity Sets**
Input parameters for SAP HANA calculation views and analytic views
- **Property Projection**
Restrict the number of selected columns to expose



© SAP

oData Dev Process

- **Data End Point**
Define which data to expose (for example, table)
- **UI Call to OData Service**
Link UI table view to the OData service
- **OData Service Definition**
Define mechanism to expose authorized data
- **OData Display/Test**
URL filters: \$top, \$format, \$orderby, \$filter, \$select...

Partner ID	Email Address	Phone Number	Fax Number	Legal Form	Currency	Web Addr
0100000000	karl.mueller@sap.c	0622734567		AG	EUR	http://www.i
0100000001	dagmar.schulze@b	3088530		GmbH	EUR	http://www.i
0100000002	maria.braun@delbr	3023352668		Ltd.	USD	http://www.i
0100000003	saskia.sommer@ta	511403266		GmbH	EUR	http://www.i
0100000004	bob.buyer@panora	2244998800		Inc.	USD	http://www.i
0100000005	bart.loening@tecum	2511415		AG	EUR	http://www.i
0100000006	yoko.nakamura@a	9078563412		Inc.	JPY	http://www.i
0100000007	sophie.ribery@laun	149744423		S.A.	EUR	http://www.i
0100000008	victor.sanchez@avi	7257716		S.A.	MXN	http://www.i
0100000009	jorge.velaz@teleco	1133557799		S.A.	ARS	http://www.i

© SAP

oData Service Def

An OData service is defined with the activation of an .xsodata file in the Repository.

```
Service (service {  
    "sample.odata::table" as "MyTable";  
})
```

Resulting service at:

```
ht<service xml:base="http://localhost:8002/sample/odata/repo.xsodata/"  
  xmlns:atom="http://www.w3.org/2005/Atom"  
  xmlns:app="http://www.w3.org/2007/app" xmlns="http://www.w3.org/2007/app">  
  <workspace>  
    <atom:title>Default</atom:title>  
    <collection href="MyTable">  
      <atom:title>MyTable</atom:title>  
    </collection>  
  </workspace>  
</service>
```

© SAP

oData Service Def

Resulting service metadata:

`http://myHost:myPort/sample/odata/repo.xsodata/$metadata`

```
<edmx:Edmx Version="1.0"
xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx">
  <edmx:DataServices m:DataServiceVersion="2.0"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
    <Schema Namespace="sample.odata.repo"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns="http://schemas.microsoft.com/ado/2007/05/edm">
      <EntityType Name="MyTableType">
        <Key>
          <PropertyRef Name="ID"/>
        </Key>
        <Property Name="ID" Type="Edm.Int32" Nullable="false"/>
        <Property Name="Text" Type="Edm.String" Nullable="true"
MaxLength="1000"/>
        <Property Name="Time" Type="Edm.DateTime" Nullable="true"/>
      </EntityType>
      <EntityContainer Name="repo" m:IsDefaultEntityContainer="true">
        <EntitySet Name="MyTable"
EntityType="sample.odata.repo.MyTableType"/>
      </EntityContainer>
    </Schema>
  </edmx:DataServices>
</edmx:Edmx>
```

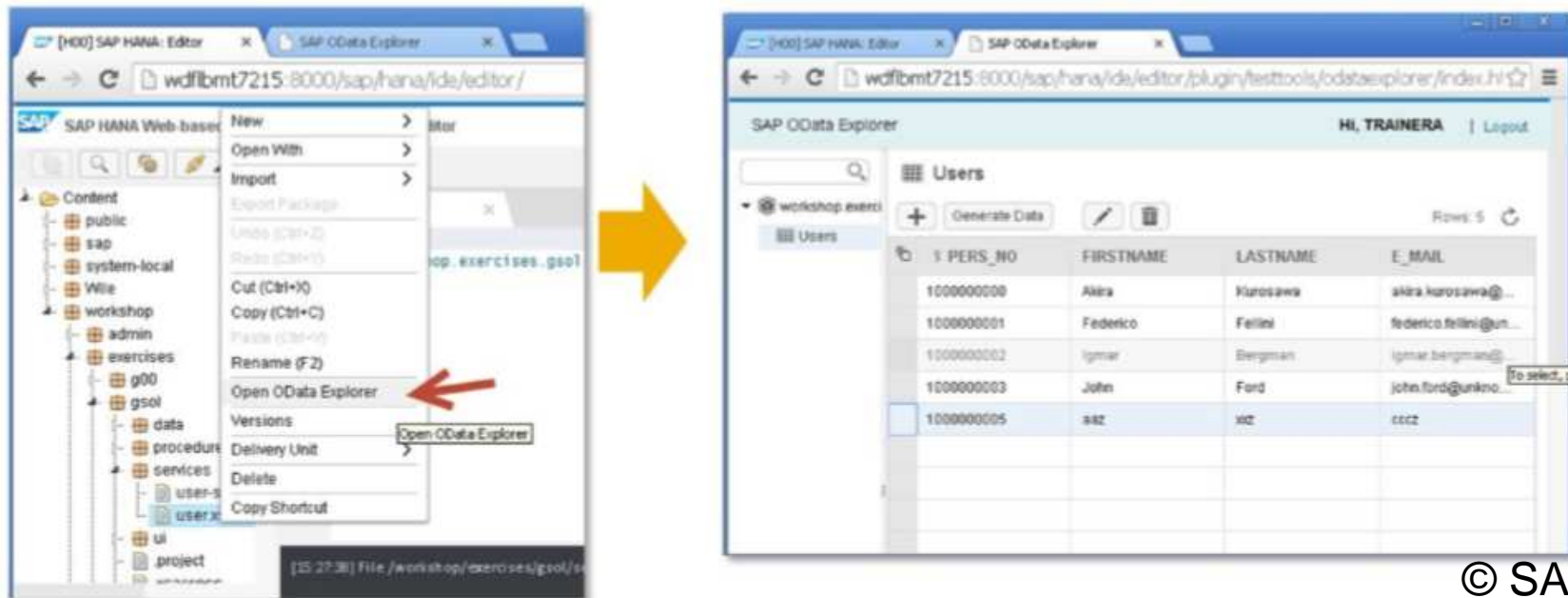
© SAP

oData Explorer

SAP OData Explorer allows you to browse the data accessible via an OData service.

In addition, it allows you to execute *Create*, *Update*, and *Delete* actions on the service content.

It is accessible via the Web-based IDE Editor, running Open OData Explorer from the context menu.



© SAP

oData

Read / Write

By default, the created OData service is write-enabled, so a user with proper authorizations on the destination table can use it to create, change, or delete table lines.

It is possible to forbid data modification via explicit options:

```
service { "sap.test::myTable"  
    create forbidden  
    update forbidden  
    delete forbidden;  
}
```

oData

DataBinding in SAPUI5

- Read-only binding with OData is similar to binding with JSON.
- You just need to use a different data model type.

JSON-Model:

```
var oModel = new sap.ui.model.json.JSONModel(dataUrlOrData);
```

OData-Model:

```
var oModel = new sap.ui.model.odata.ODataModel(dataUrl [, useJSON, user, pass] );
```


oData

Two-Way Binding

To realize a **write-enabled** OData binding, you need to:

- Activate two-ways binding in the model:

```
oModel = new sap.ui.model.odata.ODataModel(  
    "/workshop/exercises/gsol/services/user.xsodata/", true  
);  
oModel.setDefaultBindingMode(sap.ui.model.BindingMode.TwoWay);
```

- Call properly, when required, the three model methods:

```
model.create()
```

```
model.remove()
```

```
Model.submitChanges()
```

© SAP

oData oModel.create()

```
var oEntry = {};  
oEntry.PERS_NO      =  
oEntry.FIRSTNAME   =  
oEntry.LASTNAME     =  
oEntry.E_MAIL      =  
  
oModel.create('/Users', oEntry, null,  
  function() {alert("Create successful"); },  
  function() {alert("Create failed");} );
```

New table entry

Callback functions

Binding path

© SAP

oData

oModel.remove()

```
model.remove( context.sPath, {  
    fnSuccess: function(oData, oResponse) {  
        alert("User deleted successfully.");  
        model.refresh();  
    },  
    fnError: function() {  
        alert("Could not delete the user.");  
    }  
});
```

Binding path

Callback functions

© SAP

oData

oModel.submitChanges()

```
oModel.submitChanges (  
    function() {alert("Update successful");},  
    function() {alert("Update failed");} );
```



**Callback
functions**

© SAP

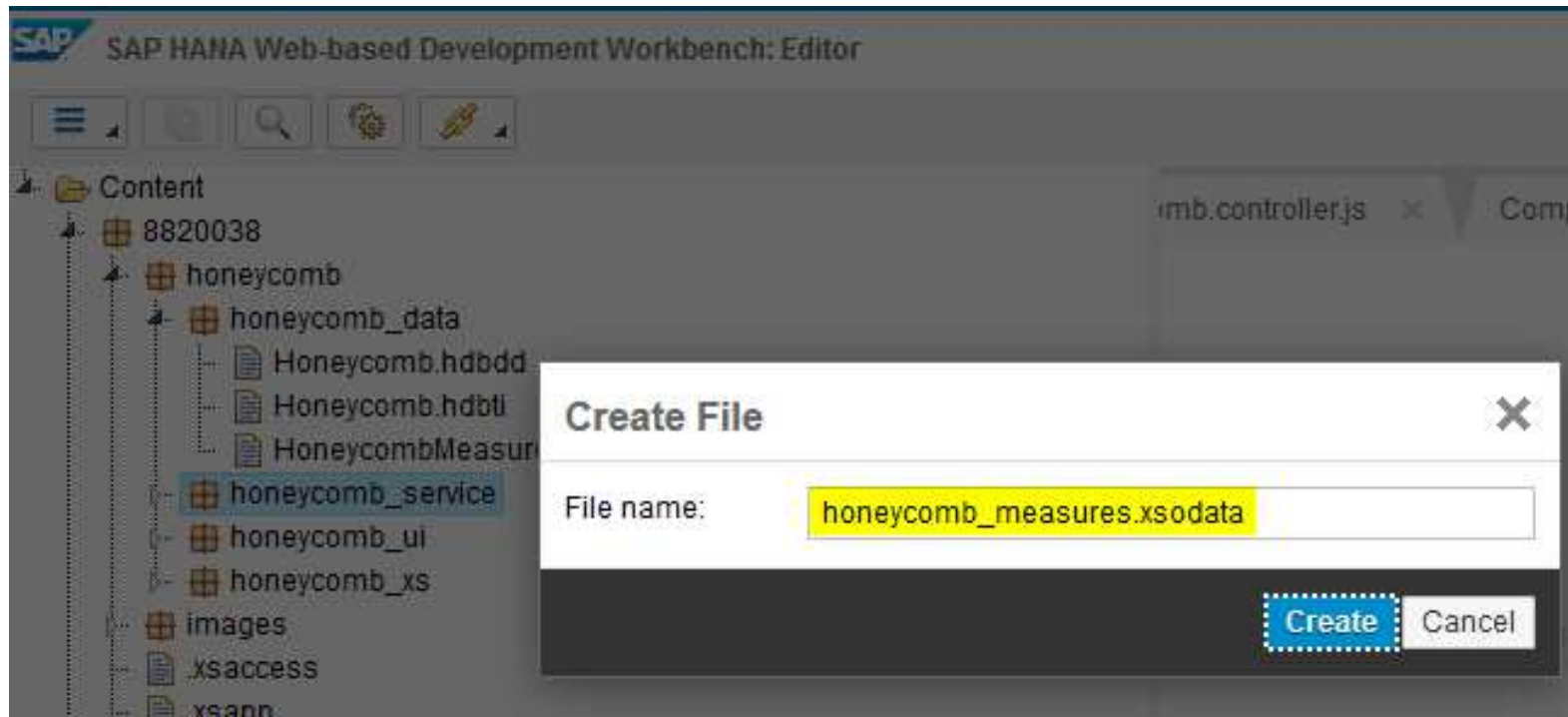
Exercise

oData

- Create an **honeycomb_service/honeycomb_measures.xsodata** service for the table **Honeycomb:Measures**.
- Test your service (oData Explorer).
- Modify the **honeycomb ui (Honeycomb.controller.js)** so that it displays the data from the **honeycomb_measures.xsodata** service created in the previous step.
- Change the JSON data binding of the table to a **data binding** with the oData service created in the previous exercise.
- Activate and **run** the files.
- Create and set a formatter for the Date typed columns.
- Activate and run the files.

Solution

honeycomb_measures.xsodata



Solution

honeycomb_measures.xsodata

```
service{
    "8820038.honeycomb.honeycomb_data::Honeycomb.Measures" as
    "HCMeasures";
}
```

Pfad:

[https://ws20162017p1942292818trial.hana.trial.ondemand.com/8820038/honeycomb/honeycomb_service/honeycomb_measures.xsodata/HCMeasures/\\$format=json](https://ws20162017p1942292818trial.hana.trial.ondemand.com/8820038/honeycomb/honeycomb_service/honeycomb_measures.xsodata/HCMeasures/$format=json)

Solution Test oData

The screenshot shows the SAP HANA Web-based Development Workbench Editor interface. On the left, a file tree displays the project structure under 'Content'. A context menu is open over the file 'honeycomb_measures.xodata'. A yellow arrow points from the 'Open OData Explorer' option in the menu to the SAP OData Explorer window on the right. The OData Explorer window shows the 'HCMeasures' entity selected, with a table of data below it.

ID	BEEKEEPERID	HONEYCOMBID	MEASUREID
1	1	0001	01.0
2	1	0001	02.0
3	1	0001	03.0
4	1	0001	04.0
5	1	0001	05.0
6	1	0001	06.0

Solution

Change App for oData

1) Change manifest.json for the new data source in the „sap.app“ section.

```
"dataSources": {  
  „hcmeasureRemote“: {  
    "uri": "8820038/honeycomb/honeycomb_service/honeycomb_measures.xsodata/",  
    "type": "OData",  
    "settings": {  
      "odataVersion": "2.0"  
    }  
  }  
}
```

2) Add a new model in the manifest.json for the new data source in the „sap.ui5/models“ section.

```
"hcmeasure": {  
  "dataSource": "hcmeasureRemote"  
}
```

3) In Honeycomb.controller.js insert in init method the creation of the model.

Request URL

https://ws201617p020420trial.hanatrial.ondemand.com/8820038/honeycomb/honeycomb_service/honeycomb_measures.xsodata/HCMeasures

Solution

Change App for oData (B)

- **Problem:** manifest.json instantiation does not work, Component instantiation does not work, App.controller instantiation does not work
- **Solution:** Instantiate oData JSON model in **init method** of Honeycomb.controller and store it it
`this.getOwnerComponent().setModel(„measures“)`
- **Change:** Binding of measures table in Honeycomb.controller
- **Problem:** oData service gives back JS Date object for Date column
- **Solution:** Create Formatter for corresponding columns

SQLSCRIPT

- Explain basic concepts about SQLScript
- Use the SQLScript Wizard and Editor
- Use the SQLScript Debugger
- Explain main SQLScript language features
- Use Functions in SQLScript
- Use Triggers in SQLScript
- Explain how the R language can be used in SAP HANA

http://help.sap.com/hana/SAP_HANA_SQL_Script_Reference_en.pdf

SQLScript Definition

SQLScript is:

- An interface for applications to access SAP HANA.
- An extension of ANSI Standard SQL.
- A language for creating stored procedures and user defined functions in HANA.
- A combination set based on declarative SQL with imperative control flow constructs, also suitable for mass data processing and OLTP scenarios.

Two reasons why this is required to achieve the best performance:

- Eliminates data transfer between database and application tiers.
- Calculations need to be executed in the database layer to gain maximum benefit from SAP HANA features, such as fast column operations, query optimization, and parallel execution.

SQLScript

SQL vs. SQLScript

Compared to plain SQL queries, SQLScript has the following advantages:

- Returns **multiple results**, while a SQL query returns only one result set.
- **Complex logic can be broken down into smaller chunks of code**. This enables modular programming, reuse and a better understanding by functional abstraction. For structuring complex queries, standard SQL only allows the definition of SQL views. However, SQL views have no parameters.
- Complex SQL statement can be well structured, using **table variables**, improving maintainability, and gaining performance by parallel execution based on statement dependencies. With standard SQL, it would be necessary to define globally visible views, even for intermediate steps.
- SQL Script has **control logic**, such as if/else, which is not available in SQL.

SQLScript Technologies

Front-end Technologies

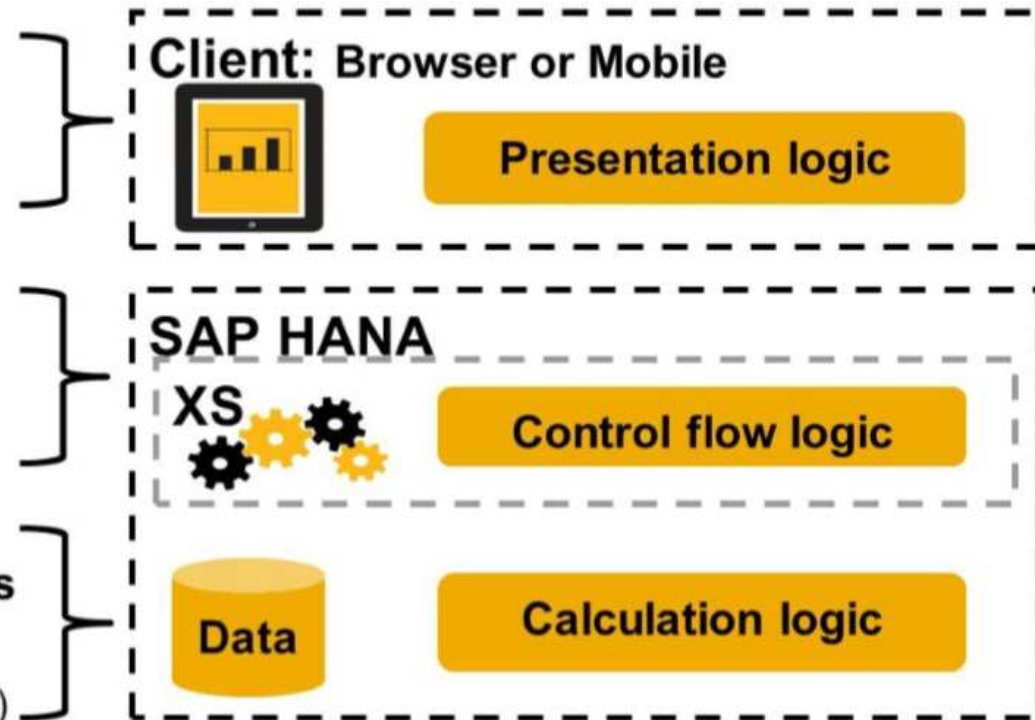
- http/s
- HTML5 / SAPUI5
- Client-side JavaScript

Control Flow Technologies

- OData
- Server-Side JavaScript
- XMLA

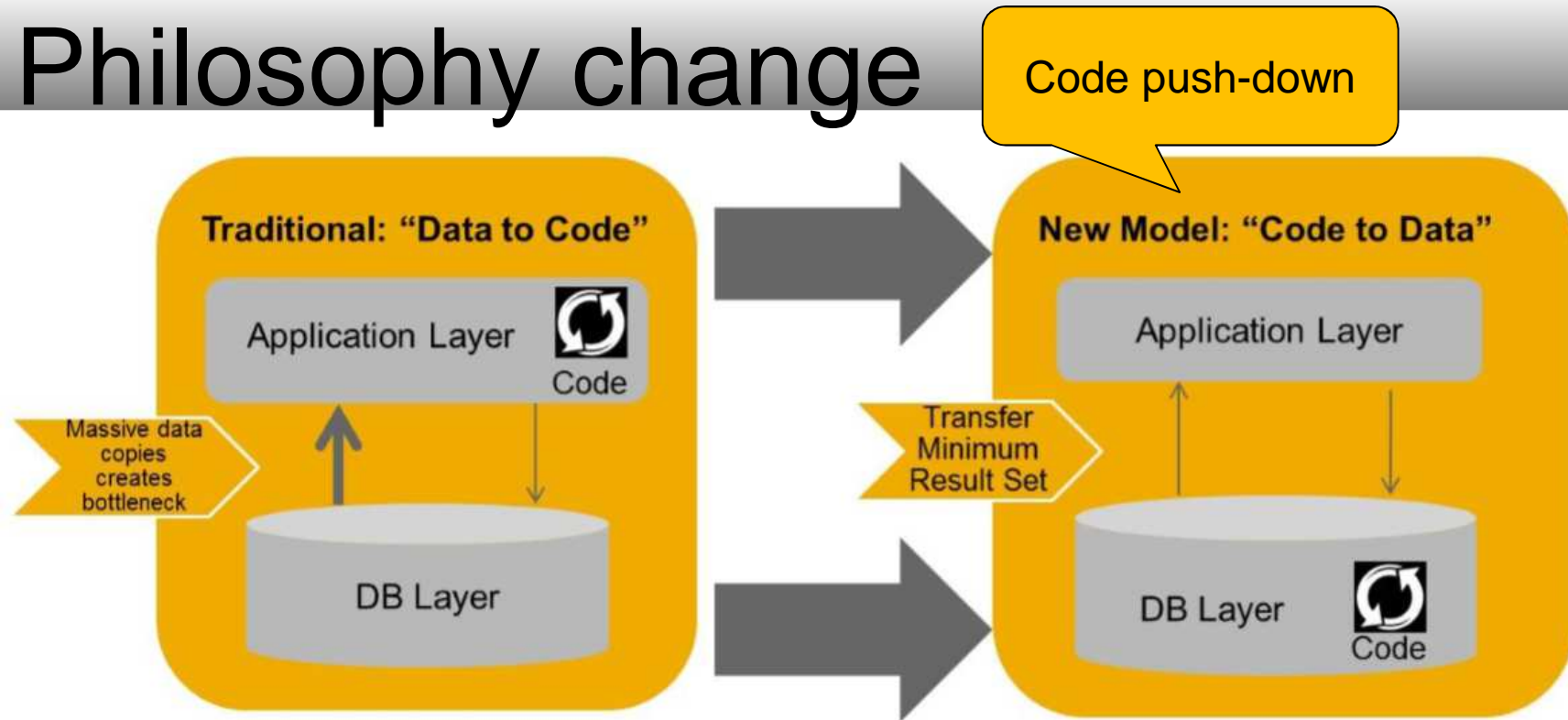
Data Processing Technologies

- SQL / SqlScript
- Application Function Library (AFL)



© SAP

SQLScript Philosophy change



Code push-down: The SAP Business Suite on HANA does this in a couple of ways, for example:

- in ABAP, we can now leverage SAP HANA views by exposing them to the ABAP dictionary via **External Views**.
- We can also expose **SQLScript procedures** and call them directly from ABAP using the `CALL DATABASE PROCEDURE` statement

© SAP

SQLScript Code example

```
BEGIN
...

-- Query 1
product_ids = select "ProductId", "Category", "DescId"
  from "SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::products"
  where "Category" = 'Notebooks' or "Category" = 'PC';

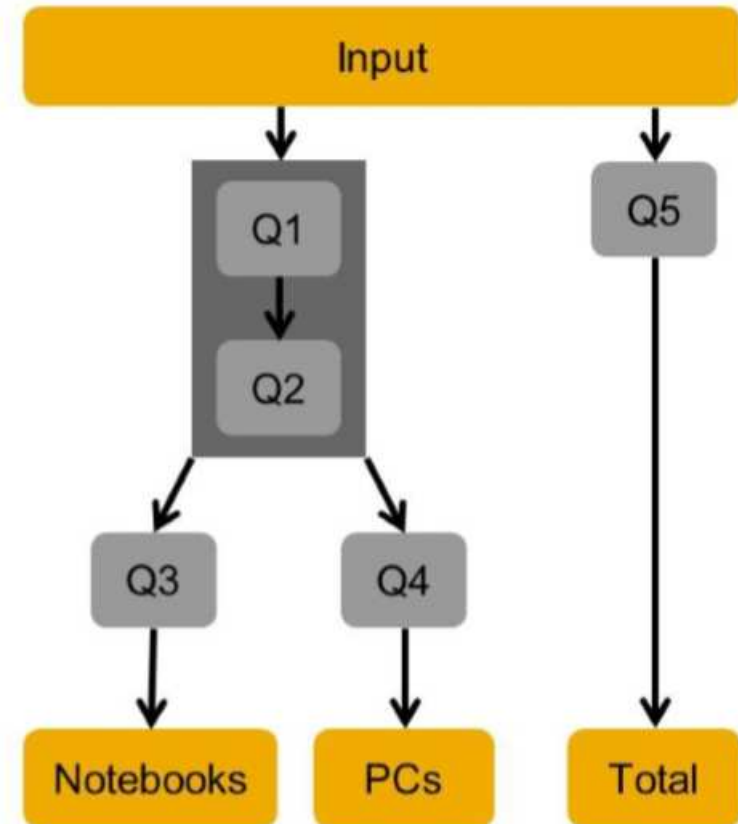
-- Query 2
product_texts = select "ProductId", "Category", "DescId", "Text"
  from :product_ids as prod_ids
  inner join "SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::texts"
  as texts on prod_ids."DescId" = texts."TextId";

-- Query 3
out_notebook_count = select count(*) as cnt from
  :product_texts where "Category" = 'Notebooks';

-- Query 4
out_pc_count = select count(*) as cnt from
  :product_texts where "Category" = 'PC';

-- Query 5
out_total_count = select count(*) as cnt
  from "SAP_HANA_EPM_DEMO"."sap.hana.democontent.epm.data::products";
...

END;
```



© SAP

SQLScript

Select

SELECT statements are executed in parallel unless:

- Any local scalar parameters and variables are used in the procedure
- Any read/write procedures or DML/DDDL operations are executed
- Any imperative logic is used within the procedure, such as IF statements or FOR loops
- Any SQL statements are used that are not assigned to an intermediate variable or parameter

SQLScript Schema

- Schema definition – developer can define which schema in which the run-time object of the procedure will be created.

```
SAP HANA SQLScript
1 PROCEDURE SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role" (
2     IN im_partnerrole NVARCHAR(3) DEFAULT '01',
3     OUT ex_bp_addresses SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.data::EPM.Procedures.tt_bp_addresses" )
4     LANGUAGE SQLSCRIPT
5     SQL SECURITY INVOKER
6     DEFAULT SCHEMA SAP_HANA_EPM_NEXT
7     READS SQL DATA AS
8 BEGIN
9 /*****
10  Write your procedure logic
11  *****/
12 ex_bp_addresses =
13     select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS", a."COMPANYNAME",
14         a."ADDRESSES.ADDRESSID" as "ADDRESSID", b."CITY", b."POSTALCODE", b."STREET"
15     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.BusinessPartner" as a
16         inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Addresses" as b|
17         on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
18         where a."PARTNERROLE" = :im_partnerrole;
19
20 END;
```

© SAP

SQLScript

Package Hierarchy

- Package hierarchy and procedure name – contains the complete package hierarchy as well as the name of the procedure, separated by double colon (::).

```
SAP HANA SQLScript
1 PROCEDURE SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role"
2   IN im_partnerrole NVARCHAR(3) DEFAULT '01',
3   OUT ex_bp_addresses SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.data::EPM.Procedures.tt_bp_addresses" )
4   LANGUAGE SQLSCRIPT
5   SQL SECURITY INVOKER
6   DEFAULT SCHEMA SAP_HANA_EPM_NEXT
7   READS SQL DATA AS
8   BEGIN
9   /*****
10    Write your procedure logic
11    *****/
12   ex_bp_addresses =
13   select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS", a."COMPANYNAME",
14          a."ADDRESSES.ADDRESSID" as "ADDRESSID", b."CITY", b."POSTALCODE", b."STREET"
15   from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.BusinessPartner" as a
16        inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Addresses" as b|
17        on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
18        where a."PARTNERROLE" = :im_partnerrole;
19
20 END;
```

© SAP

SQLScript

Input/Output parameter

- Input/output parameter definition – developer can define both input parameters with default values as well as output parameters. Parameters can reference simple types, or global types defined via CDS.

```
SAP HANA SQLScript
1 PROCEDURE SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.procedures::get bp addresses by role" (
2     IN im_partnerrole NVARCHAR(3) DEFAULT '01',
3     OUT ex_bp_addresses SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.data::EPM.Procedures.tt_bp_addresses" )
4     LANGUAGE SQLSCRIPT
5     SQL SECURITY INVOKER
6     DEFAULT SCHEMA SAP_HANA_EPM_NEXT
7     READS SQL DATA AS
8 BEGIN
9     /*****
10      Write your procedure logic
11      *****/
12 ex_bp_addresses =
13     select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS", a."COMPANYNAME",
14           a."ADDRESSES.ADDRESSID" as "ADDRESSID", b."CITY", b."POSTALCODE", b."STREET"
15     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.BusinessPartner" as a
16        inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Addresses" as b|
17        on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
18        where a."PARTNERROLE" = :im_partnerrole;
19
20 END;
```

© SAP

SQLScript Metadata

- Metadata declarations: developer can set language(SQLScript/R), security(invoker/definer), default schema, and read/write access.

```
SAP HANA SQLScript
1 PROCEDURE SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role" (
2     IN im_partnerrole NVARCHAR(3) DEFAULT '01',
3     OUT ex_bp_addresses SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.data::EPM.Procedures.tt_bp_addresses" )
4
5     LANGUAGE SQLSCRIPT
6     SQL SECURITY INVOKER
7     DEFAULT SCHEMA SAP_HANA_EPM_NEXT
8     READS SQL DATA AS
9
10 BEGIN
11 /*****
12  Write your procedure logic
13 *****/
14 ex_bp_addresses =
15     select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS", a."COMPANYNAME",
16           a."ADDRESSES.ADDRESSID" as "ADDRESSID", b."CITY", b."POSTALCODE", b."STREET"
17     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.BusinessPartner" as a
18        inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Addresses" as b
19        on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
20        where a."PARTNERROLE" = :im_partnerrole;
21
22 END;
```

© SAP

SQLScript Script Body

- Script body – developer writes the body of the script between the BEGIN and END statements.

```
SAP HANA SQLScript
1 PROCEDURE SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role" (
2     IN im_partnerrole NVARCHAR(3) DEFAULT '01',
3     OUT ex_bp_addresses SAP_HANA_EPM_NEXT."sap.hana.democontent.epmNext.data::EPM.Procedures.tt_bp_addresses" )
4     LANGUAGE SQLSCRIPT
5     SQL SECURITY INVOKER
6     DEFAULT SCHEMA SAP_HANA_EPM_NEXT
7     READS SQL DATA AS
8 BEGIN
9     /*****
10      Write your procedure logic
11      *****/
12 ex_bp_addresses =
13     select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS", a."COMPANYNAME",
14            a."ADDRESSES.ADDRESSID" as "ADDRESSID", b."CITY", b."POSTALCODE", b."STREET"
15     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.BusinessPartner" as a
16         inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Addresses" as b
17         on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
18         where a."PARTNERROLE" = :im_partnerrole;
19
20 END;
```

© SAP

Exercise

SQLScript

1. In the data sub package, create a data definition file **GlobalTypes.hdbdd** defining the following type in the schema <MATNR>.

```
namespace "8820038".honeycomb.honeycomb_data;
```

```
@Schema: '8820038'
```

```
context GlobalTypes {
```

```
  type tt_measures {
```

```
    ID: Integer;
```

```
    BEEKEEPERID: String(10);
```

```
    HONEYCOMBID:String(4);
```

```
    MEASUREDATE: LocalDate;
```

```
    MEASURETIME: LocalTime;
```

```
    WEIGHT: Decimal(5,2);
```

```
    WEIGHTUNIT: String(2);
```

```
    TEMPERATURE: Decimal(5,2);
```

```
    TEMPERATUREUNIT: String(2);
```

```
    BEEKEEPERNAME: String(10);
```

```
  };
```

```
};
```

Solution

SQLScript

The screenshot displays the SAP HANA Web-based Development Workbench: Editor interface. The top bar shows the SAP logo, the title 'SAP HANA Web-based Development Workbench: Editor', and the version 'v 1.120.14 | Help'. Below the title bar is a toolbar with icons for navigation and editing. The main area is divided into two panes. The left pane shows a project tree under 'Content' with the following structure:

- 8820038
 - honeycomb
 - honeycomb_connectivity
 - honeycomb_data
 - GlobalTypes.hdbdd (selected)
 - Honeycomb.hdbdd
 - Honeycomb.hdbti
 - HoneycombMeasures.csv
 - honeycomb_procedures
 - get_measures_by_beekeeper.hdbprocedure
 - honeycomb_service
 - honeycomb_measures.xsodata
 - honeycomb_ui
 - honeycomb_xs
 - public
 - sap
 - bc
 - hana

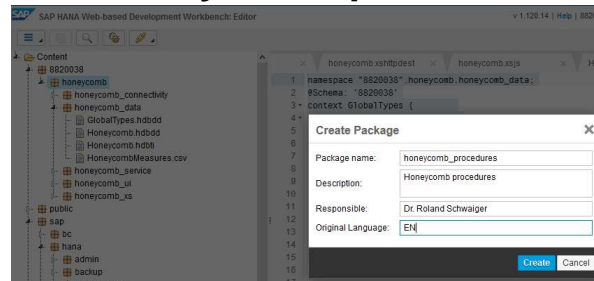
The right pane shows the code editor for 'GlobalTypes.hdbdd'. The code is as follows:

```
1 namespace "8820038".honeycomb.honeycomb_data;  
2 @Schema: '8820038'  
3 context GlobalTypes {  
4     type tt_measures {  
5         ID: Integer;  
6         BEEKEEPERID: String(10);  
7         HONEYCOMBID: String(4);  
8         MEASUREDATE: LocalDate;  
9         MEASURETIME: LocalTime;  
10        WEIGHT: Decimal(5,2);  
11        WEIGHTUNIT: String(2);  
12        TEMPERATURE: Decimal(5,2);  
13        TEMPERATUREUNIT: String(2);  
14        BEEKEEPERNAME: String(10);  
15    };  
16 };  
17
```


Solution

SQLScript

2. Create a folder named **honeycomb_procedures**.



3. Inside the folder, create an SQLScript procedure named **get_measures_by_beekeeper.hdbprocedure**, with the following behavior:

Input : variable named im_beekeeperid of type nvarchar(10)

Output: variable named ex_measures of type tt_measures (just defined)

Procedure:

ex_measures =

```
select ID, BEEKEEPERID, HONEYCOMBID, MEASUREDATE, MEASURETIME, WEIGHT,  
WEIGHTUNIT,
```

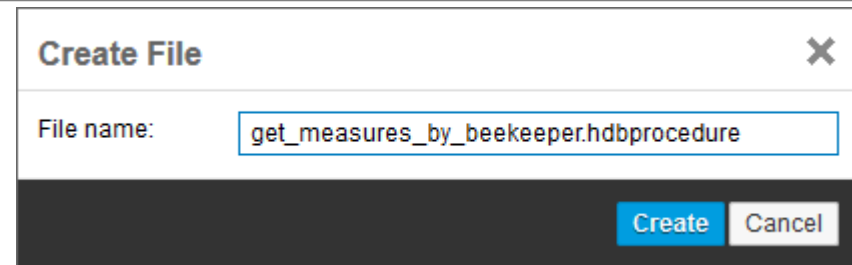
```
    TEMPERATURE, TEMPERATUREUNIT, BEEKEEPERNAME
```

```
from "8820038"."8820038.honeycomb.honeycomb_data::Honeycomb.Measures"
```

```
where "BEEKEEPERID" = :im_beekeeperid;
```

Solution

SQLScript

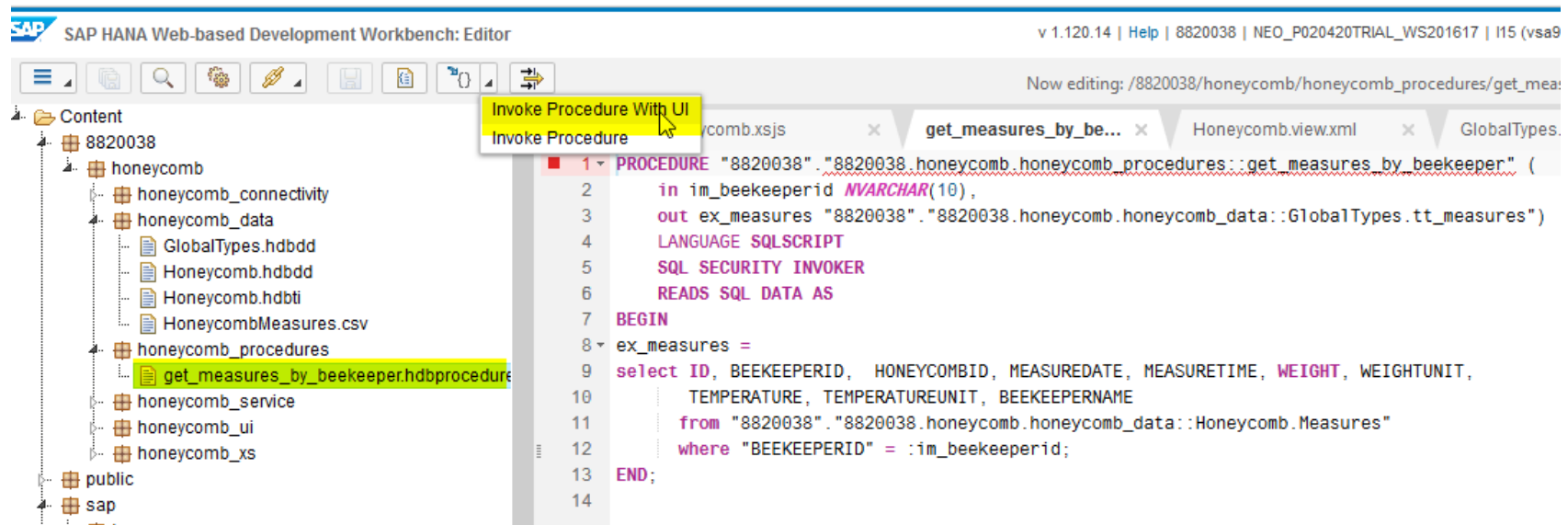


```
PROCEDURE
"8820038"."8820038.honeycomb.honeycomb_procedures::get_measures_by_beekeeper" (
    in im_beekeeperid NVARCHAR(10),
    out ex_measures
"8820038"."8820038.honeycomb.honeycomb_data::GlobalTypes.tt_measures")
    LANGUAGE SQLSCRIPT
    SQL SECURITY INVOKER
    READS SQL DATA AS
BEGIN
ex_measures =
select ID, BEEKEEPERID, HONEYCOMBID, MEASUREDATE, MEASURETIME, WEIGHT,
WEIGHTUNIT,
    TEMPERATURE, TEMPERATUREUNIT, BEEKEEPERNAME
from "8820038"."8820038.honeycomb.honeycomb_data::Honeycomb.Measures"
where "BEEKEEPERID" = :im_beekeeperid;
END;
```

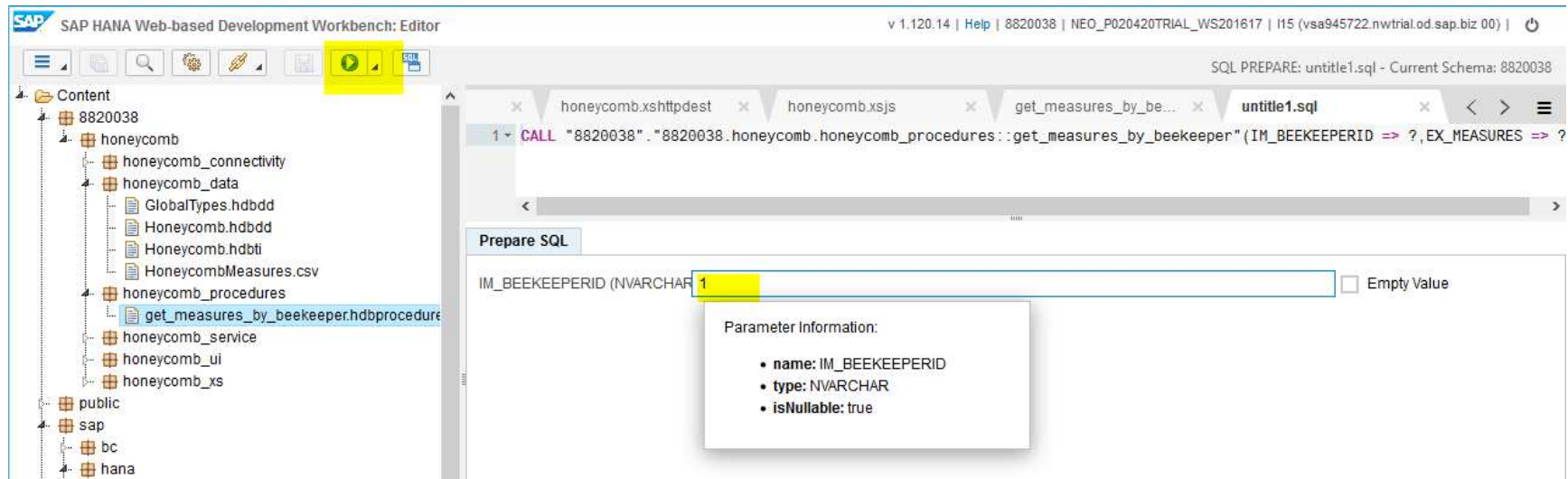
Solution

SQLScript

4. Run the procedure, setting the input parameter `im_beekeeperid` to 1.



Solution SQLScript



CALL
"8820038"."8820038.honeycomb.honeycomb_p
rocedures::get_measures_by_beekeeper"(IM_
BEEKEEPERID => ?,EX_MEASURES => ?)

Problem with missing permissions

- Try: GRANT SELECT ON SCHEMA 8820038 to _SYS_REPO WITH GRANT OPTION;
- Not working
- Created User 8820038D, provided the authorizations as described at the beginning of the course. Changed the code to schema 8820038D.
- Now
 - GRANT SELECT ON SCHEMA 8820038D to _SYS_REPO WITH GRANT OPTION; works.
 - The procedure call via UI works
 - XSODATA does not work
- Create .xaccess in 8820038D
 - Everything seems to work ... WHY?

SQLScript Features

Declarative Logic

- Allows the developer to declare the data selection via SELECT statements
 - Developer defines the what
 - Engine defines the how, and executes accordingly
 - Massive parallelization

```
19 ...../
20 - ex_bp_addresses =
21     select a."PARTNERID", a."PARTNERROLE", a."EMAILADDRESS",
22           a."COMPANYNAME", a."ADDRESSES.ADDRESSID" as "ADDRESSID",
23           b."CITY", b."POSTALCODE", b."STREET"
24     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::MD.BusinessPartner" as a
25         inner join "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::MD.Addresses" as b
26         on a."ADDRESSES.ADDRESSID" = b."ADDRESSID"
27         where a."PARTNERROLE" = :im_partnerrole;
28
29 END;
```

SQLScript Features

Imperative Logic

- Allows developer to control the flow of the logic within SQLScript.
 - Scalar variable manipulation
 - DDL/DML logic
 - WHILE loops
 - Branching logic based on some conditions, for example IF/ELSE
 - Executed exactly as scripted, procedurally

```
17 declare lv_category varchar(40) := null;
18 declare lv_discount decimal(15,2) := 0;
19
20 - lt_product = select PRODUCTID, CATEGORY, PRICE
21                from "sap.hana.democontent.epmNext.data::MD.Product
22                where PRODUCTID = :l_m_productid;
23
24 select CATEGORY into lv_category from :lt_product;
25
26 - if :lv_category = 'Notebooks' then
27     lv_discount := .20;
28 - elseif :lv_category = 'Handhelds' then
29     lv_discount := .25;
30 - elseif :lv_category = 'Flat screens' then
31     lv_discount := .30;
32 - elseif :lv_category like '%printers%' then
33     lv_discount := .30;
34 else
35     lv_discount := 0.00; -- No discount
36 end if;
37
38 - ex_product_sale_price =
39     select PRODUCTID, CATEGORY, PRICE,
40            PRICE * (PRICE * :lv_discount) as "SALEPRICE"
41            from :lt_product;
42 END;
```

© SAP

SQLScript Features

Arrays

- Allows the developer to define and construct arrays within SQLScript
 - Set elements
 - Return elements
 - Remove elements
 - Concatenate two arrays
 - Turn array into a table
 - Turn a column of a table into an array
 - Return cardinality of an array

```
11 declare productid nvarchar(10) ARRAY;  
12 declare category nvarchar(40) ARRAY;  
13 declare price decimal(15,2) ARRAY;  
14 declare saleprice decimal(15,2) ARRAY;  
15  
16 productid[1] := 'ProductA';  
17 productid[2] := 'ProductB';  
18 productid[3] := 'ProductC';  
19  
20 category[1] := 'CategoryA';  
21 category[2] := 'CategoryB';  
22 category[3] := 'CategoryC';  
23  
24 price[1] := 19.99;  
25 price[2] := 29.99;  
26 price[3] := 39.99;  
27  
28 saleprice[1] := 15.99;  
29 saleprice[2] := 25.99;  
30 saleprice[3] := 35.99;  
31  
32 ex_products = unnest(:productid, :category, :price, :saleprice)  
33 | | | | as ('PRODUCTID', 'CATEGORY', 'PRICE', 'SALEPRICE');  
34  
35 END;
```

© SAP

SQLScript Features

Dynamic Filter

- Allows developers to apply a dynamic WHERE clause to SELECT statements by using the APPLY_FILTER statement
 - Both database tables and intermediate table variables are supported

```
6 --DEFAULT_SCHEMA <default_schema_name>
7 READS SQL DATA AS
8 BEGIN
9 /-----
10 Write your procedure logic
11 -----/
12 ex_products = APPLY_FILTER("SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::MD_Products"
13                             :im_filter_string) ;
14
15 END
```

The screenshot shows a SQLScript editor window with a procedure call and its result. The procedure call is:

```
1 CALL "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.procedures::get_products_by_filter"(
2   IM_FILTER_STRING => 'CATEGORY like ''Soft%''',
3   EX_PRODUCTS => ?
4 );
```

A red arrow points to the filter string parameter. Below the code, the result table is displayed with 8 rows. The 'CATEGORY' column is highlighted with a red box.

	PRODUCTID	TYPECODE	CATEGORY	HISTORY.CR...
1	HT-1100	PR	Software	0000000033
2	HT-1101	PR	Software	0000000033
3	HT-1102	PR	Software	0000000033

SQLScript Features

Exception Handling 1/4

- Exception handling is a method for handling exception and completion conditions in an SQLScript procedures.
- The `DECLARE EXIT HANDLER` statement allows you to define exception handlers to process exception conditions in your procedures.
- You use the `DECLARE CONDITION` parameter to name exception conditions, and, optionally, their associated SQL state values.
- You can use `SIGNAL` or `RESIGNAL` with specified error code in user-defined error code range. A user-defined exception can be handled by the handler declared in the procedure. Also, it can be handled by the caller, which can be another procedure or client.

SQLScript Features

Exception Handling 2/4

- Declare exit handlers for generic SQL exceptions

```
23 DECLARE EXIT HANDLER FOR lcon_10001
24 BEGIN
25   ex_message := 'Error Code: ' || ::SQL_ERROR_CODE || ' ' || ::SQL_ERROR_MESSAGE;
26 END;
27
28 DECLARE EXIT HANDLER FOR SQL_EXCEPTION
29 BEGIN
30   ex_message := 'SQL Exception occurred!!';
31 END;
32
33 if :im_product = '' or :im_category = '' or :im_price = 0 then
34   SIGNAL lcon_10001 SET MESSAGE_TEXT = 'Input parameter can not be empty';
35 end if;
```

- Declare exit handlers for specific SQL error codes

```
21 DECLARE EXIT HANDLER FOR lcon_10001
22
23 DECLARE EXIT HANDLER FOR lcon_10001
24 BEGIN
25   ex_message := 'Error Code: ' || ::SQL_ERROR_CODE || ' ' || ::SQL_ERROR_MESSAGE;
26 END;
27
28 DECLARE EXIT HANDLER FOR SQL_ERROR_CODE 301
29 BEGIN
30   ex_message := 'SQL Exception occurred!!';
31 END;
32
33 if :im_product = '' or :im_category = '' or :im_price = 0 then
34   SIGNAL lcon_10001 SET MESSAGE_TEXT = 'Input parameter can not be empty';
35 end if;
```

© SAP

SQLScript Features

Exception Handling 3/4

- Signaling and catching user-defined conditions

```
19
20 DECLARE lcon_10001 CONDITION FOR SQL_ERROR_CODE 10001;
21 DECLARE EXIT HANDLER FOR lcon_10001 -- RESIGNAL
22 BEGIN
23     ex_message := 'Error Code: ' || ::SQL_ERROR_CODE || ' ' || ::SQL_ERROR_MESSAGE;
24 END;
25
26 IF :im_product = '' OR
27    :im_category = '' OR
28    :im_price = 0 THEN
29     SIGNAL lcon_10001 SET MESSAGE_TEXT = 'Input parameter can not be empty';
30 END IF;
31
```



The screenshot shows a SQLScript editor window with a tab titled 'untitled1.sql'. The code contains a CALL statement for a procedure named 'insert_product_data'. The parameters are: IM_PRODUCT => 'ProductA', IM_CATEGORY => '', IM_PRICE => 19.99, and EX_MESSAGE => ?. A red arrow points to the empty string value for IM_CATEGORY. Below the code, the 'Result1' window shows the output of the procedure call. It displays a table with one row: EX_MESSAGE | 1 | Error Code: 10001 input parameter can not be empty. A red arrow points to the error message text.

© SAP

SQLScript Features

Exception Handling 4/4

- RESIGNAL user defined exceptions to the caller
- RESIGNAL can also be executed explicitly within the body of the exit handler

```
20
21 DECLARE lcon_10001 CONDITION FOR SQL_ERROR_CODE 10001;
22 DECLARE EXIT HANDLER FOR lcon_10001 RESIGNAL; ←
23 --BEGIN
24 -- ex_message := 'Error Code: ' || :SQL_ERROR_CODE || ' ' || :SQL_ERROR_MESSAGE
25 --END;
26
27 IF :im_product = '' OR
28     :im_category = '' OR
29     :im_price = 0 THEN
30     SIGNAL lcon_10001 SET MESSAGE_TEXT = 'Input parameter can not be empty';
31 END IF;
32
33
```



The screenshot shows a SQL Editor window with two tabs: 'insert_product_data.h...' and 'untitled1.sql'. The 'untitled1.sql' tab is active and contains the following code:

```
1 CALL "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.procedures::insert_product_data"(
2   IM_PRODUCT => 'ProductA',
3   IM_CATEGORY => '',
4   IM_PRICE => 19.99,
5   EX_MESSAGE => ?
6 );
7
```

A red arrow points to the end of the call statement on line 7. Below the code, an error message is displayed:

```
[13:30:47] (SQL Editor) Could not execute "CALL "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.procedures::insert_product_data"( ...
Error: [dberror] 10001 - user-defined error: "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.procedures::insert_product_data": line 22
```

SQLScript

Autonomous Transaction

- Allows developer to create an isolated block of code which runs as an independent transaction.
- **BEGIN AUTONOMOUS TRANSACTION ... END** statement block.
- Committed statements inside autonomous transaction block will be persistent regardless of a rollback of the main transaction.
- For tables updated within the main procedure body, access to those tables is not allowed in the autonomous transaction block.
- Used commonly for logging tasks.

```
31 - DECLARE EXIT HANDLER FOR SQLEXCEPTION -- SQL_ERROR_CODE 301
32 BEGIN
33   ex_message := 'SQL Exception occurred!! ' || ::SQL_ERROR_CODE || ' ' || ::SQL_ERROR_MESSAGE;
34   BEGIN AUTONOMOUS TRANSACTION
35     INSERT INTO "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"
36     VALUES (:im_product,
37             (select IFNULL(MAX(logid), 0) + 1
38              from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"
39              where productid = :im_product),
40             now(), CURRENT_USER, ex_message );
41
42   COMMIT;
43   RESIGNAL;
44 END;
45 END;
```

© SAP

SQLScript Cursors

- Allows developers to iterate over a result set and perform row-based processing and calculations.

```
12 ..... /
13 declare v_new_price decimal(15,2);
14 declare CURSOR c_products FOR
15     SELECT PRODUCTID, CATEGORY, PRICE
16     from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data":MD.Products";
17
18 FOR cur_row as c_products DO
19
20     if :im_direction = 'INCREASE' then
21         v_new_price := cur_row.PRICE + (cur_row.PRICE * :im_rate);
22     elseif :im_direction = 'DECREASE' then
23         v_new_price := cur_row.PRICE - (cur_row.PRICE * :im_rate);
24     end if;
25 UPDATE "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data":MD.Products
26     SET PRICE = v_new_price where PRODUCTID = cur_row.PRODUCTID;
27
28 END FOR;
29
30 ex_products = select * from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data":MD.Products
31
32 END;
```

SQLScript

Commit/Rollback 1/2

User defined functions

- Supported in procedures only, not supported for scalar or table UDFs.
- **COMMIT** command commits the current transaction and all changes before the COMMIT command.
- **ROLLBACK** command rolls back the current transaction and undoes all changes since the last COMMIT.
- Transaction boundary is not tied to the procedure block, so, if there are nested procedures that contain COMMIT/ROLLBACK, then **all statements in the top-level procedure are affected.**
- If dynamic SQL was used in the past to execute COMMIT and ROLLBACK statements, we recommend that you replace all occurrences with the native command, because they are more secure.

© SAP

SQLScript Commit/Rollback 2/2

- The first and third INSERT statements affect the productLog table, but not the second, since it was rolled back.
- With the first COMMIT, the first transaction is written to persistence, and a new transaction is started.
- By triggering ROLLBACK, all changes in the second transaction are reverted, and a new transaction is started.

```
10 .....  
11 declare lv_message varchar(100);  
12  
13 lv_message := 'First log entry for Product ' || :m_product || ' inserted successfully';  
14 INSERT INTO "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
15 VALUES(:m_product,  
16         (select IFNULL(MAX(logid), 0) + 1  
17         from "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
18         where productid = :m_product),  
19         now(), CURRENT_USER, lv_message );  
20 COMMIT;  
21  
22 lv_message := 'Second log entry for Product ' || :m_product || ' inserted successfully';  
23 INSERT INTO "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
24 VALUES(:m_product,  
25         (select IFNULL(MAX(logid), 0) + 1  
26         from "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
27         where productid = :m_product),  
28         now(), CURRENT_USER, lv_message );  
29 ROLLBACK;  
30  
31 lv_message := 'Third log entry for Product ' || :m_product || ' inserted successfully';  
32 INSERT INTO "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
33 VALUES(:m_product,  
34         (select IFNULL(MAX(logid), 0) + 1  
35         from "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.data::MD.productLog"  
36         where productid = :m_product),  
37         now(), CURRENT_USER, lv_message );  
38 COMMIT;  
39
```

© SAP

SQLScript Functions

Table User defined functions (UDFs)

- Can have any number of input parameters.
- Returns exactly one table.
- Table operations are allowed within the body.
- Consumed in the FROM clause of a SELECT statement.

```

get_employees_by_n...
1 FUNCTION "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epnNext.functions::get_employees_by_name_filter"
2 (1m_lastNameFilter nvarchar(40))
3 RETURNS table ( EMPLOYEEID NVARCHAR(10),
4               "NAME.FIRST" NVARCHAR(40),
5               "NAME.LAST" NVARCHAR(40),
6               EMAILADDRESS NVARCHAR(255),
7               ADDRESSID NVARCHAR(10), CITY NVARCHAR(40),
8               POSTALCODE NVARCHAR(10), STREET NVARCHAR(60))
9
10 LANGUAGE SQLSCRIPT
11 SQL SECURITY INVOKER AS
12 BEGIN
13     /*
14     Write your function logic
15     */
16
17 RETURN
18     select a.EMPLOYEEID, a."NAME.FIRST",
19           a."NAME.LAST", a.EMAILADDRESS,
20           a."ADDRESSES.ADDRESSID" as ADDRESSID, b.CITY, b.POSTALCODE, b.STREET
21     from "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epnNext.data::MD_Employees" as a
22     inner join "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epnNext.data::MD_Addresses" as b
23       on a."ADDRESSES.ADDRESSID" = b.ADDRESSID
24        where contains("NAME.LAST", :1m_lastNameFilter, FUZZY(0.9));
25
26 END;
  
```

get_employees_by... | untitled2.sql

```

1 select * from "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epnNext.functions::get_employees_by_name_filter"('11');
  
```

Result

4 row(s)

	EMPLOYEEID	NAME.FIRST	NAME.LAST	EMAILADDRESS	ADDRESSID	
1	000000003	Franco	Fall	franco.fall@tele.info	1000000004	Sar
2	000000010	Michael	Miller	michael.miller@tele.inf	1000000011	Lor
3	000000018	Al	Wallace	al.wallace@tele.info	1000000019	Sar
4	000000027	Barbara	Ingalls	barbara.ingalls@tele.in	1000000028	Sar

© SAP

SQLScript Functions

Scalar UDFs

- Can have any number of input parameters.
- Can return multiple values.
- Input parameters cannot be table type.
- Consumed from the field list or the WHERE clause of the SELECT statement.
- Also callable via direct assignment(`x := my_scalar_func()`), multiple parameter assignments also supported.

```
apply_discount.hdbcs... x
1 FUNCTION "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.functions::apply_discount" (
2     im_price decimal(15,2),
3     im_discount decimal(15,2))
4     RETURNS ex_result decimal(15,2)
5     LANGUAGE SQLSCRIPT
6     SQL SECURITY INVOKER AS
7     BEGIN
8     /*****
9     Write your function logic
10    *****/
11     ex_result := :im_price - (:im_price * :im_discount);
12 END;
```

```
apply_discount.hdbcs... x  untitled3.sql x
1 - select PRODUCTID, CATEGORY, PRICE,
2     "SAP_HANA_EPH_NEXT"."sap.hana.democontent.epmNext.functions::apply_discount"(PRICE, 0.33 )
3     as "SALEPRICE" from "sap.hana.democontent.epmNext.data::MD.Products";
4
```

Result 1 x 106 row(s)

	PRODUCTID	CATEGORY	PRICE	SALEPRICE
1	HT-1000	Notebooks	956	640.52
2	HT-1001	Notebooks	1249	836.83
3	HT-1002	Notebooks	1570	1051.9
4	HT-1003	Notebooks	1650	

© SAP

SQLScript Triggers

- Special type of stored procedure that automatically execute when an event occurs in the database server.
- Triggers can be executed BEFORE or AFTER an event on a given table, such as INSERT, UPDATE, or DELETE.
- Management of objects can only be done via SQL Console. Support for storing the artifacts in the repository is coming in a future support package.

SERVER SIDE JAVASCRIPT

- Explain basic concepts of XSJS
- Explain how requests to XSJS programs are processed
- Access SAP HANA Database using XSJS
- Use XSJS Debugger
- Server Side JavaScript changes in XS Advanced
- Access server side XSJS scripts from SAPUI5 applications

http://help.sap.com/hana/SAP_HANA_XS_JavaScript_API_Reference_en/

XSJS

Server Side JavaScript (XSJS)

Lightweight procedural logic

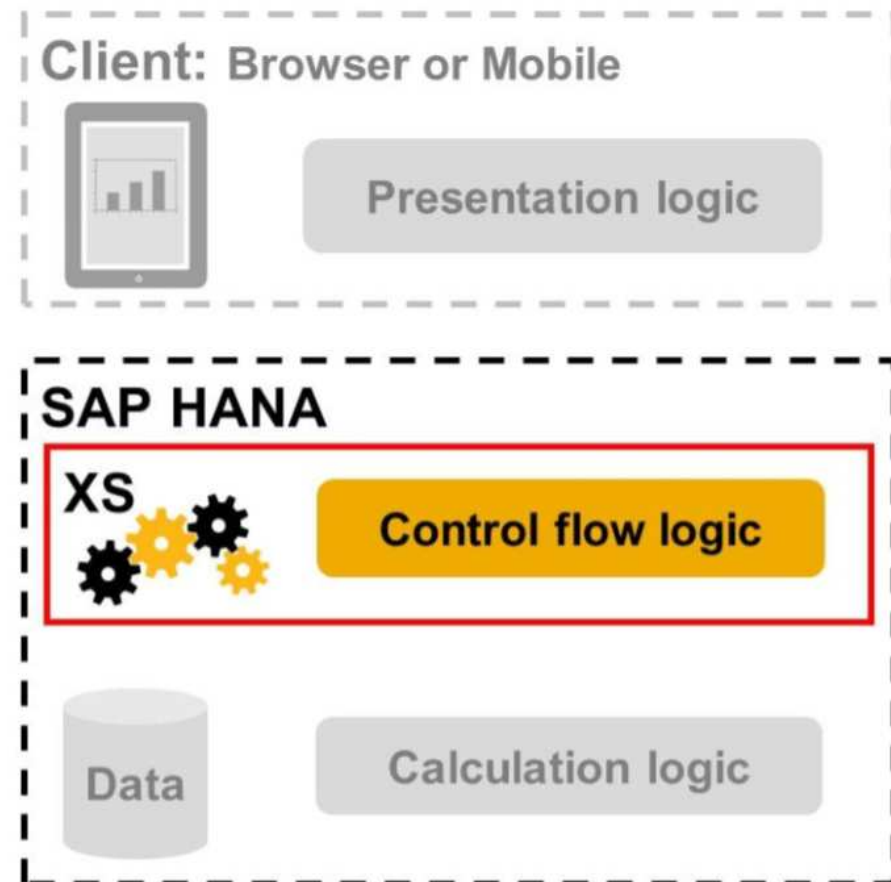
Reusable

One language:

- Client
- Server side

Widespread knowledge

Rapid development



© SAP

XSJS

Native application-specific code

Control data flow logic

Store XS JavaScript code in the repository

JavaScript editor/debugger



```
*retrieveData.xsjs X
$.response.contentType = "text/html";
var output = "Hello, World !<br><br>";

var conn = $.db.getConnection();
var pstmt = conn.prepareStatement( "select * from DUMMY" );
var rs = pstmt.executeQuery();

if (!rs.next()) {
    $.response.setBody( "Failed to retrieve data" );
    $.response.status = $.net.http.INTERNAL_SERVER_ERROR;
}
else {
    output = output + "This is the response from my SQL: "
    + rs.getString(1);
}
rs.close();
pstmt.close();
conn.close();

$.response.setBody(output);
```


XSJS

Interact with the SAP HANA XS runtime environment

Access SAP HANA database capabilities directly

Expose, update, insert, and delete data

Request API

- Control HTTP request/response

Database API

- Open DB connection, roll back changes in SAP HANA

Index

SAP HANA XS JavaScript API Reference

This API provides server-side JavaScript code running inside SAP HANA XS with access to request processing-related data and operations.

- [Request Processing API](#)
- [Database API](#)
- [HTTP Outbound API](#)
- [Trace API](#)
- [Job Scheduling API](#)
- [URL API](#)

Index

Classes

[Application](#)
[CallableStatement](#)
[Connection](#)
[ParameterMetaData](#)
[PreparedStatement](#)
[ResultSet](#)
[ResultSetMetaData](#)
[SQLException](#)
[Job](#)
[JobSchedules](#)
[Client](#)
[Destination](#)
[Request](#)
[Mail](#)
[Part](#)
[SMTPConnection](#)
[Session](#)
[Zip](#)
[Body](#)
[EntityList](#)
[TupelList](#)
[WebEntityRequest](#)
[WebEntityResponse](#)
[WebRequest](#)
[WebResponse](#)

XSJS

Reuse program elements

Reuse 3rd party or open source code

Perform repetitive tasks

- Handle forms
- Manage date/time strings
- Parse URLs

Import into other XSJS programs

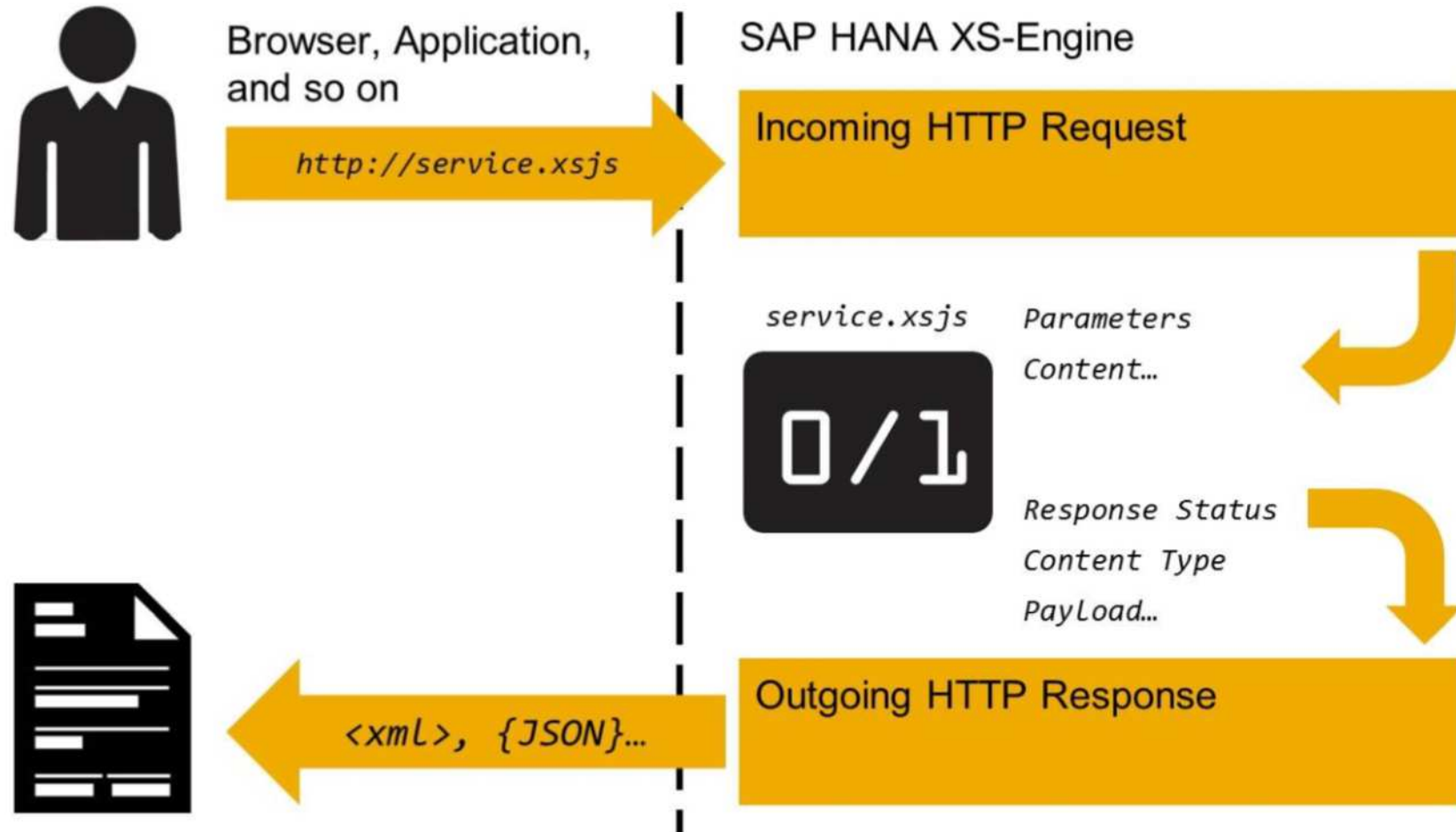
Call from other XSJS programs

```
// import math lib
$.import("sap.myapp.lib", "math");

// use math lib
var max_res = $.sap.myapp.lib.math.max(3, 7);
```

XSJS

Request Processing



XSJS

Request processing API

Namespace: web

Access via: *\$.web*

Important Classes

- **WebRequest**

Current Request: *\$.request*

- **WebResponse**

Current Response: *\$.response*

XSJS

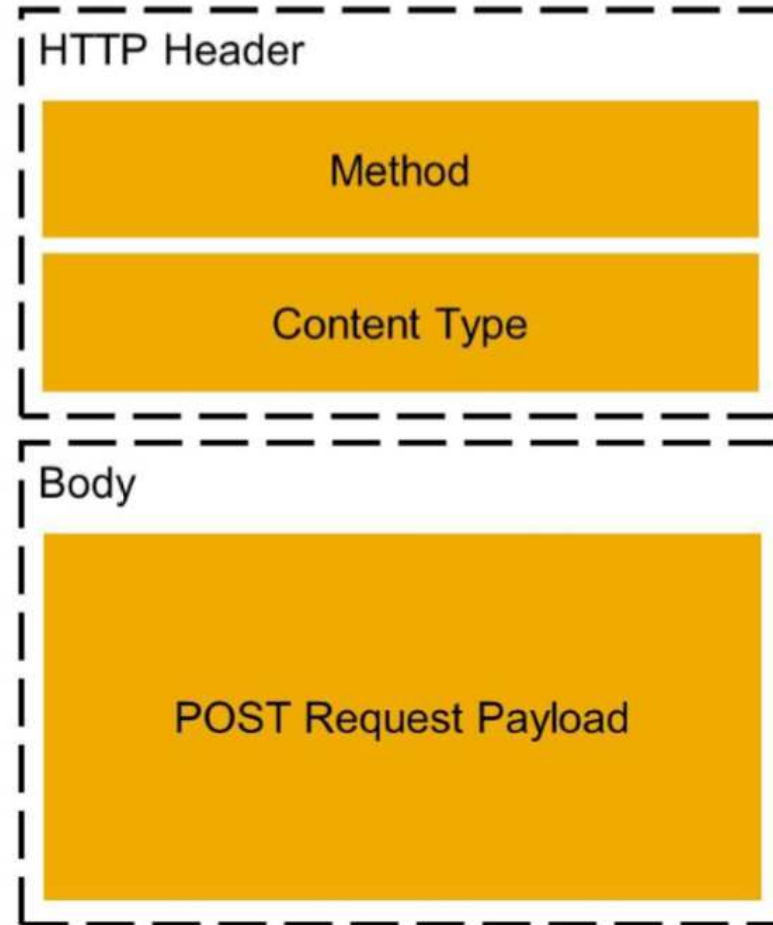
WebRequest

Represents the client HTTP request currently being processed.

Access via: `$.request`

Important members:

- `method`
- `contentType`
- `body`
- `parameters`



XSJS

WebResponse

Represents the HTTP response currently being populated.

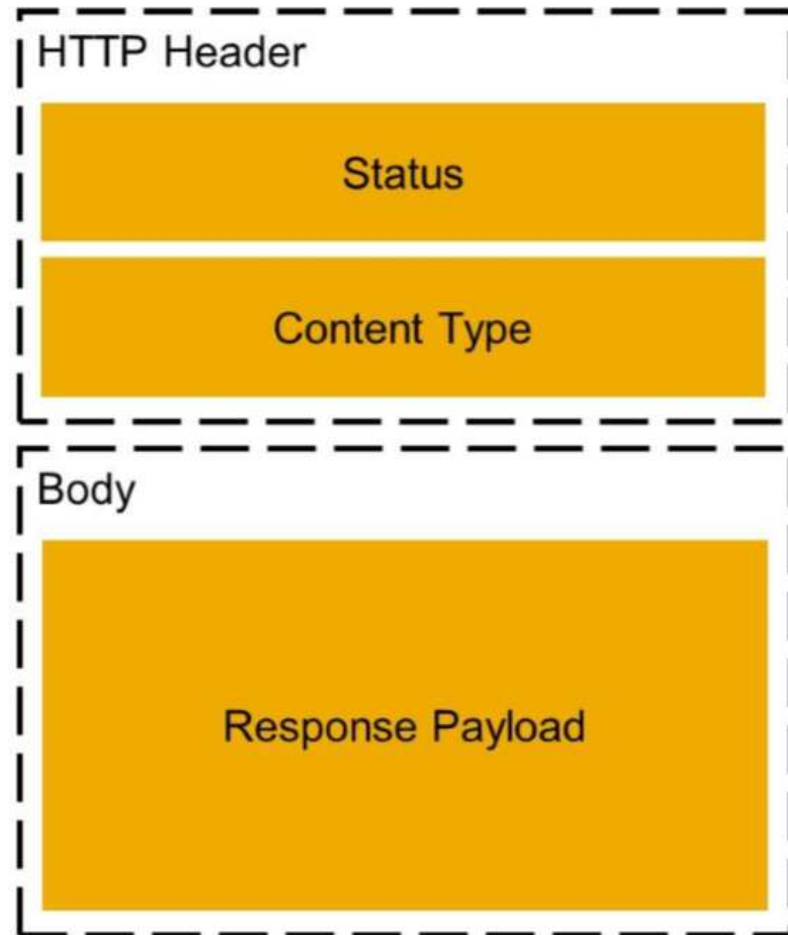
Access via: `$.response`

Important members:

- `contentType`
- `Status`

Important method:

- `setBody()`



XSJS

HTTP Status Codes

Represents the HTTP constants for status codes, inserted into the response.

Access via: `$.net.http`

Important properties:

- `OK`
- `BAD_REQUEST`
- `INTERNAL_SERVER_ERROR`
- `NOT_YET_IMPLEMENTED`

<static, readonly> **http**

HTTP constants for status codes

Properties:

Name	Type	Default	Description
CONTINUE	number	100	
SWITCH_PROTOCOL	number	101	
OK	number	200	
CREATED	number	201	
ACCEPTED	number	202	
NON_AUTHORITATIVE	number	203	
NO_CONTENT	number	204	
RESET_CONTENT	number	205	
PARTIAL_CONTENT	number	206	
MULTIPLE_CHOICES	number	300	
MOVED_PERMANENTLY	number	301	
FOUND	number	302	
SEE_OTHER	number	303	
NOT_MODIFIED	number	304	
USE_PROXY	number	305	
TEMPORARY_REDIRECT	number	307	
BAD_REQUEST	number	400	
UNAUTHORIZED	number	401	
PAYMENT_REQUIRED	number	402	
FORBIDDEN	number	403	
NOT_FOUND	number	404	

Exercise

XSSJS

- Write a request handler which sends „Hello World“

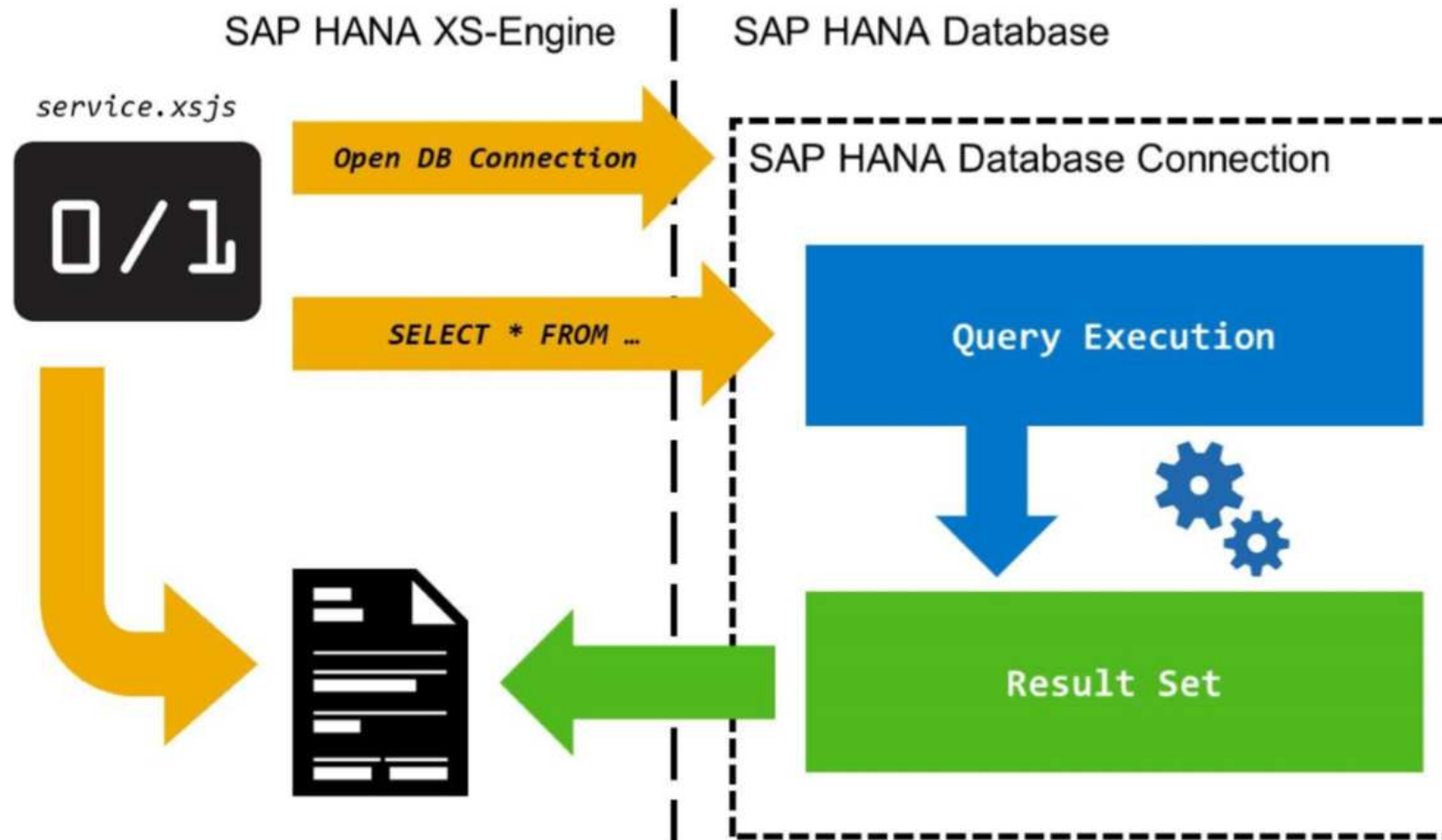
Solution

XSSJS

```
function getUsername(){  
    var username = $.session.getUsername();  
    return username;  
}  
  
var result = "Hello World from User " +  
getUsername();  
$.response.setBody(result);
```

XSJS

Accessing the HANA DB



XSJS

DB API

This namespace provides means for seamless SAP HANA database access.

Namespace: `hdb`

Access via: `$.hdb`

Important Classes

- `Connection`
- `ResultSet`
- `ProcedureResult`

Important Methods

- `getConnection()`

XSJS

Establish connection

Represents the SAP HANA database connection.

Access via: `$.hdb.getConnection()`

Important Methods

- `executeQuery(query [,arguments])`
- `executeUpdate(statement [,arguments])`
- `LoadProcedure(schema, procedure)`
- `commit()`
- `rollback()`
- `close()`

XSJS

Execute Statement

SQL Queries to the database are executed on the connection object:

```
connection.executeUpdate(query, arguments);

var oConnection= $.hdb.getConnection();

var sQuery = 'SELECT * FROM "workshop.exercises.g<group>.data::User.Details" ' +
            'LIMIT ?';
var oResultSet = oConnection.executeQuery(sQuery, vLimit);
```

The result set can be accessed like any JavaScript via the *ResultSet* object.

```
for(var i = 0; i < oResultSet.length; i++){

    sBody +=    oResultSet[i].PERS_NO +    "," +
                oResultSet[i].FIRSTNAME +  "," +
                oResultSet[i].LASTNAME +   "," +
                oResultSet[i].E_MAIL +     "\n";
}
```

It can also be used easily by a JavaScript frontend by sending it as a JSON string, using the method `JSON.stringify()`.

XSJS

Call a procedure

The Database API provides mechanisms to call a database procedure in an XSJS program like a JavaScript procedure.

The database procedure is loaded via the connection object:

```
connection.LoadProcedure(schema, procedure);  
  
var oConnection = $.hdb.getConnection();  
  
var fnCreateUser = oConnection.loadProcedure("HANA_WORKSHOP_SOL",  
                                             "workshop.exercises.g<group>.procedures::createUser");
```

Subsequently, the procedure is called using its JavaScript representation. The function has the same parameters as the database procedure.

```
var oResult = fnIncreasePrice(1.5);
```

Exercise

XSJS

- Accessing the database
- `downloadMeasures.xsjs`

Solution

XSJS

```
function downloadMeasures() {
    var vLimit = $.request.parameters.get("limit");
    if (isNaN(parseInt(vLimit, 10)) || vLimit < 0) {
        $.response.status = $.net.http.BAD_REQUEST;
        $.response.contentType = "text/json";
        var oResponse = {
            message: "The Parameter 'limit' must be set!"
        };
        $.response.setBody(JSON.stringify(oResponse));
        return;
    }
    var oConnection = $.hdb.getConnection();
    var sQuery = "SELECT * FROM \"8820038D.honeycomb.honeycomb_data::honeycomb8820038.Measures\" + " LIMIT ?";
    var oResultSet = oConnection.executeQuery(sQuery, vLimit);
    var sBody = "";
    for (var i = 0; i < oResultSet.length; i++) {
        sBody += oResultSet[i].ID + "," + oResultSet[i].HONEYCOMBID + "," + oResultSet[i].MEASUREDATE + "," +
oResultSet[i].WEIGHT + "\n";
    }
    $.response.status = $.net.http.OK;
    $.response.contentType = "application/csv";
    $.response.headers.set("Content-Disposition", "attachment;filename=list.csv");
    $.response.setBody(sBody);
    oConnection.close();
}
downloadMeasures();
```


Exercise

XSJS

- Call procedure
- `createMeasure.xsjs`

Solution

XSJS

- Create a procedure **create_measure.hdbprocedure**

```
PROCEDURE
```

```
"8820038D"."8820038D.honeycomb.honeycomb_procedures::create_measure" (
```

```
  in im_measures
```

```
"8820038D"."8820038D.honeycomb.honeycomb_data::GlobalTypes.tt_measures" )
```

```
  LANGUAGE SQLSCRIPT
```

```
  SQL SECURITY INVOKER
```

```
  READS SQL DATA AS
```

```
BEGIN
```

```
  // here goes your code
```

```
END;
```

Solution

XSJS

- Create a XSJS service **createMeasure.xsjs**

```
function createMeasure() {
    $.response.contentType = "text/json";
    var oConnection = $.hdb.getConnection();
    var fnCreateMeasure = oConnection.loadProcedure("8820038D", "8820038D.honeycomb.honeycomb_procedures::create_measure");
    var oNewMeasure = [{
        ID: "00000",
        HONEYCOMBID: $.request.parameters.get("honeycombID"),
        MEASUREDATE: $.request.parameters.get("measureDate"),
        WEIGHT: $.request.parameters.get("measureWeight")
    }];
    var oResult;
    try {
        oResult = fnCreateMeasure(oNewMeasure);
    } catch (oException) {
        $.response.status = $.net.http.INTERNAL_SERVER_ERROR;
        $.response.setBody(JSON.stringify({
            message: oException.message
        }));
        return;
    }
    if (oResult.ERROR[0] !== undefined) {
        $.response.status = parseInt(oResult.ERROR[0].HTTP_STATUS_CODE, 10);
        $.response.setBody(JSON.stringify({
            message: oResult.ERROR[0].ERROR_MESSAGE
        }));
    } else {
        $.response.status = $.net.http.OK;
        $.response.setBody(JSON.stringify({
            message: "The procedure call was successful..."
        }));
    }
    oConnection.commit();
    oConnection.close();
}
createMeasure();
```

Solution

XSJS

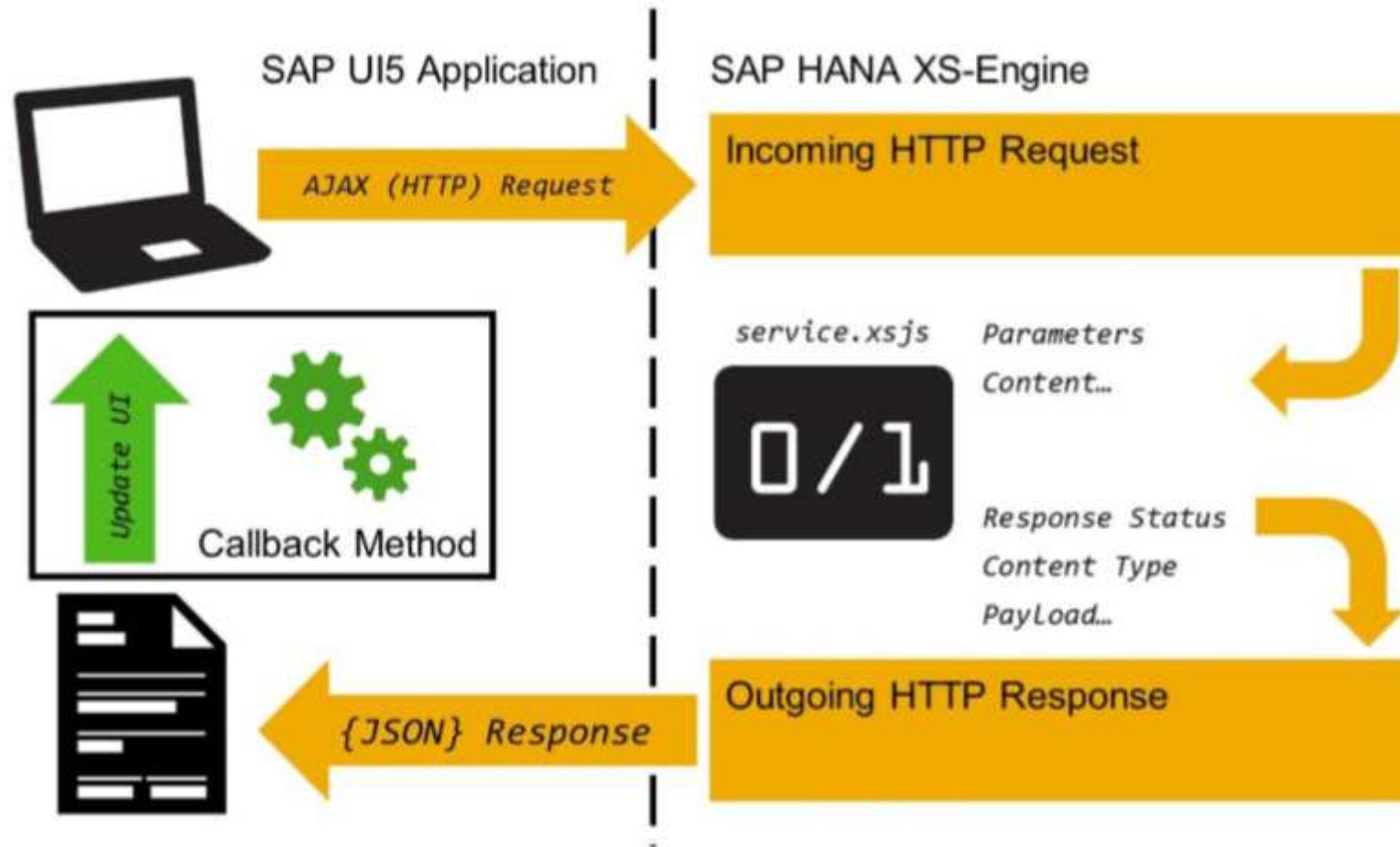
Test the URL, which will look like the following one:

`https://<domain>:<port>/<path>/`

`createMeasure.xsjs?`

`id=1&measuredate=2017-02-03&weight=34`

XSJS - UI5



XSJS – UI5

jQuery

Namespace: jQuery

Access via: `jQuery.ajax(settings)`

Important settings

- **url**
- **method**
GET, POST, PUT
- **data**
Request payload
- **dataType**
xml, html, script, json, text
- **context**
Callback method context
- **success**
Callback function if success
- **error**
Callback function if error

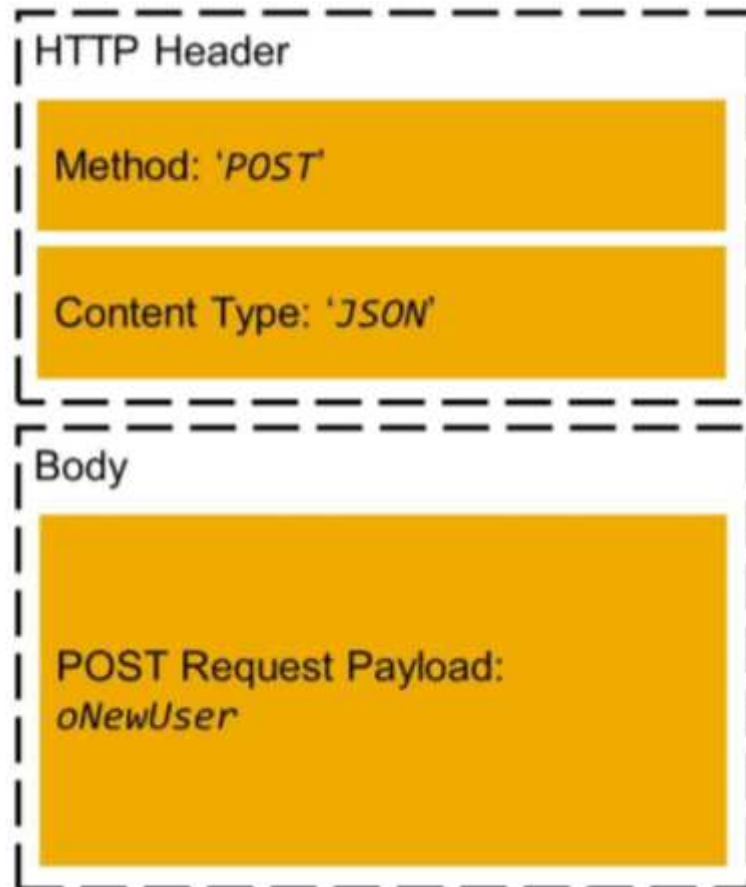
<http://api.jquery.com/jquery.ajax/>

XSJS – UI5

AJAX request

A typical AJAX request, which creates a POST HTTP Request to the service CreateUser.xsjs, sending the JSON object oNewUser.

```
jQuery.ajax({  
  
    url: "../..services/CreateUser.xsjs",  
    method: 'POST',  
    data: oNewUser,  
    dataType: 'JSON',  
    context: this,  
  
    success: function(oResponse, sStatus, oXHR){  
        // Process the server response  
    },  
    error: function(oXHR, sStatus, sError){  
        // Signal an error  
    }  
});
```



XSJS – UI5

AJAX response

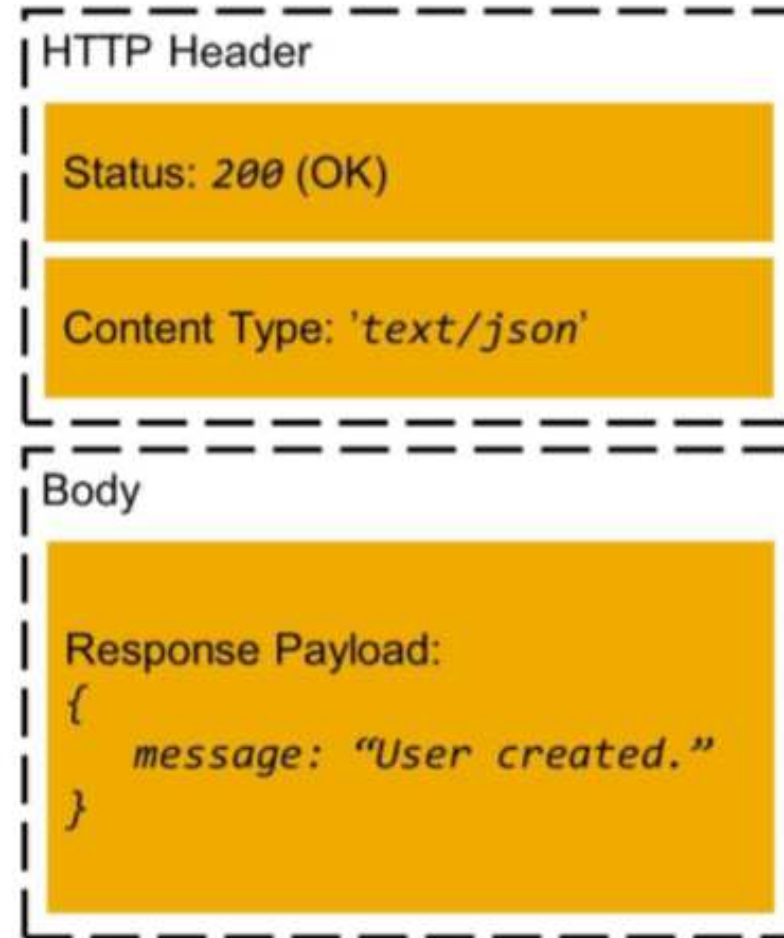
As the AJAX request is done asynchronously, the servers response is handled with the callback methods.

Depending on the response status (you set them, for example, in XSJS programs), the success or the error function is executed.

- success: status = 2xx / 304
- error: everything else

The response content type should correspond to the data type you specified to expect in the call. E.g. JSON objects are parsed by jQuery and can be accessed directly in the callback method.

The callback methods are executed with an own context. If you need access to variables of another context you need to set the context setting of the AJAX call accordingly.



CONNECTIVITY 4 HANA XS

- Make XS destinations for consuming HTTP Internet services
- Use existing services (Google and Honeycomb.Measures)

Exercise

Connectivity

- Create package honeycomb_connectivity

Connectivity

Grant access to package 4 user

- Logon as user SYSTEM in order to change the user <MATNR>

The screenshot displays the SAP HANA Security Workbench interface. On the left, the 'Users' tree shows the user '8820038' selected. The main area shows the user configuration for '8820038', including authentication settings (Password, Kerberos, SAML, X509) and validity dates. The 'Package Privileges' tab is active, showing a table of granted packages and a list of privileges for the selected package '8820038.honeycomb.honeycomb_connectivity'.

Package Name	Grantor
8820038.honeycomb.honeycomb_connectivity	8820038

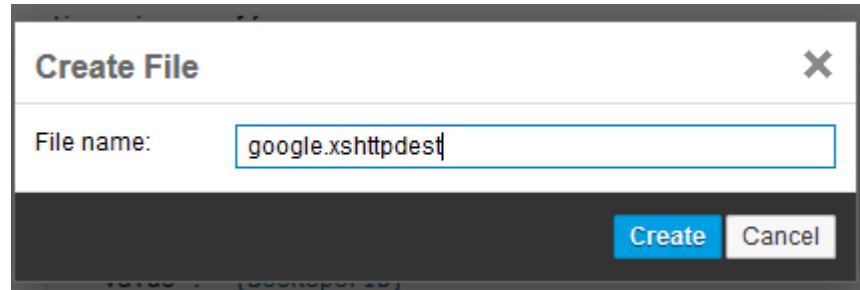
Privilege	Grantable to Others
<input checked="" type="checkbox"/> ALL	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.READ	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.EDIT_NATIVE_OBJECTS	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.ACTIVATE_NATIVE_OBJECTS	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.MAINTAIN_NATIVE_PACKAGES	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.EDIT_IMPORTED_OBJECTS	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.ACTIVATE_IMPORTED_OBJECTS	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input checked="" type="checkbox"/> REPO.MAINTAIN_IMPORTED_PACKAGES	<input type="radio"/> Yes <input checked="" type="radio"/> No

Connectivity

XS Destination File

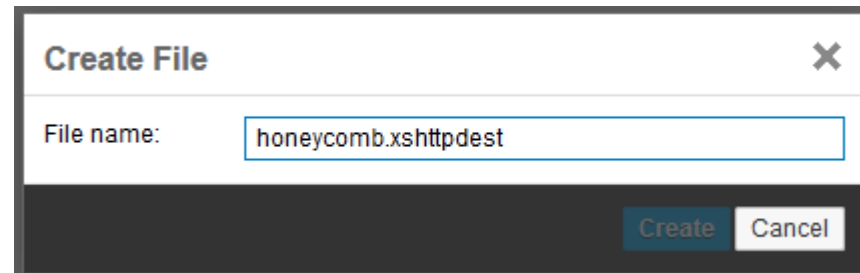
Google

```
host = "maps.googleapis.com";  
port = 80;  
pathPrefix = "/maps/api/distancematrix/json";  
useProxy = true;  
proxyHost = "proxy-trial";  
proxyPort = 8080;  
authType = none;  
useSSL = false;  
timeout = 30000;
```



Honeycomb

```
host = "www.niun.org";  
port = 80;  
pathPrefix = "/beelog";  
useProxy = false;  
authType = none;  
useSSL = false;  
timeout = 30000;
```



Connectivity

XSJS File

honeycomb_connectivity/honeycomb.xsjs

```
var destination_package = "8820038.honeycomb.honeycomb_connectivity";
var destination_name = "honeycomb";

try {
    var dest = $.net.http.readDestination(destination_package, destination_name);

    var client = new $.net.http.Client();
    var req = new $.web.WebRequest($.net.http.GET, "/current.php");
    client.request(req, dest);
    var response = client.getResponse();

    $.response.contentType = "application/json";
    $.response.setBody(response.body.asString());
    $.response.status = $.net.http.OK;
} catch (e) {
    $.response.contentType = "text/plain";
    $.response.setBody(e.message);
}
```

Connectivity

XSJS File

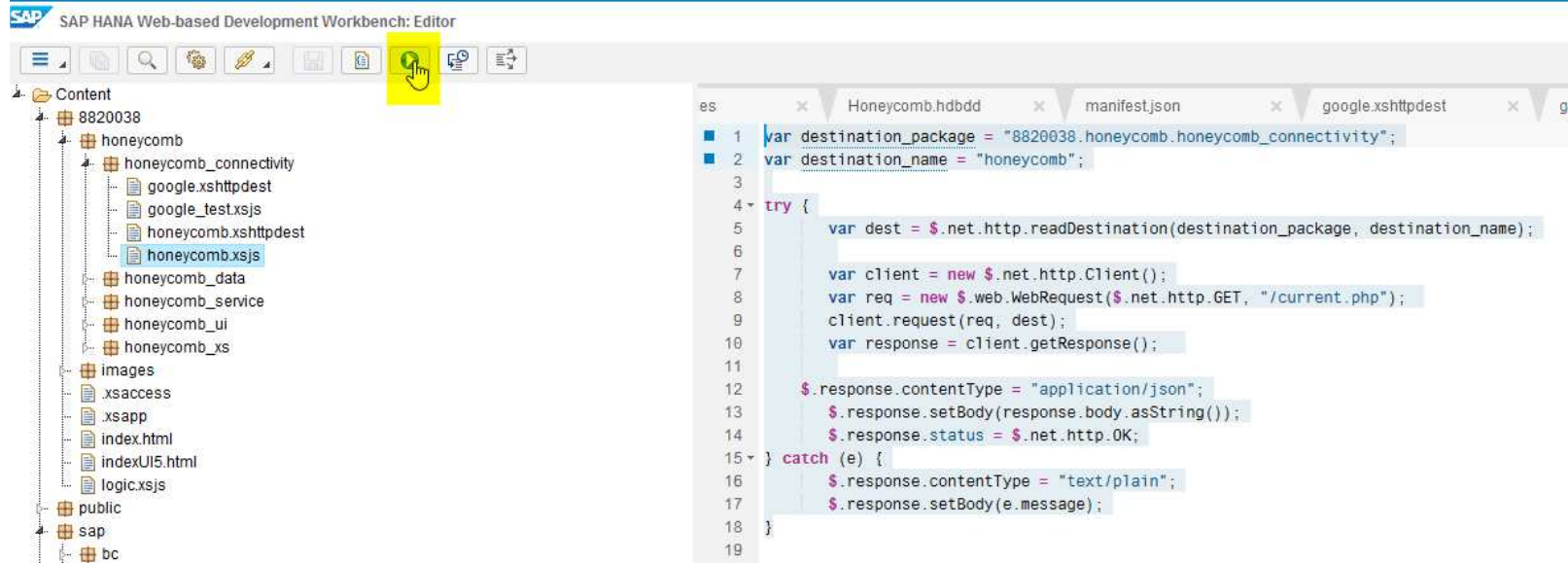
honeycomb_connectivity/google_test.xsjs

```
var destination_package = "8820038.honeycomb.honeycomb_connectivity";
var destination_name = "google";

try {
    var dest = $.net.http.readDestination(destination_package, destination_name);
    var client = new $.net.http.Client();
    var req = new $.web.WebRequest($.net.http.GET,
    "?origins=Frankfurt&destinations=Cologne&mode=driving&language=en-US&sensor=false");
    client.request(req, dest);
    var response = client.getResponse();

    $.response.contentType = "application/json";
    $.response.setBody(response.body.asString());
    $.response.status = $.net.http.OK;
} catch (e) {
    $.response.contentType = "text/plain";
    $.response.setBody(e.message);
}
```

Connectivity Test XSJS File



```
{
    "Weights": {
        "BeekeeperName": "inkerlois",
        "BeekeeperID": "inkerlois",
        "HoneycombID": "1",
        "MeasureDate": "2017-01-05",
        "MeasureTime": "15:23:24",
        "Weight": "24.10",
        "Temperature": "0.10"
    }
}
```

Connectivity

Test XSJS File

Google

```
{
  "destination_addresses": ["Cologne, Germany"],
  "origin_addresses": ["Frankfurt, Germany"],
  "rows": [{
    "elements": [{
      "distance": {
        "text": "190 km",
        "value": 190447
      },
      "duration": {
        "text": "2 hours 1 min",
        "value": 7283
      },
      "status": "OK"
    }
  ]
},
  "status": "OK"
}
```


Abschluss LV

- Der **Abschluss** der LV findet im Rahmen des LV Blocks statt. Dieser besteht aus:
 - **Implementierung** von SAP Entwicklungs Objekten im Rahmen der LV
 - Abschlussprüfung
 - **Theoretische** Prüfung mit zehn Fragen aus den Inhalten der LV
 - **Entwicklung** von SAP Entwicklungs Objekten

Literaturliste



Create XS Application <https://help.hana.ondemand.com/help/frameset.htm?4959458552574c77b62fe27b0eb363ef.html>

SAP HANA Multitenant Database Containers (MDC) <https://blogs.sap.com/2016/01/13/sap-hana-multitenant-database-containers-mdc-scenarios-now-on-trial-landscape/>

Install SHINE <https://blogs.sap.com/2016/05/20/shine-for-sap-hana-extended-services-classic-model-xsc-as-self-service/>

VizChart <https://archive.sap.com/discussions/thread/3745279>

HAMA SBS 11 <https://blogs.sap.com/2016/03/31/sap-hana-sps-11-new-developer-features-sap-web-ide-for-sap-hana/>

DataSource <http://www.niun.org/beelog/current.php>
<http://www.niun.org/beelog/all.php>

CDS https://help.sap.com/saphelp_hanaplatform/helpdata/en/b5/23afd66f5a40469573d9c47d7af831/content.htm?frameset=/en/ad/036c56b5e545ae8b31ece0ab95379f/frameset.htm¤t_toc=/en/8f/796f93e6c3419c9b8029197aa61874/plain.htm&node_id=50&show_children=false

Connectivity <https://help.hana.ondemand.com/help/frameset.htm?2eb13e1217a74561af5950e006fd1e36.html>

Debugging <https://help.hana.ondemand.com/help/frameset.htm?1beaa7aaadc743568c8144066d005dab.html>

JobSchedule https://help.sap.com/saphelp_hanaplatform/helpdata/en/62/15446213334d9fa96c662c18fb66f7/content.htm

Arduino to HANA mit IoT <https://create.arduino.cc/projecthub/derapados/arduino-iot-with-sap-hana-cloud-platform-031988>

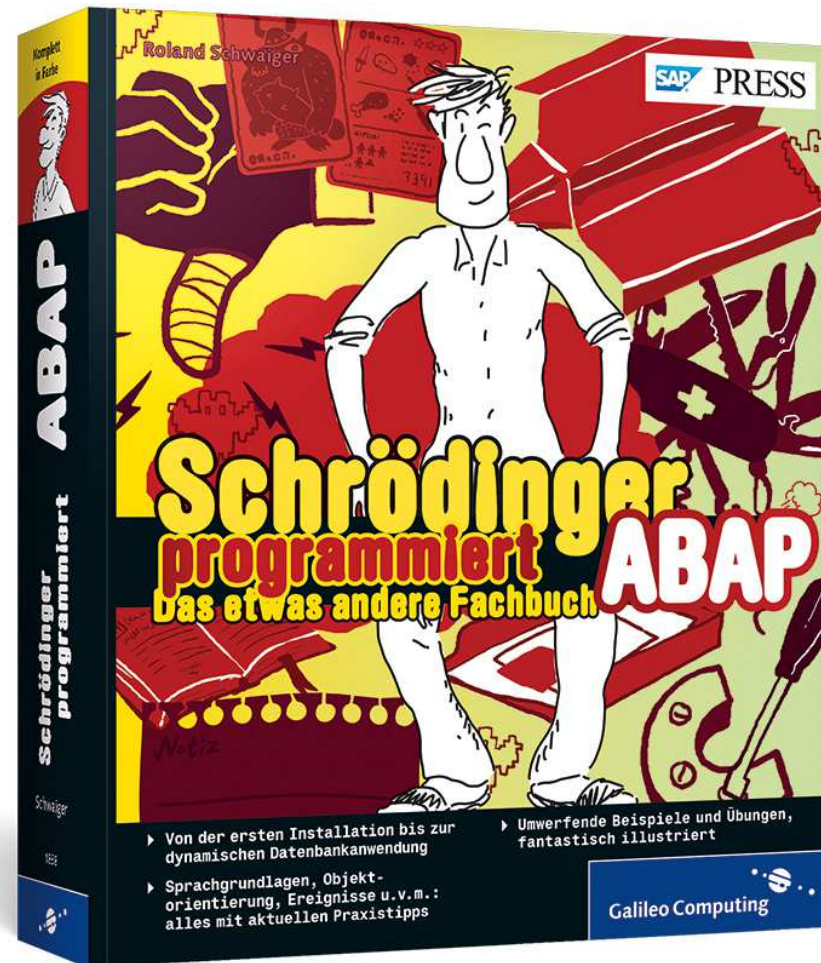
IoT FAQ HANA <https://help.hana.ondemand.com/iot/frameset.htm?a012a3788b6e498a8971dd27c97ce6bf.html>

sdn.sap.com
help.sap.com
www.facet.at

Glossary

Abbreviation	Explanation
MDC	SAP HANA Multitenant Database Containers

Empfehlenswert



APPENDIX

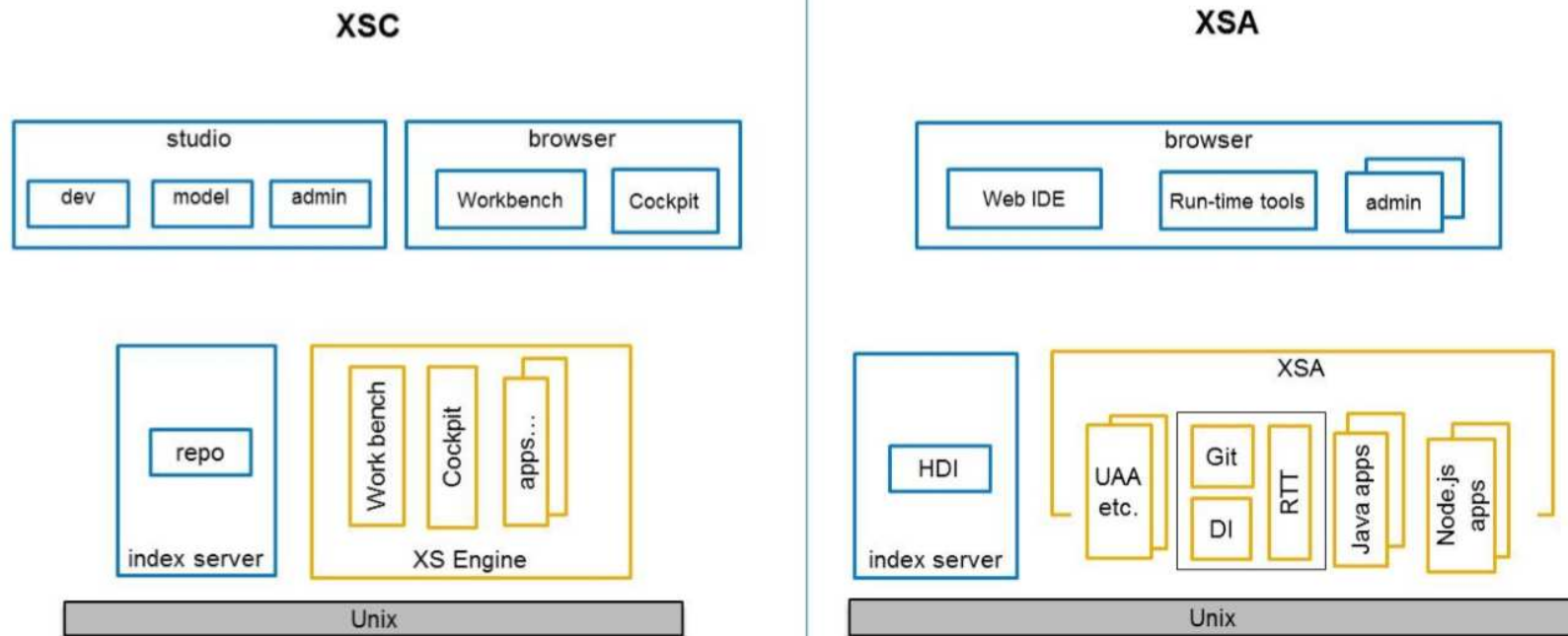
All the things that did not make it into the main release

XSA

Because XSA (SPS 11) is not available in the HANA Trial at the moment of writing.

Introduction to HANA

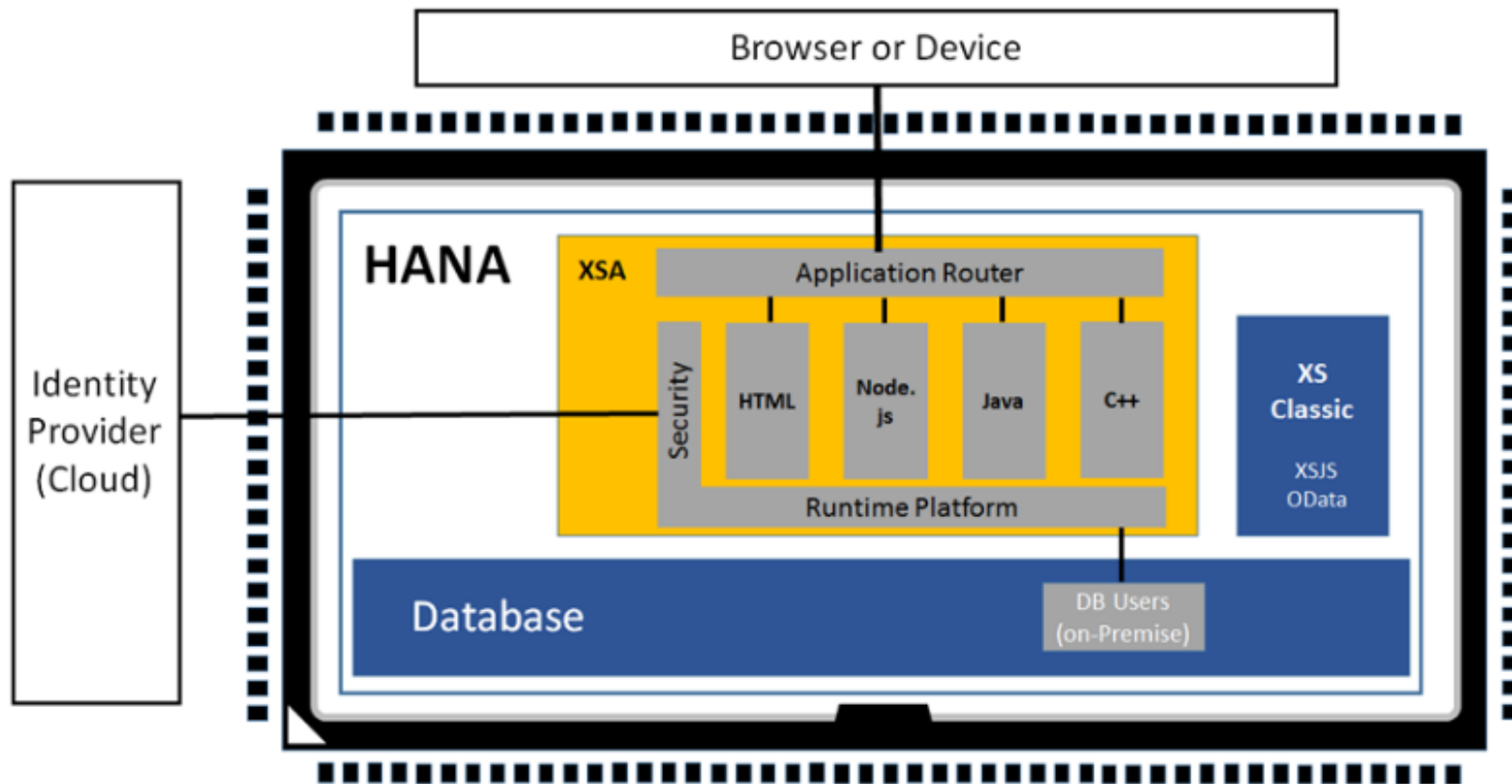
XS Classic vs. XS Advanced



© SAP

Introduction to HANA

XSA - Arch



XSA - Arch

- SAP HANA XS advanced (XSA) is a completely re-engineered application server for native development of applications in SAP HANA environment which is available with **SP11**. It brings dramatic improvements in terms of architecture with microservices.
- XSA has an **application router** to route browser requests to relevant applications. XSA supports **multiple language** runtimes like Node.js, Java, C++ and so on, as well as a runtime platform. One application can use one or more of these runtimes by combining them as **microservices** running side by side in one application.
- In SAP HANA XS, if one service running on HANA server fails, it impacts the dependent processes. In XS Advanced on the other hand, these language runtimes run separately for every app by using a copy of the runtime version or SDKs as required. They also run independently of one another, which reduces the impact of a failed service and high scalability.
- When it comes to security, no HANA DB User is needed for accessing the HANA applications. SAP HANA **XSA** uses an **identity provider**, which provides user authentication info using SAP Cloud Identity Service or SAP ID Service (IDS) as in SCN, SAP.com or SAP store.
- **Note:** SAP HANA XS and SAP HANA XSA co-exist in HANA, but are completely separated and work independently of one another. SAP HANA XSA supersedes SAP HANA XS in HANA native development.

XSC vs XSA

XSC

- + Familiar development environment (HANA Studio, Web IDE and the HANA repository)
- + System in a Box
- + Supports JavaScript as application run-time
- + Direct access to database objects
- + Integrated transportation management
- Single source code repository per system
- Limited scalability due to tight coupling with DB
- Requires DB authorization management to be used
- Requires additional systems for production emergency fix delivery

XSA

- + Application layer separable from DB
- + Support of different run-time containers, e.g. Node.js (JavaScript), Java, C++
- + Highly scalable due to container concept
- + Advanced Git/Gerrit Source Code Management
- + Advanced resource management per container
- Additional HANA software components to be installed
- External Source Code Repository to be set up
- Additional development knowledge necessary to create database objects
- ❖ Web-based tools for developers and administrators (**no Eclipse/HANA Studio support!**)
- ❖ Development: SAP Web IDE + server-side Development Infrastructure (DI)
- ❖ HANA container isolation and activation: HDI
- ❖ Deploy MTA archives

XSC vs XSA

XS Classic

- HANA repository contains design-time objects
- Objects are organized in repository packages
- Database run-time objects are created upon activation of design-time objects
- Objects can be activated individually
- User `_SYS_REPO` owns all run-time objects

XS Advanced

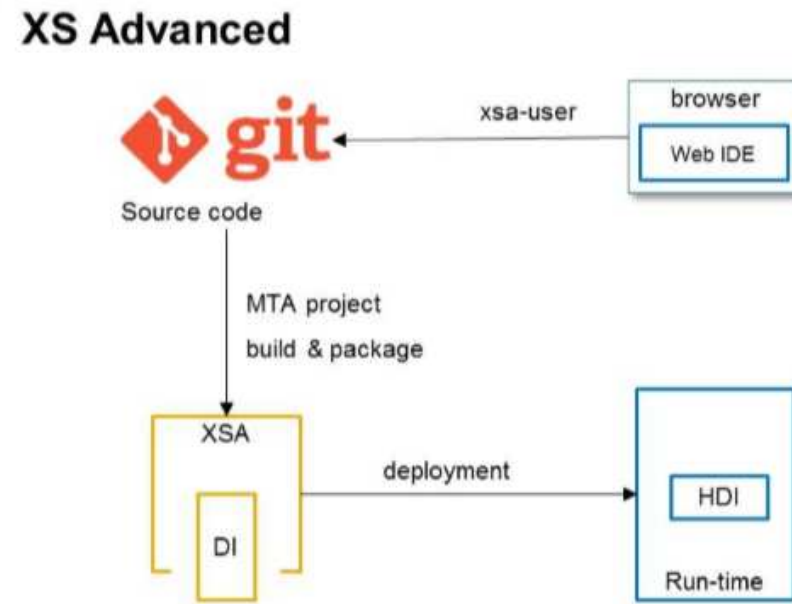
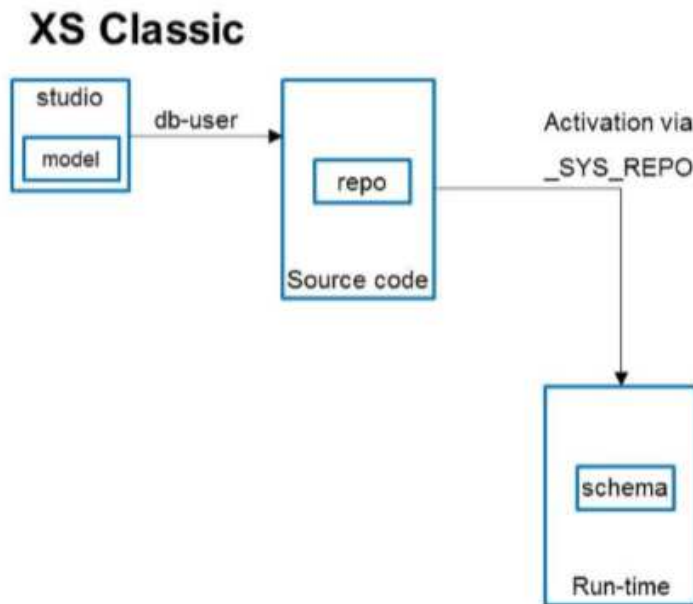
- External GIT repository contains design-time objects
- Objects are organized in MTA Projects
- Projects have to be built by the XSA infrastructure
- Database run-time objects are created upon deployment of the built HDI Container
- A generated `...#OO` user owns the run-time objects for a specific HDI container

Multi Target Application (MTA)

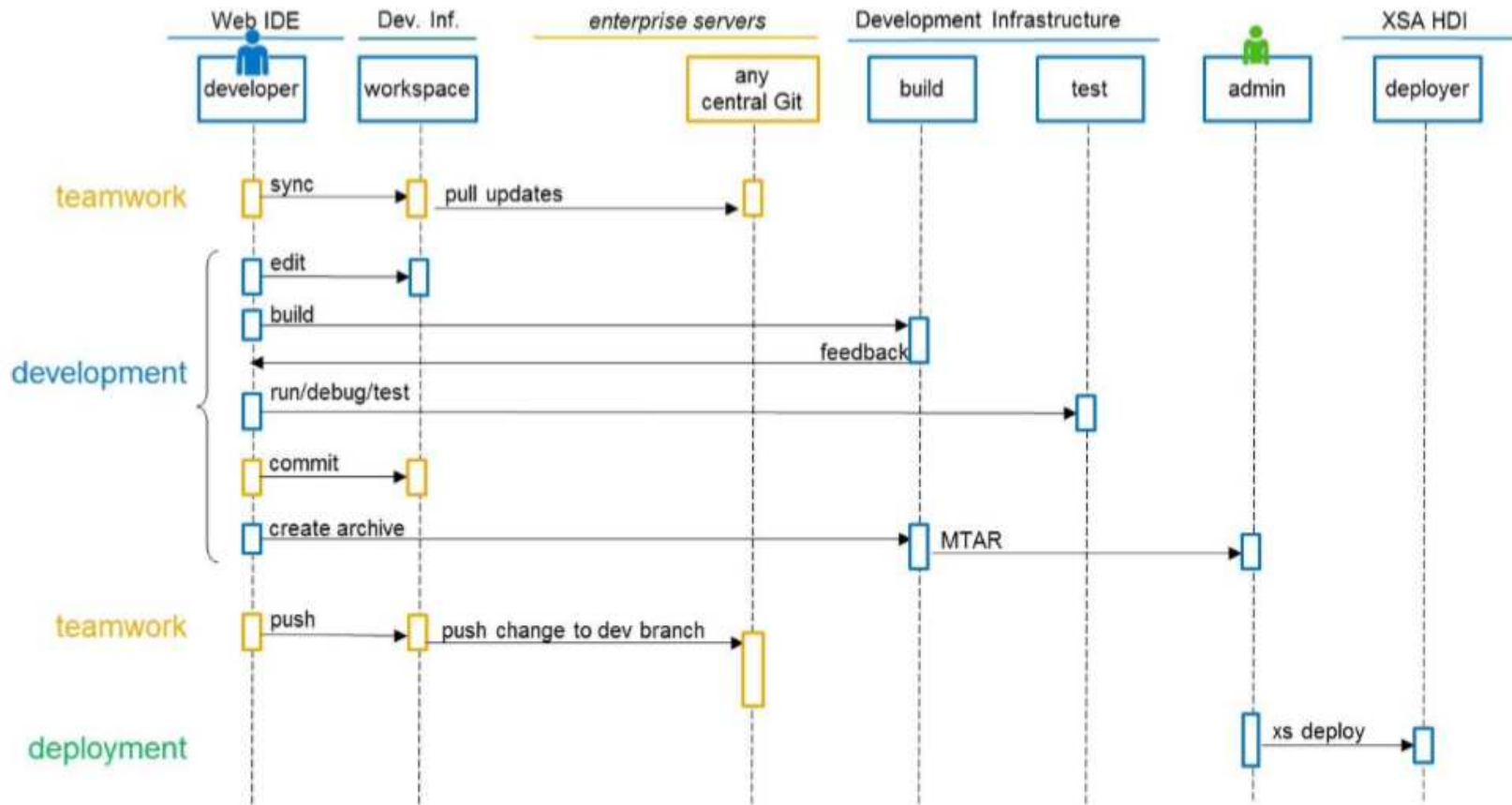
HANA Development Infrastructure (HDI)

© SAP

XSC vs XSA Dev Process



XSA Dev Process



© SAP

XSA Dev Process

Usually used components

- SAP HANA Web IDE
- Development Infrastructure (DI)
- GIT source code repository
- HANA Deployment Infrastructure (HDI)

The process itself works the following way:

- 1.The developer pulls the latest sources from the GIT repository into his Web IDE development workspace
- 2.He makes some changes to the code
- 3.He builds his local version of the project which is done by the Development Infrastructure
- 4.In order to run/test/debug his version he uses again the DI
- 5.The developer commits the changes to the local workspace
- 6.When he is satisfied with the changes he creates the MTAR archive which contains the built objects
- 7.From the local workspace the changes are pushed to the central GIT repository
- 8.The MTAR archive will be deployed by an administrator in the target system and made available

XSA Application Dev Tools

- **XS Advanced Development Tools**

- Code editors
- Debugging tools
- Unit Test tools
- Version Control tools
- Code visualization tool

- **XS Advanced Run-Time Tools**

- SAP HANA database catalog browser
- SQL Console
- SQL Debugger
- Job Log Viewer

- **XS Advanced Administration Tools**

- Application Monitor
- Organization and Space Management
- Application Role Builder
- SAML Identity Providers Configuration
- User Management
- SAP HANA Logical Database Setup
- SAP HANA Service Broker Configuration
- Job Scheduler Service Dashboard

© SAP

SAP Web IDE: Code Editor Tool

The screenshot shows the SAP Web IDE interface with several callout boxes explaining its components:

- Menu Bar:** provides access to all operations available in SAP Web IDE.
- Global Toolbar:** depending on the item that is activated in the workspace, you can choose from the icons in the global toolbar (icons of actions that are not applicable are grayed out).
- Left sidebar:** use the buttons to switch between the development workspace and user preferences.
- Package structure:** shows the file and folder structure of the workspace.
- Source code editor:** includes features like Code completion, Code checking, Syntax highlighting / Themes, Beautify file formatting, Basic Navigating and Editing, Commenting, Using multiple cursors, and Generating JSDoc Comment Snippets.
- Right sidebar:** use the buttons to switch between the different panes available in SAP Web IDE (for example, Git pane, Outline pane, and so on).
- Right-click folder or file to open context-sensitive menu:** provides actions for the selected file or folder.
- Message console:** displays application status messages such as "Application is running", "Application is stopping", and "Application stopped".

© SAP

SAP HANA

Runtime Tools: Catalog

The screenshot shows the SAP HANA Runtime Tools: Catalog interface. The main window is titled "SAP HANA Runtime Tools: Catalog" and includes a "logout" button in the top right corner. The interface is divided into several panes:

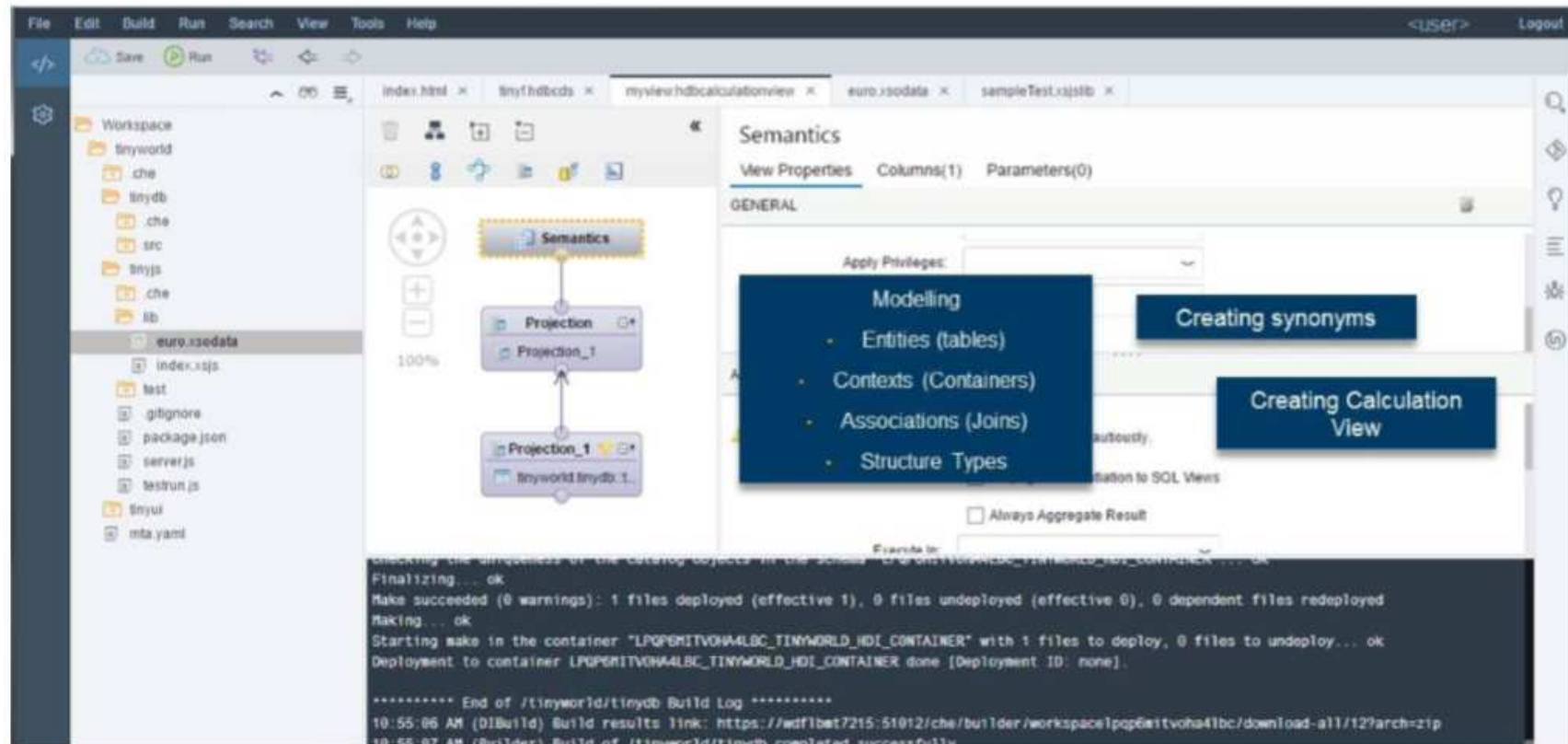
- Left sidebar:** A tree view showing the database catalog structure, including folders for Synonyms, Tables, Triggers, Views, Functions, and Indexes. A blue callout box points to this sidebar with the text: "Left sidebar - use the buttons to switch between the database catalog and user preferences." A specific folder is highlighted with a blue callout box: "Catalog structure".
- Global Toolbar:** A horizontal bar at the top containing various icons for actions. A blue callout box explains: "Global Toolbar - depending on the item that is activated in the workspace, you can choose from the icons in the global toolbar (icons of actions that are not applicable are grayed out)."
- Workspace:** The central area where a SQL statement is entered and executed. The statement is: `select * from "LPOPHNITVOH4ALBC_TINYWORLD_HDI_CONTAINER"."tinyworld.tinydb.tinyf.world"`. The result is displayed in a table with 2 rows:

	continent
1	Europe
2	Asia

. A blue callout box points to the workspace with the text: "Right-click folder or file to open context-sensitive menu".
- Message Console:** A panel at the bottom showing the execution log: "1:02:14 PM (SQL Editor) Statement 'select * from ...' successfully executed in 1 ms." A blue callout box points to this panel: "Message console".
- Right sidebar:** A panel for debugging and search, containing sections for Expressions, Variables, Call Stack, Line Breakpoints, and Watchpoints. A blue callout box explains: "Right sidebar - use the buttons to switch between the debugger and search pane".

© SAP

SAP Web IDE: Graphical Editor



© SAP

SAP Web IDE: Git Source Control System

The screenshot shows the SAP Web IDE interface with the Git pane open. The menu bar includes File, Edit, Build, Run, Search, View, and Tools. The Git pane contains a 'Repository' dropdown, a 'Branch' dropdown, and five icons for Git operations: Pull, Fetch, Rebase, Merge, and Reset. Below these is a 'Commit' section with a 'Stage All' checkbox and a 'Discard All Changes' link. A status table is present, and a 'Commit Description' field is at the bottom.

Git Pane – for executing Git commands on a specific Git repository

Cloning a repository

Status table –listing all uncommitted changes that you have made to your project

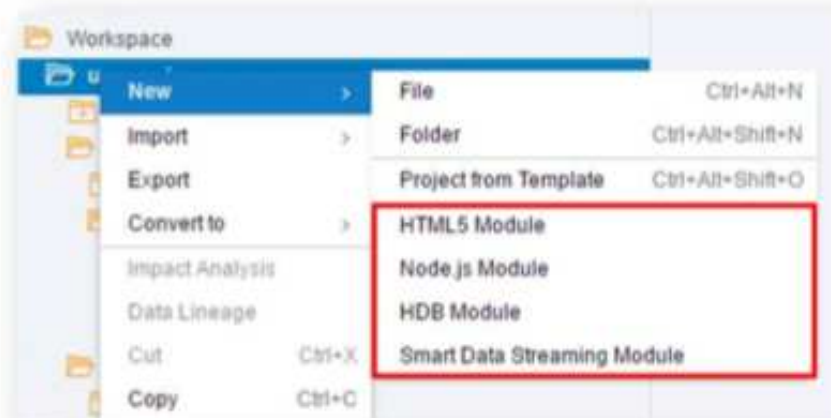
- Git Decorations
- Cloning Repositories
- Fetching Changes
- Rebasing Changes
- Merging Changes
- Pulling Changes
- Staging Files
- Committing Changes
- Pushing Changes
- Resetting a Branch

© SAP

XSA Multi Target App Example

Example of an MTA project in the WEB IDE

- Sub-structures for
 - HDB Module
 - JS Module
 - Web Module
- Configuration files
 - .hdiconfig
 - package.json
 - mta.yaml



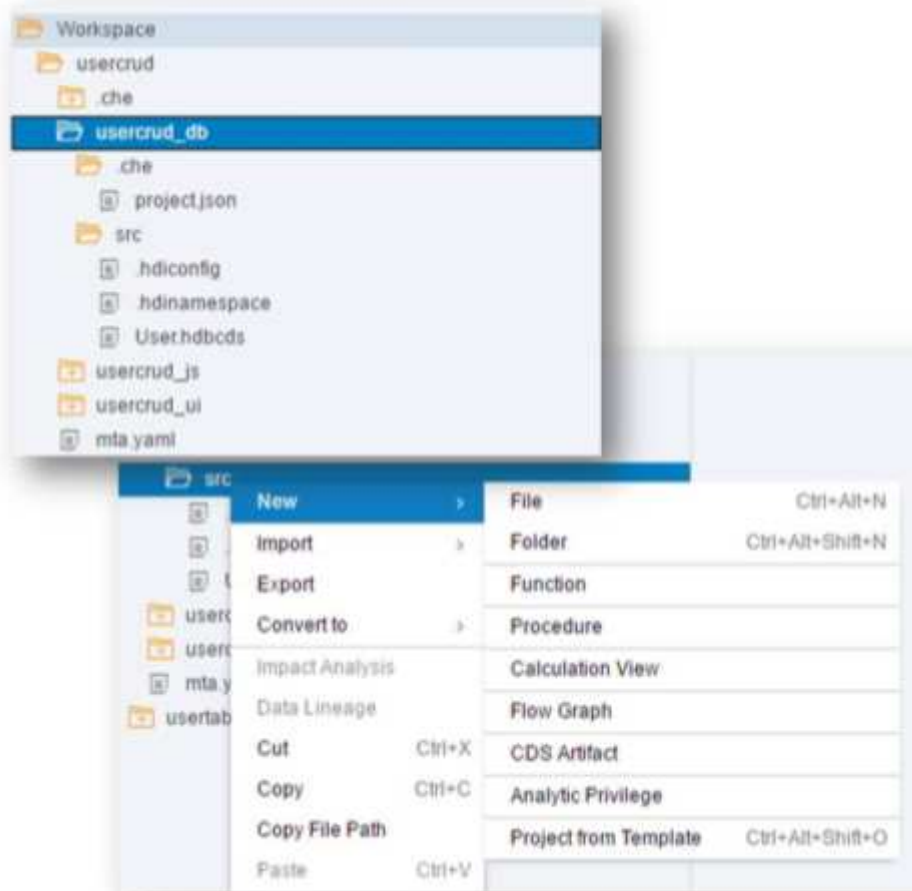
© SAP

XSA Multi Target App

HDB Module (to create DB content)

Module for Database Content

- Standard files
 - project.json
 - .hdiconfig
 - .hdinamespace
- Web IDE templates
 - Function
 - Procedure
 - Calculation View
 - Flow Graph
 - CDS Artefact
 - Analytic Privilege



© SAP

XSA Multi Target App

.hdiconfig

.hdiconfig

- Contains the plugin name and version for every supported file suffix
- Filled automatically by the Web IDE
- Versions can be adapted to match different HANA DB Versions

```
.hdiconfig x
1 {
2   "file_suffixes": {
3     "hdbflowgraph": {
4       "plugin_name": "com.sap.hana.di.flowgraph",
5       "plugin_version": "12.1.0"
6     },
7     "hdbreptask": {
8       "plugin_name": "com.sap.hana.di.reptask",
9       "plugin_version": "12.1.0"
10    },
11    "hdbsynonym": {
12      "plugin_name": "com.sap.hana.di.synonym",
13      "plugin_version": "12.1.0"
14    },
15    "hdbsynonymconfig": {
16      "plugin_name": "com.sap.hana.di.synonym.config",
17      "plugin_version": "12.1.0"
18    },
19    "hdbtable": {
20      "plugin_name": "com.sap.hana.di.table",
21      "plugin_version": "12.1.0"
22    },
23    "hdbdropcreatetable": {
24      "plugin_name": "com.sap.hana.di.dropcreatetable",
25      "plugin_version": "12.1.0"
26    },
27  }
```

XSA Multi Target App

.hdinamespace

.hdinamespace

- Contains namespace configuration for the HDB module
- name
- subfolder:
 - append
 - ignore

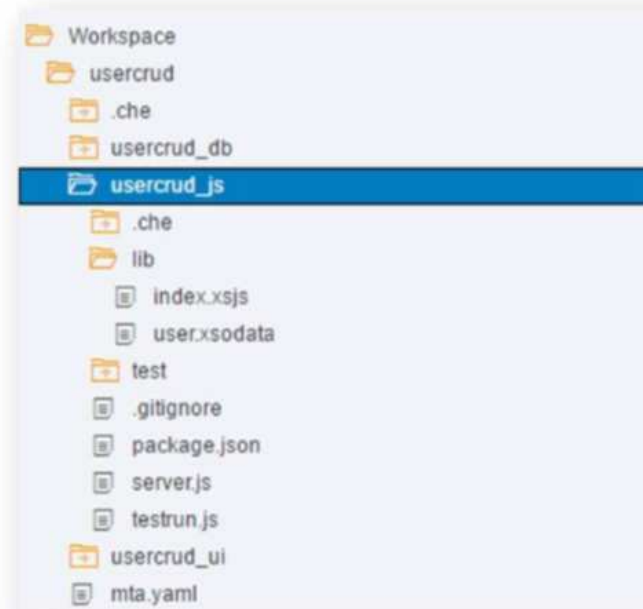
```
.hdinamespace x
1 {
2   "name" : "com.sap.hana.example",
3   "subfolder" : "<[append | ignore]>"
4 }
5
```

Design-time Resource Path	Name of Run-Time Object (append)
/src/	<namespace>::<objectName>
/src/load	<namespace>.load::<objectName>
/src/load/data	<namespace>.load.data::<objectName>

XSA Multi Target App Node.js (JS) Module

Module for Server-Side JavaScript

- Standard files
 - package.json
 - server.js
 - testrun.js
 - /lib/index.xsjs



XSA Multi Target App package.json

XS Advanced Application Package Descriptor

- Used by the Node Package Manager (NPM)
- name
- version
- dependencies
- engines (Node.js version)
- scripts
- main (main Program)

```
package.json x
1 {
2   "dependencies": {
3     "sap-xsenv": "1.2.2",
4     "sap-xsjs": "1.6.4"
5   },
6   "description": "",
7   "devDependencies": {
8     "sap-xsjs-test": "1.0.7"
9   },
10  "files": [],
11  "main": "server.js",
12  "name": "usercrud_js",
13  "scripts": {
14    "start": "node server.js",
15    "test": "node testrun.js"
16  },
17  "version": "1.0.0"
18 }
```

XSA Multi Target App server.js

Main JavaScript file

- Started by default when running the Node.js module
- XSJS compatibility layer
- XSENV variables
- HANA DB connection
- XSUAA security service
- Webserver start

XS Advanced
Environment Variables (XSENV)

XS Advanced
User Account and
Authorization Service (XSUAA)

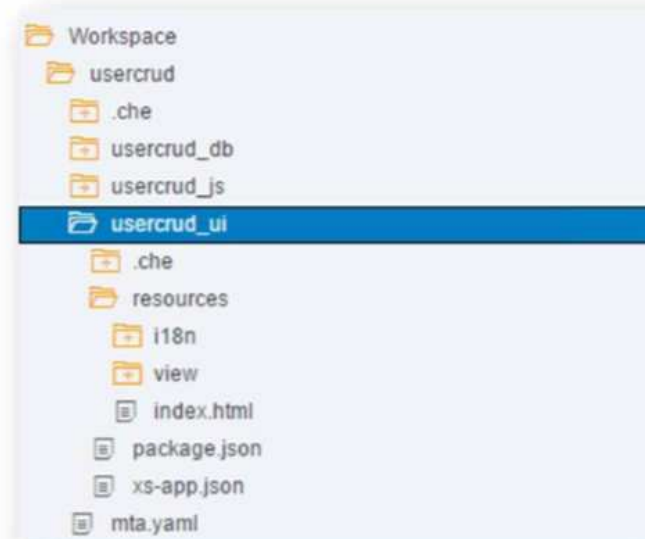
```
server.js x
1  /*eslint no-console: 0, no-unused-vars: 0*/
2  "use strict";
3
4  var xsjs = require("sap-xsjs");
5  var xsekv = require("sap-xsekv");
6  var port = process.env.PORT || 3000;
7
8  var options = xsjs.extend({
9    anonymous : true, // remove to authenticate calls
10   redirectUrl : "/index.xsjs"
11 });
12
13 //configure HANA
14 try {
15   options = xsjs.extend(options, xsekv.getServices({ hana: {tag: "hana"} }));
16 } catch (err) {
17   console.error(err);
18 }
19
20 // configure UAA
21 try {
22   options = xsjs.extend(options, xsekv.getServices({ uaa: {tag: "xsuaa"} }));
23 } catch (err) {
24   console.error(err);
25 }
26
27 // start server
28 xsjs(options).listen(port);
29
30 console.log("Server listening on port %d", port);
31
```

XSA Multi Target App

HTML5 Module

HTML5 Module for frontend/user interface

- Contains static content to be delivered to the client side
- Standard files
 - package.json
 - xs-app.json



XSA Multi Target App

mta.yaml

Multi Target Application Development Descriptor

- Parts
 - Global elements
 - Modules
 - Resources
 - Properties
 - Parameters

The development descriptor (mta.yaml) is used to define the elements and dependencies of an XS advanced-compliant multi-target application (MTA)

```
mta.yaml x
1  _schema-version: 2.0.0
2  ID: usercrud
3  version: 0.0.1
4
5  modules:
6  - name: usercrud_db
7    type: hdb
8    path: usercrud_db
9    requires:
10   - name: hdi-container
11
12  - name: usercrud_js
13    type: nodejs
14    path: usercrud_js
15
16  # ----- dependency on usercrud_db
17  requires:
18   - name: usercrud_db
19   - name: hdi-container
20
21  #----- exposes SERVICE URL to consumers
22  provides:
23   - name: usercrud_js_api
24     properties:
25       service_url: ${default-url}
26
```

XSA Multi Target App

mta.yaml - Global elements

Global elements

- `_schema_version*`
- `ID*`
- `description`
- `version*`
- `provider`
- `copyright`

```
mta.yaml x
1  _schema-version: 2.0.0
2  ID: usercrud
3  version: 0.0.1
4
5  modules:
6  - name: usercrud_db
7    type: hdb
8    path: usercrud_db
9  - name: usercrud_js
10   type: nodejs
11   path: usercrud_js
12
13 # ----- dependency on usercrud_db
14 requires:
15   - name: usercrud_db
16   - name: hdi-container
17
18 #----- exposes SERVICE URL to consumers
19 provides:
20   - name: usercrud_js_api
21     properties:
22       service_url: ${default-url}
23
24
25
26
```

XSA Multi Target App

mta.yaml Modules

Modules

- Source modules of the MTA project
 - name*
 - type*
 - path*
 - description
 - requires
 - provides
 - properties
 - parameters

```
mta.yaml x
1  _schema-version: 2.0.0
2  ID: usercrud
3  version: 0.0.1
4
5  modules:
6  - name: usercrud_db
7    type: hdb
8    path: usercrud_db
9    requires:
10   - name: hdi-container
11
12  - name: usercrud_js
13    type: nodejs
14    path: usercrud_js
15
16  # ----- dependency on usercrud_db
17  requires:
18   - name: usercrud_db
19   - name: hdi-container
20
21  #----- exposes SERVICE URL to consumers
22  provides:
23   - name: usercrud_js_api
24     properties:
25       service_url: ${default-url}
26
```

XSA Multi Target App

mta.yaml Resources

Resources

- Dependency declaration on external resources
 - name*
 - type*
 - description
 - properties
 - parameters

```
mta.yaml x
31
32 - name: usercrud_ui
33   type: html5
34   path: usercrud_ui
35
36 # -- requires usercrud_js service URL
37 requires:
38   - name: usercrud_js_api
39     group: destinations
40     properties:
41       name: usercrud_js_be
42       url: ~{service_url}
43
44
45 resources:
46 - name: hdi-container
47   type: com.sap.xs.hdi-container
48
```

XSA Multi Target App

mta.yaml

Properties

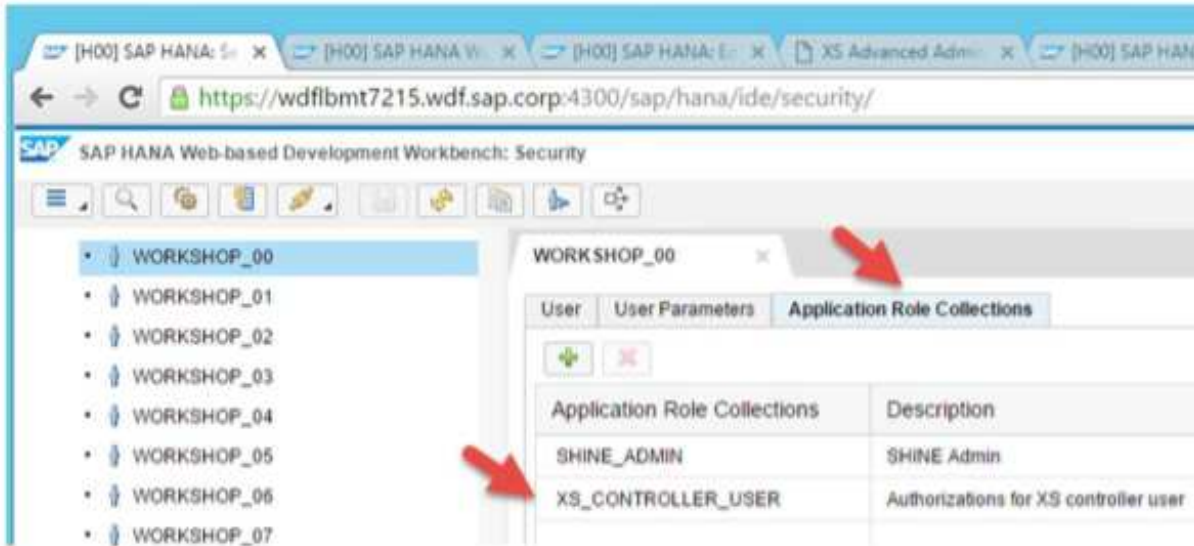
- Variables specified during deployment
 - `<key>: <value>`
 - use defined placeholders via `${<parameter>}`
 - refer to property values via `~{<property>}`

```
mta.yaml x
20 # ----- dependency on usercrud_db
21   requires:
22     - name: usercrud_db
23     - name: hdi-container
24
25 #----- exposes SERVICE URL to consumers
26   provides:
27     - name: usercrud_js_api
28       properties:
29         service_url: ${default-url}
30
31
32 - name: usercrud_ui
33   type: html5
34   path: usercrud_ui
35
36 # -- requires usercrud_js service URL
37   requires:
38     - name: usercrud_js_api
39       group: destinations
40       properties:
41         name: usercrud_js_be
42         url: ~{service_url}
```


Creating an XSA Project Auth

For HANA users to be able to access XS Advance you need:

- To have the XS_CONTROLLER_USER collection assigned in the user maintenance screen.



The screenshot shows the SAP HANA Web-based Development Workbench: Security interface. The browser address bar displays <https://wdfbmt7215.wdf.sap.corp:4300/sap/hana/ide/security/>. The interface includes a navigation pane on the left with a tree view of workshop folders (WORKSHOP_00 through WORKSHOP_07). The main content area shows the 'WORKSHOP_00' user selected, with the 'Application Role Collections' tab active. A table lists the assigned role collections:

Application Role Collections	Description
SHINE_ADMIN	SHINE Admin
XS_CONTROLLER_USER	Authorizations for XS controller user

Two red arrows point to the 'Application Role Collections' tab and the 'XS_CONTROLLER_USER' row in the table.

SQLScript Triggers

Example 1/2

```

SQL SQL
1 CREATE TRIGGER "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.sql/add_insert_to_log"
2 AFTER INSERT ON "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Products"
3 REFERENCING NEW ROW newrow FOR EACH ROW
4 BEGIN
5
6 INSERT INTO "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.ProductLog"
7 VALUES(:newrow.PRODUCTID, now(), CURRENT_USER,
8 :newrow.PRODUCTID || ' has been created');
9 END;

```

```

SQL SQL
1 INSERT into "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Products"
2 values( 'ProductA', 'PR', 'Handheld', '0000000033', '20121003', '0000000033', '20121003',
3 '1000000149', '1000000150', '0100000029', 1, 'EA', 0.5, 'KG', 'CAD', 2490,
4 '/sap/hana/democontent/epmSP6/data/images/HT-7030.jpg', 0.09, 0.15, 0.1, 'M');

```

SQL Result Result

select PRODUCTID, CATEGORY, PRICE from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Products" where PRODUCTID = 'ProductA'

	PRODUCTID	CATEGORY	PRICE
1	ProductA	Handheld	2,490

SQL Result Result

select * from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.ProductLog"

	PRODUCTID	DATETIME	USERNAME	LOGTEXT
1	ProductA	Jun 27, 2014 10:40:10.0 AM	SYSTEM	ProductA has been created

SQLScript Triggers

Example 2/2

```

SQL
1 CREATE TRIGGER "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.sql/add_update_price_to_log"
2 AFTER UPDATE ON "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Products"
3 REFERENCING NEW ROW newrow, OLD ROW oldrow FOR EACH ROW
4 BEGIN
5
6 declare lv_price_difference decimal(15,2) := 0;
7 if :oldrow.price <> :newrow.price then
8     lv_price_difference := :newrow.price - :oldrow.price;
9     INSERT INTO "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.ProductLog"
10 VALUES(:newrow.PRODUCTID, now(), CURRENT_USER,
11         :newrow.PRODUCTID || ' has been updated with price difference of ' ||
12         to_decimal(lv_price_difference, 15, 2));
13 end if;
14
15 END;

```

```

SQL
1 UPDATE "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.Products"
2 SET PRICE = 2690.00 WHERE PRODUCTID = 'ProductA';
3

```

SQL Result

```
select PRODUCTID, CATEGORY, PRICE from "SAP_HANA_EPM_NEXT"."s"
where PRODUCTID = 'ProductA'
```

	PRODUCTID	CATEGORY	PRICE
1	ProductA	Handheld	2,690

SQL Result

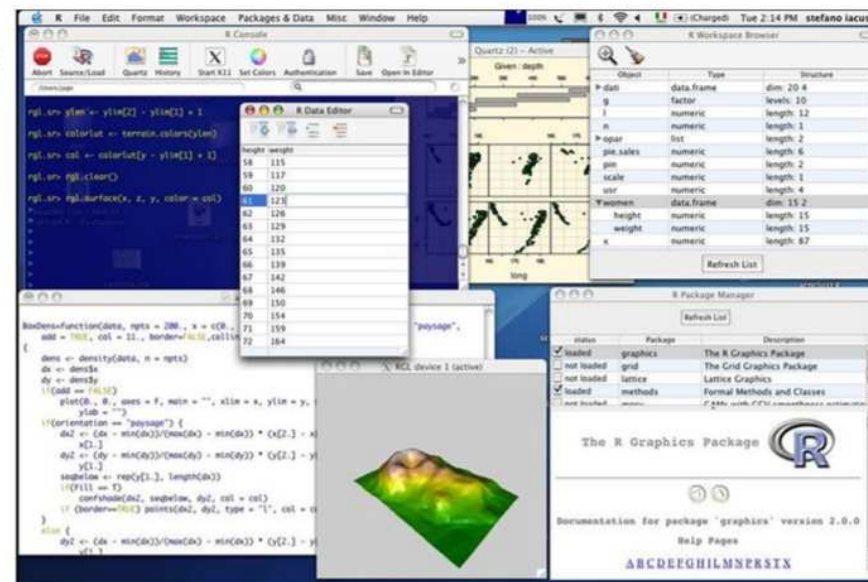
```
select * from "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext.data::EPM.MasterData.ProductLog"
```

	PRODUCTID	DATETIME	USERNAME	LOGTEXT
1	ProductA	Jun 27, 2014 10:40:10.0 AM	SYSTEM	ProductA has been created
2	ProductA	Jun 27, 2014 10:42:37.0 AM	SYSTEM	ProductA has been updated with price difference of 200.00

© SAP

R

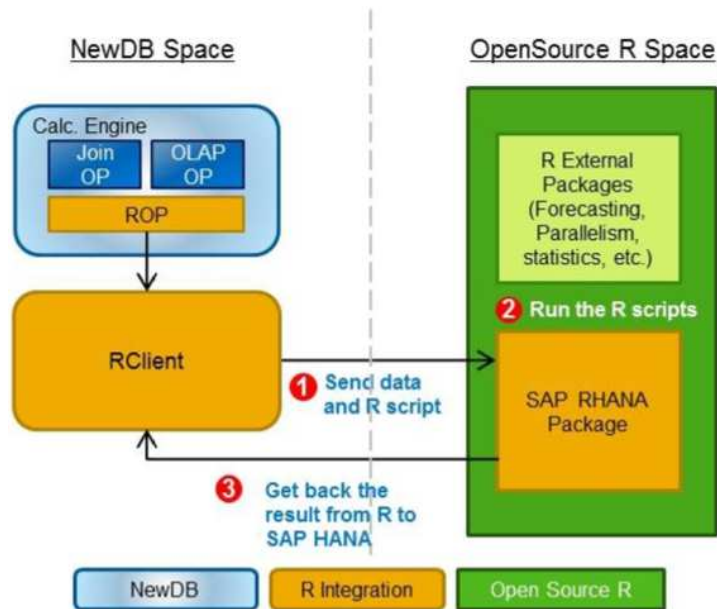
- An open source software language and environment for statistical computing and graphics with over 3000 add-on packages.
- <http://www.r-project.org/>
- The packages cover a wide range of topics:
 - Cluster Analysis & Finite Mixture Models
 - Probability Distributions
 - Computational Econometrics
 - Empirical Finance
 - Statistical Genetics
 - Graphic Displays, Dynamic Graphics, Graphic Devices, & Visualisation
 - Machine Learning & Statistical Learning
 - Medical Image Analysis
 - Multivariate Statistics
 - Natural Language Processing
 - Statistics for the Social Sciences
 - Time Series Analysis



© SAP

R

HANA Integration



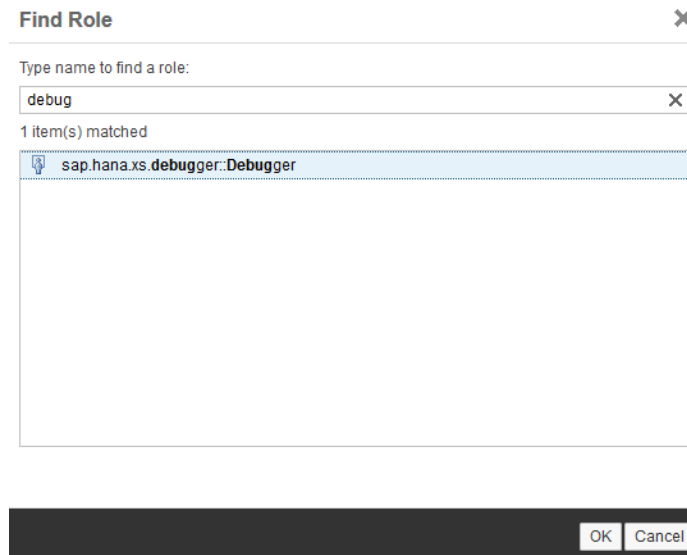
```
*build_product_list.procedure ✕  
ORG (SYSTEM)  
SAP HANA SQLScript Local Table Types  
1 CREATE PROCEDURE build_product_list ( )  
2   LANGUAGE RLANG  
3   SQL SECURITY INVOKER  
4   --DEFAULT SCHEMA <default_schema_name>  
5   READS SQL DATA AS  
6 BEGIN  
7 /*****  
8   Write your procedure logic  
9   *****/  
10  ProductId <- c('HT-1000', 'HT-1095', 'HT-1100')  
11  Category <- c('Notebooks', 'Headset', 'Software')  
12  Price <- c(199.99, 299.99, 399.99)  
13  result <- as.data.frame(cbind(ProductId,Category,Price))  
14  
15 END;
```

XSJS

Debugging Precondition for
HANA Web-based Development Workbench



- Assign the database user the *sap.hana.xs.debugger::Debugger* role.

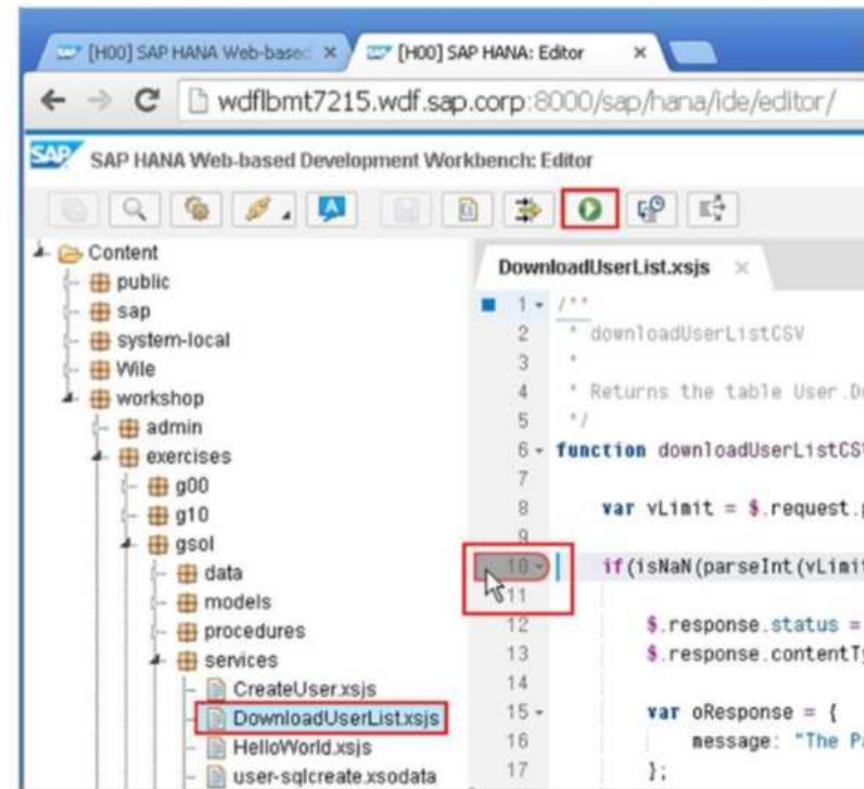


XSJS

Debugging

The debug functionality is also included in the **SAP HANA WEB IDE**.

- Open the XSJS Program that you want to debug.
- Set a breakpoint by clicking the line number.
- Execute the program.
- A new browser window will appear, executing the program.
- If you switch back to the WEB IDE window, you can access the debugger.



XSJS

Debugging

Resume (F7)
Step Into (F3)
Step Over (F4)
Step Return (F5)

Variables

Executed Code

```
DownloadUserList.xsjs
1  /**
2   * downloadUserListCSV
3   *
4   * Returns the table User.Details as a CSV file
5   */
6  function downloadUserListCSV() {
7
8     var vLimit = $.request.parameters.get("limit");
9
10
11     if (!isNaN(parseInt(vLimit, 10)) && vLimit > 0) {
12
13         $.response.status = $.net.http.BAD_REQUEST;
14         $.response.contentType = "text/json";
15
16         var oResponse = {
17             message: "The Parameter 'limit' must be set!"
18         };
19         $.response.setBody(JSON.stringify(oResponse));
20         return;
21     }
22
23     var oConnection = $.hdb.getConnection();
24
25     var sQuery = "SELECT * FROM /workshop/exercises/gsol/data/User.I
26                 * LIMIT ?";
27
28     var oResultSet = oConnection.executeQuery(sQuery, vLimit);
29
30     var sBody = "";
31 }
```


GIT



Git

Create Repo

This repository Search Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/username/repository.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# README.md" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/username/repository.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/username/repository.git
git push -u origin master
```

...or import code from another repository

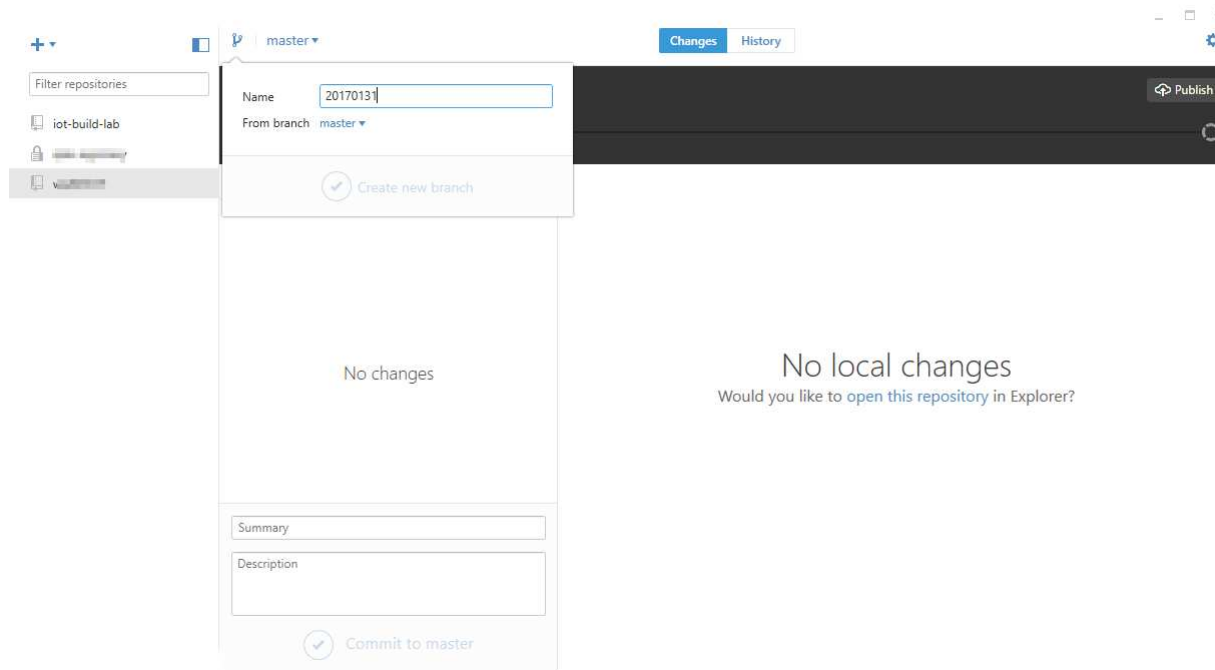
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

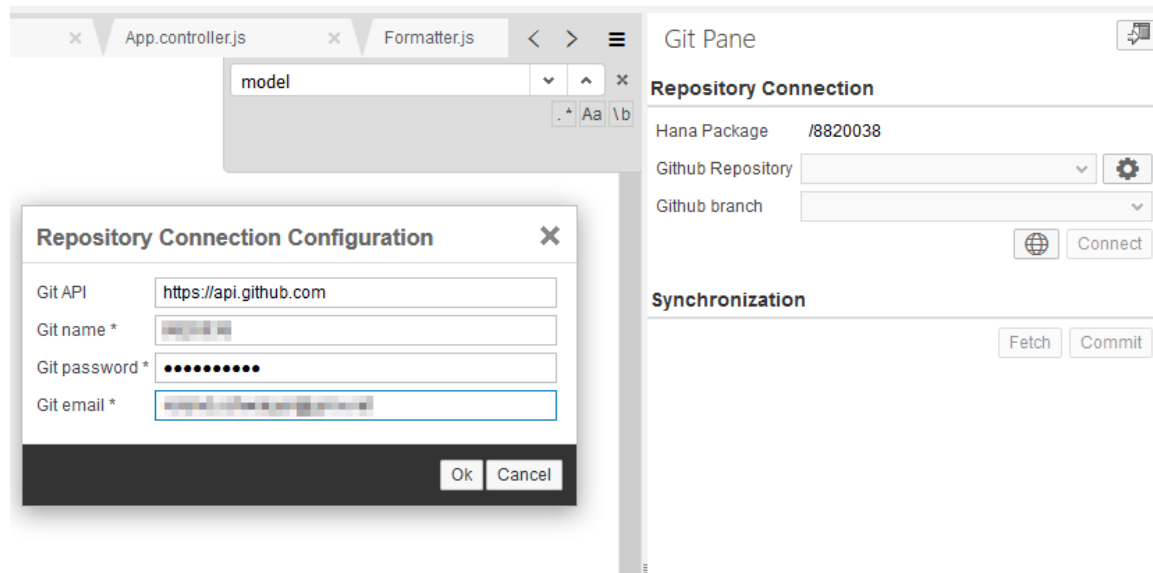
ProTip! Use the URL for this page when adding GitHub as a remote.

Git

Create Branch



Git in HANA



PREDICTION(?)

**Trial
Leider Nein**

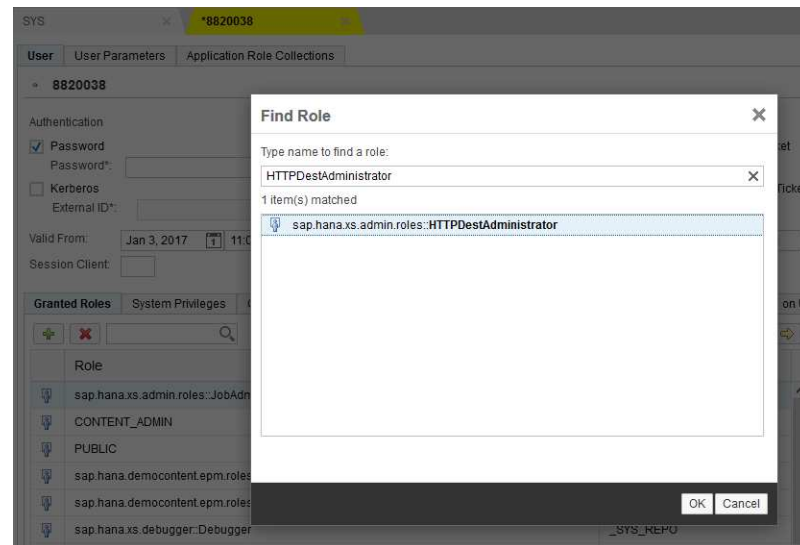
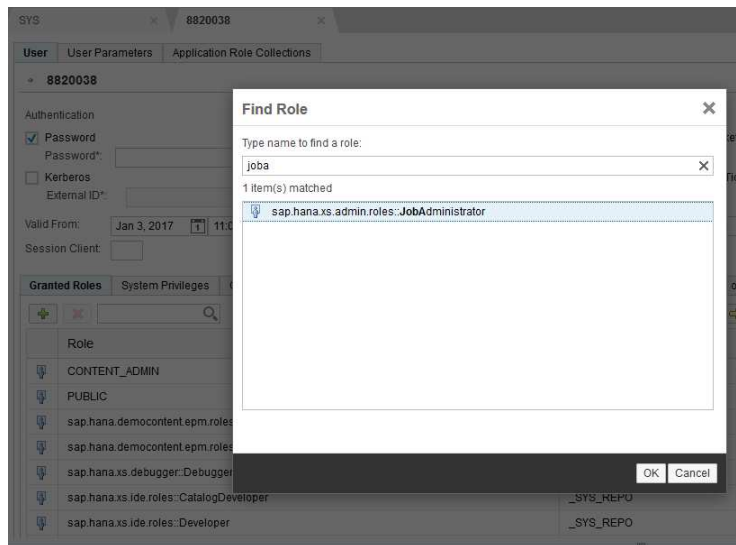


JOB SCHEDULE

- **Schedule a job** that triggers an **XS JavaScript application** that reads the latest value from the public **honeycomb service** available on the Internet.
- You also see how to **check** that the **XS job** is **working** and running on **schedule**.

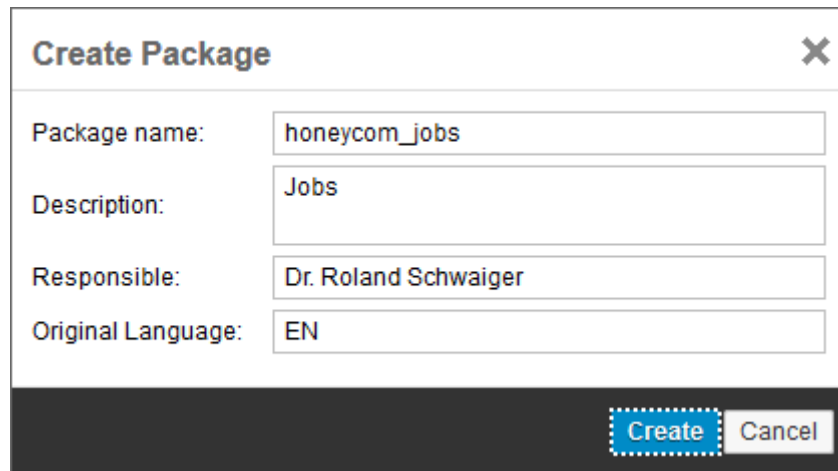
Job Schedule Prerequisites

- You have the privileges granted in the SAP HANA user role `sap.hana.xs.admin.roles::JobAdministrator`
- You have the privileges granted in the SAP HANA user role `sap.hana.xs.admin.roles::HTTPDestAdministrator`



Job Schedule

- Create new package for the job definitions



The image shows a 'Create Package' dialog box with the following fields and values:

Field	Value
Package name:	honeycom_jobs
Description:	Jobs
Responsible:	Dr. Roland Schwaiger
Original Language:	EN

At the bottom right of the dialog box, there are two buttons: 'Create' (highlighted with a blue dashed border) and 'Cancel'.

Job Schedule

yahoo.xsjs

```
function readStock(input) {
    var stock = input.stock;

    var dest = $.net.http.readDestination("yahoo", "yahoo");
    var client = new $.net.http.Client();
    var req = new $.web.WebRequest($.net.http.GET, "/d/quotes.csv?f=a&s=" + stock);

    client.request(req, dest);
    var response = client.getResponse();

    var stockValue;
    if(response.body)
        stockValue = parseInt(response.body.asString(), 10);

    var sql = "INSERT INTO stock_values VALUES (NOW(), ?)";
    var conn = $.db.getConnection();
    var pstmt = conn.prepareStatement(sql);
    pstmt.setDouble(1, stockValue);
    pstmt.execute();
    conn.commit();
    conn.close();
}
```

Job Schedule

HTTP destination (sap.hana.xs.admin.roles::HTTPDestAdministrator)

yahoo.xshttpdest

host = "download.finance.yahoo.com";

port = 80;

Job Schedule

XS job file

- The XS job file uses a cron-like syntax to define a schedule at which the XS JavaScript must run
- This job file triggers the script yahoo.xsjs on the 59th second of every minute and provides the name “SAP.DE” as the parameter for the stock value to check.

Job Schedule

yahoo.xsjob

```
{
  "description": "Read stock value",
  "action": "yahoo:yahoo.xsjs::readStock",
  "schedules": [
    {
      "description": "Read current stock value",
      "xscron": "* * * * * 59",
      "parameter": {
        "stock": "SAP.DE"
      }
    }
  ]
}
```

Job Schedule

XS job's runtime configuration

(sap.hana.xs.admin.roles::JobAdministrator)

XS Job Dashboard

- Start the SAP HANA XS Administration Tool at the following URL: `http://<WebServerHost>:80<SAPHANAinstance>/sap/hana/xs/admin/`.

You need to specify the following details:

- User
The user account in which the job runs, for example, SYSTEM
- Password
The password required for user, whose account is used to run the job.
- Locale
The language encoding required for the locale in which the job runs, for example, en_US
- Start/Stop time
An optional value to set the period of time during which the job runs. Enter the values using the syntax used for the SAP HANA data type LocalDate and LocalTime, for example, 2014-11-05 00:30:00 (thirty minutes past midnight on the 5th of November 2014).
- Active
Enable or disable the job schedule

Save

Job Schedule

- Enable the job-scheduling feature in SAP HANA XS
- In the XS Job Dashboard set the Scheduler Enabled toggle button to YES
 - > Which sets the SAP HANA configuration variable: xsengine.ini -> scheduler -> enabled
- Check the job logs to ensure the XS job is active and running according to the defined schedule. You can view the xsjob logs in the XS Job Dashboard tab of the SAP HANA XS Administration Tool.

Job Schedule



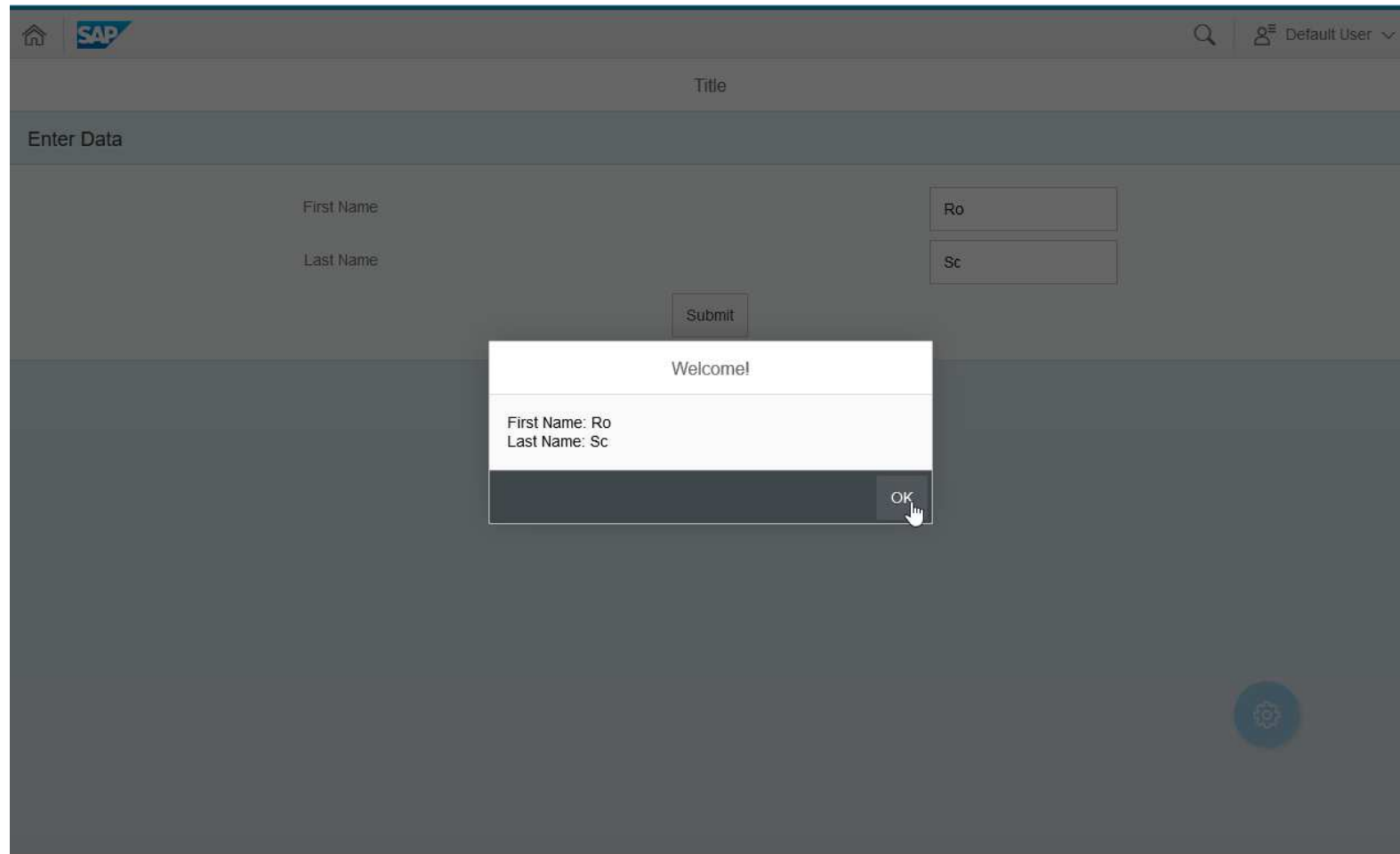
Exercise

UI5

1. Create new SAPUI5 project called panel-basic with an initial XML view called **Main**.
2. Insert a **sap.m.Panel** with headerText **Enter Data**.
3. Create tree **sap.m.FlexBox** layouts with **alignItems Center** and **justifyContent SpaceAround**.
4. Add **sap.m.Label** objects with text **FirstName / LastName** and ID **idInputFirstName / idInputLastName** to the two first FlexBoxes.
5. Add the **sap.m.Input** objects with type text behind each.
6. Add the **sap.m.Button** to the last FlexBox and assign a callback method **onButtonPressed** to the press event.
7. Include the **sap.m.MessageBox** in the view controllers define array **sap/m/MessageBox** and **MessageBox** in the parameter list of the function;
8. Implement the buttons callback method **onButtonPressed**.
9. Read the entered value from the two input fields **FirstName** and **LastName**.
10. Show a message on the screen with the retrieved values.

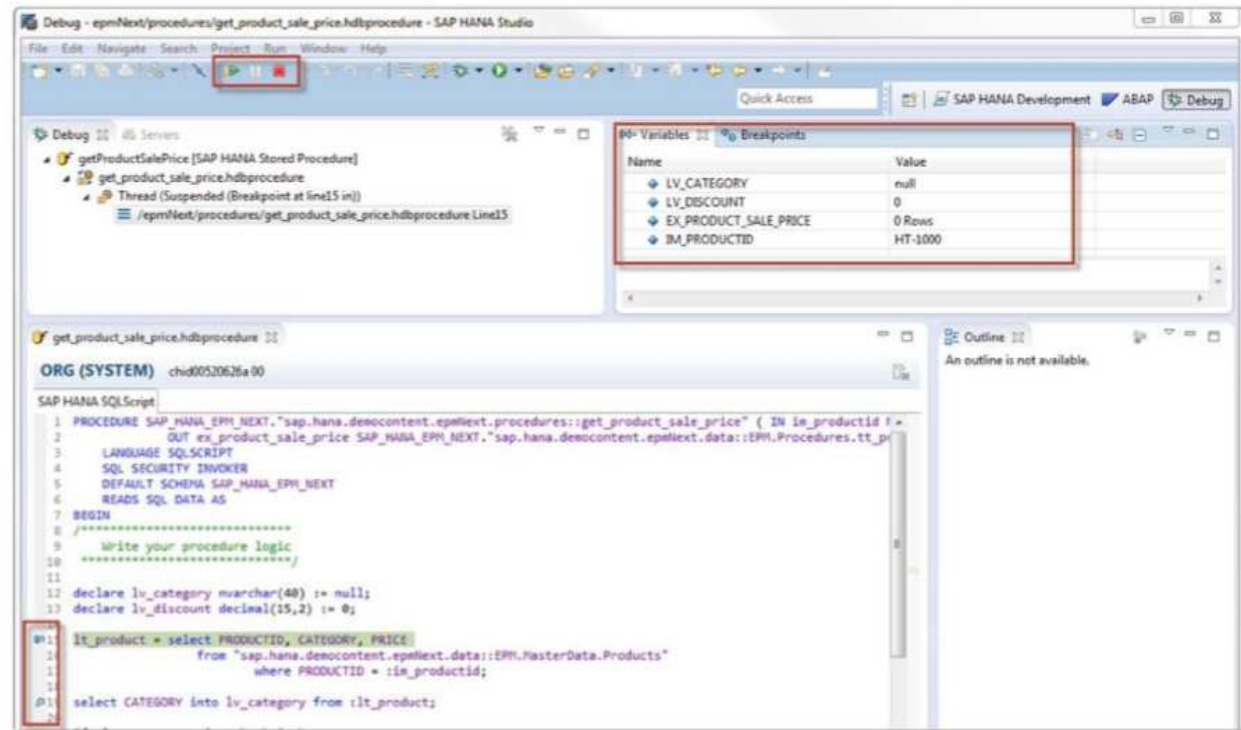
Exercise

UI5



SQLScript Debugging in HANA Dev Studio

- Resume/Terminate
- Variable evaluation
- Breakpoint management
- Break on breakpoints
- Basic step debugging



We are not using HANA Dev Studio

© SAP

Exercise

SQLScript & oData

- (optional) First create a table PRODUCT using SQLScript in the SQL console of the catalog view
- Create an oData validation exit based on SQLScript for the update operation
 - (optional) honeycomb_data:product.hdbtable
 - Create a table type honeycomb_data:error.hdbtabletype

Hint

SQLScript & oData

--REPLACE <YOUR SCHEMA> WITH YOUR SCHEMA NAME

-- Create Product table

```
create column table "<YOUR SCHEMA>". "PRODUCT"(  
    "PRODUCT_ID" INTEGER null primary key,  
    "PRODUCT_NAME" VARCHAR (100) null default "  
);
```

```
insert into "<YOUR SCHEMA>". "PRODUCT" values(1,'Shirts');  
insert into "<YOUR SCHEMA>". "PRODUCT" values(2,'Jackets');  
insert into "<YOUR SCHEMA>". "PRODUCT" values(3,'Trousers');  
insert into "<YOUR SCHEMA>". "PRODUCT" values(4,'Coats');  
insert into "<YOUR SCHEMA>". "PRODUCT" values(5,'Purse');
```

Hint

SQLScript & oData

```
table.schemaName = "8820038";
table.tableType = COLUMNSTORE;
table.columns = [
    {name = "PRODUCT_ID"; sqlType = INTEGER; nullable = false;},
    {name = "PRODUCT_NAME"; sqlType = NVARCHAR; nullable =
true; length = 100;
defaultValue = "";}}];
table.indexes = [
    {name = "MYINDEX1"; unique = true; indexColumns =
["PRODUCT_NAME"];}}];
table.primaryKey.pkcolumns = ["PRODUCT_ID"];
```

Solution

SQLScript & oData

The screenshot displays the SAP HANA Web-based Development Workbench interface. The top navigation bar includes the SAP logo and the title "SAP HANA Web-based Development Workbench: Catalog". Below the navigation bar is a toolbar with icons for menu, search, SQL, folder, edit, execute, save, download, and refresh. The left sidebar shows a catalog tree with the following structure:

- Favorite (1)
- Catalog
 - Public Synonyms
 - 8820038
 - Column Views
 - Functions
 - Indexes
 - Procedures
 - Sequences
 - Synonyms
 - Tables
 - 8820038.honeycomb.honeycomb_data::Honeycomb
 - PRODUCT
 - Triggers
 - Views

The main editor area shows a SQLScript file named "untitled1.sql" with the following code:

```
1 CREATE COLUMN TABLE "8820038"."PRODUCT"(  
2     "PRODUCT_ID" INTEGER NULL PRIMARY KEY,  
3     "PRODUCT_NAME" VARCHAR(100) DEFAULT '' NULL  
4 );  
5 INSERT INTO "8820038"."PRODUCT" VALUES(1, 'Shirts');  
6 INSERT INTO "8820038"."PRODUCT" VALUES(2, 'Jackets');  
7 INSERT INTO "8820038"."PRODUCT" VALUES(3, 'Trousers');  
8 INSERT INTO "8820038"."PRODUCT" VALUES(4, 'Coats');  
9 INSERT INTO "8820038"."PRODUCT" VALUES(5, 'Purse');  
10
```

Red circles with numbers 1, 2, and 3 are overlaid on the image. Circle 1 is on the "Favorite" icon, circle 2 is on the "PRODUCT" table name in the SQL script, and circle 3 is on the "Execute" icon in the toolbar.

Solution

SQLScript & oData

The screenshot shows the SAP HANA Web-based Development Workbench: Catalog interface. The left sidebar displays a tree view of the catalog structure, with 'PRODUCT' selected under the 'Tables' folder. A red arrow points to this selection. The main area displays the table details for 'PRODUCT' in schema '8820038'. The table type is 'COLUMN'. Below this, a table lists the columns:

Column	Name	SQL Data Type	Dim	Column Store Data Type	Key	Not Null	Default	Comment
1	PRODUCT_ID	INTEGER		INT	(X1)	X		
2	PRODUCT_NAME	VARCHAR	100	STRING				

Solution

SQLScript & oData

product.hdbtable

x

```
1 table.schemaName = "8820038";
2 table.tableType = COLUMNSTORE;
3 table.columns = [
4     {name = "PRODUCT_ID"; sqlType = INTEGER; nullable = false;},
5     {name = "PRODUCT_NAME"; sqlType = NVARCHAR; nullable = true; length = 100;
6     defaultValue = "";}];
7 table.indexes = [
8     {name = "MYINDEX1"; unique = true; indexColumns = ["PRODUCT_NAME"];}];
9 table.primaryKey.pkcolumns = ["PRODUCT_ID"];
```


Solution

SQLScript & oData

File error.hdbdd

namespace

"8820038".honeycomb.honeycomb_data;

@Schema: '8820038'

context error {

 type error{

 "HTTP_STATUS_CODE": Integer;

 "ERROR_MESSAGE": String(100);

 "DETAIL": String(100);

 };

};

Solution

SQLScript & oData

File sample.odata:beforeupdate.hdbprocedure

```
procedure "sample.odata::beforeupdate"  
  (IN new "sample.odata::table", IN old "sample.odata::table", OUT error  
  "sample.odata::error")  
language sqlscript  
sql security invoker as  
  idnew INT;  
  idold INT;  
begin  
  select ID into idnew from :new;  
  select ID into idold from :old;  
if :idnew <= :idold then  
error = select 400 as http_status_code,  
  'invalid ID' error_message,  
  'the new value must be larger than the previous' detail from dummy;  
end if;  
end;
```

Solution

SQLScript & oData

```
service {  
    "sample.odata::table"  
        update events (before  
"sample.odata::beforeupdate");  
}
```

IOT INTEGRATION (?)

IoT Deploy

The screenshot displays the SAP Internet of Things Services Cockpit. At the top left is the SAP logo. The page title is "Internet of Things Services Cockpit". In the top right corner, there is a refresh icon and the user name "Roland Schwaiger" with a dropdown arrow. The main content area is divided into two sections: "Geräteverwaltung" (Device Management) and "Nachrichtenverwaltung" (Message Management). Under "Geräteverwaltung", there are three white cards, each with a count of "1" in the top right corner. The first card has a speech bubble icon and is labeled "Nachrichtentypen" (Message Types) with the subtitle "Alle Nachrichtentypen". The second card has a network icon and is labeled "Gerätetypen" (Device Types) with the subtitle "Alle Gerätetypen". The third card has a mobile phone icon and is labeled "Geräte" (Devices) with the subtitle "Alle registrierten Geräte". Under "Nachrichtenverwaltung", there are two cards. The first is a yellow card with a download icon, labeled "Deployment für Message Management Service" (MMS-Deployment). The second is a white card with a network icon, labeled "Nachrichten senden, anzeigen und weitere Aktionen" (MMS-Cockpit). At the bottom of the page, there is a dark grey footer bar containing the text: "Impressum", "Datenschutz", "Nutzungsbedingungen", "Copyright", "Markenzeichen", and "SAP.com".

IoT Deploy

< Deployment für Message Management Service

Der Message Management Service empfängt und verarbeitet Nachrichten, die von Geräten gesendet werden. Daneben bietet er Schnittstellen an, über die Nachrichten im Push-Verfahren an Geräte gesendet werden können. Das Deployment für die Komponente erfolgt im angegebenen Benutzerkonto der SAP HANA Cloud Platform.

Kontoeinstellungen

Host https://hanatrial.ondemand.com


Konto-ID

Benutzereinstellungen

Benutzername

Kennwort

[Deployment durchführen](#)



Nach dem Deployment können Sie den Status der Message-Management-Service-Anwendung im Java Application Dashboard Ihres Kontos für das SAP-HANA-Cloud-Platform-Cockpit überwachen.

<https://account.hanatrial.ondemand.com/cockpit#/acc/XXXXXXXXXXXX/app/iotmms/dashboard>

Nur Benutzer mit der Rolle "IoT-MMS-User" erhalten Zugriff auf das Dashboard des Message Management Service. Sie können diese Berechtigung im Abschnitt "Roles" im Java Application Dashboard konfigurieren.

IoT

Authorization 1/2

- Within the IoT Service: Configure ...

SAP HANA Cloud Platform Cockpit

Overview

Europe (Trial) / trial / Internet of Things Services

Internet of Things Services - Overview

Enabled

Service Description

Enable customers and partners to develop, customize, and operate IoT business applications in the cloud.

[Documentation](#)

[Go to Service](#)

Service Configuration

[Configure Internet of Things Services](#)

IoT

Authorization 2/2

- Assign the role

The screenshot displays the SAP HANA Cloud Platform Cockpit interface. The left sidebar contains navigation options: Overview, Monitoring, Configuration, Security, Roles (selected), OAuth Scopes, and Authentication Configuration. The main content area shows the 'Roles (All: 1)' configuration page. A table lists the role 'IoT-MMS-User' as a predefined role that is shared. Below the table, the 'IoT-MMS-User' details are shown, including 'Predefined: Provisioned by the application'. There are two sections for assignment: 'Individual Users' and 'Groups'. The 'Individual Users' section has an 'Assign' button and a table with one user ID and an 'Unassign' button. The 'Groups' section has an 'Assign' button and a message stating 'Role "IoT-MMS-User" is not assigned to any groups.'

Name	Type	Shared	Actions
IoT-MMS-User	Predefined	<input checked="" type="checkbox"/>	

User ID	Actions
[Redacted]	Unassign

Group: [Empty] Actions

Role "IoT-MMS-User" is not assigned to any groups.

IoT Authorization

The screenshot displays the SAP HANA Cloud Platform interface. The top right corner shows 'SAP HANA Cloud Platform'. The left sidebar contains a navigation menu with items: Overview, Applications, Java Applications, HTML5 Applications, HANA XS Applications, Subscriptions (highlighted), Services, and Persistence. The main content area shows 'Europe (Trial)' and 'p020420trial' in the top bar. Below this, it says 'Subscribed Java Applications (All: 3)'. A table lists three applications:

State	Provider Account	Application
✓	iotservices	iotcockpit
✓	iotservices	iotrdms
✓	trial	portal

Below the table, it says 'Subscribed HTML5 Applications (All: 8)'. Red arrows point from the 'Applications' menu item to the 'Subscriptions' menu item, and from the 'Subscriptions' menu item to the 'iotcockpit' application link in the table.

IoT Authorization

SAP HANA Cloud Platform Cockpit

Europe (Trial) / p00470trial / iotcockpit (iotervices)

Roles (All: 1)

New Role

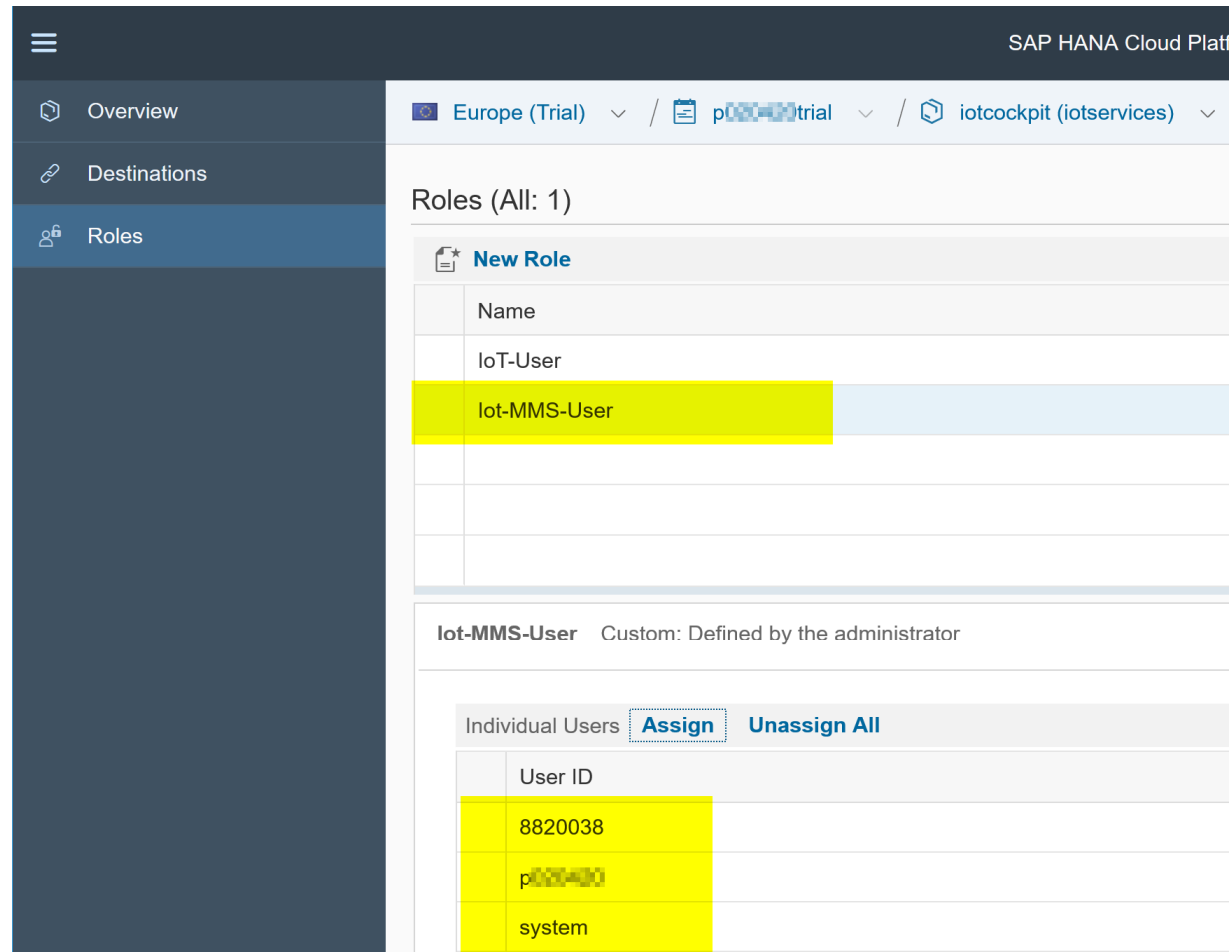
Name	Type
IoT-User	Predefined

IoT-User Predefined: Provisioned by the application

Individual Users **Assign** Unassign All

User ID	Actions
8820038	Unassign
8820038	Unassign
system	Unassign

IoT Authorization



The screenshot shows the SAP HANA Cloud Platform IoT Cockpit interface. The left sidebar contains navigation options: Overview, Destinations, and Roles (selected). The main content area displays the 'Roles (All: 1)' section. A 'New Role' button is visible. Below it, a table lists roles, with 'IoT-User' and 'IoT-MMS-User' highlighted in yellow. The 'IoT-MMS-User' role is further detailed as 'Custom: Defined by the administrator'. Below this, there are buttons for 'Assign' and 'Unassign All' under the 'Individual Users' section. A table lists individual users, with '8820038', 'p000400', and 'system' highlighted in yellow.

SAP HANA Cloud Platform

Europe (Trial) / p000400trial / iotcockpit (iot-services)

Roles (All: 1)

New Role

Name
IoT-User
IoT-MMS-User

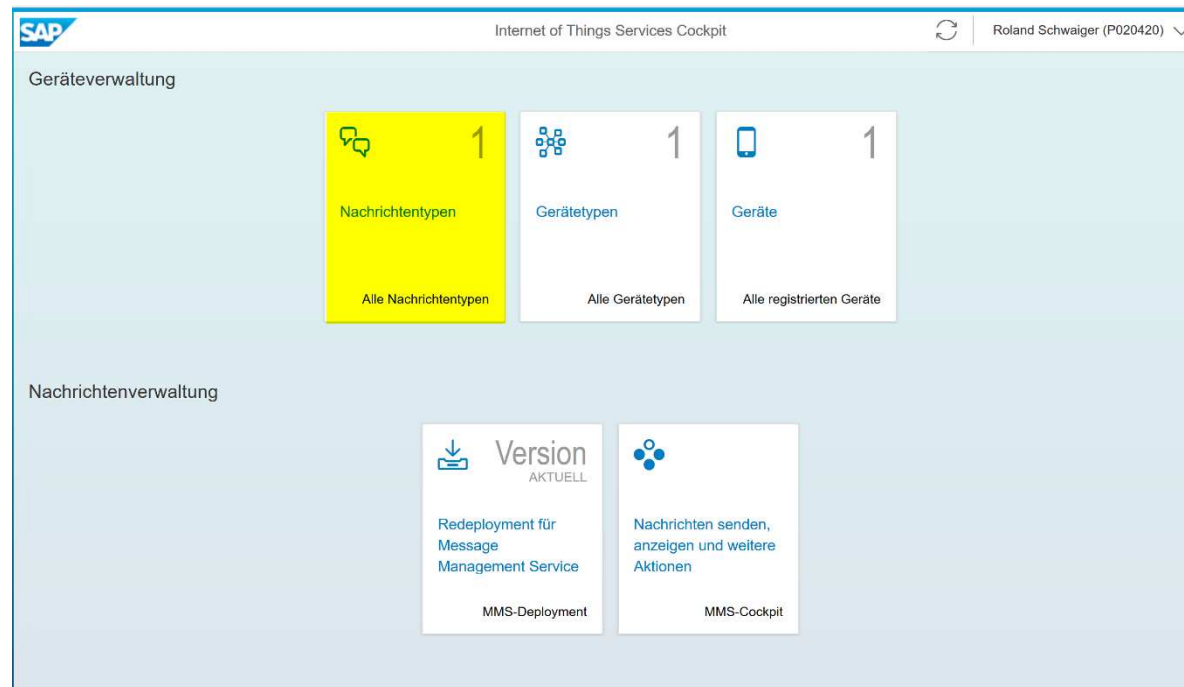
IoT-MMS-User Custom: Defined by the administrator

Individual Users Assign Unassign All

User ID
8820038
p000400
system

IoT Service Cockpit Message Type

- Hana Account -> Services -> IoT Service -> Go to Service



IoT Service Cockpit

Nachrichtentyp anlegen

Nachrichtentypen

Nachrichtentyp anlegen

Suchen

test

Informationen

*Name: honeycombmeasur

Felder

+ Feld hinzufügen

Position	Name	Typ
1	beekeeperid	string
		Max. Länge: 10
2	honeycombid	string
		Max. Länge: 4
3	measuredate	date

+ 1

Anlegen Abbrechen

IoT Service Cockpit

Nachrichtentyp anlegen

The screenshot shows the 'Nachrichtentypen' (Message Types) section of the IoT Service Cockpit. The main view is for the 'honeycombmeasure' type. It features a search bar with 'Suchen' and a refresh icon. Below the search bar, there are two tabs: 'Informationen' (Information) and 'Felder' (Fields), with 'Felder' currently selected. The 'Felder' tab displays a table of fields for the 'honeycombmeasure' type.

Position	Name	Typ	Optionale Einstellungen
1	beekeeperid	string	Max. Länge 10
2	honeycombid	string	Max. Länge 4
3	measuredate	date	
4	measuretime	date	
5	weight	double	
6	weightunit	string	Max. Länge 2
7	temperature	double	
8	temperatureunit	string	Max. Länge 2
9	beekeepername	string	Max. Länge 40

At the bottom of the interface, there is a dark bar with a plus sign (+), a warning icon with the number '1', and a 'Löschen' (Delete) button.

IoT Service Cockpit

Gerätetyp anlegen

< Gerätetypen

Suchen

testtyp

Gerätetyp anlegen

Informationen

*Name:

Weitere Details:

Nachrichtentypen

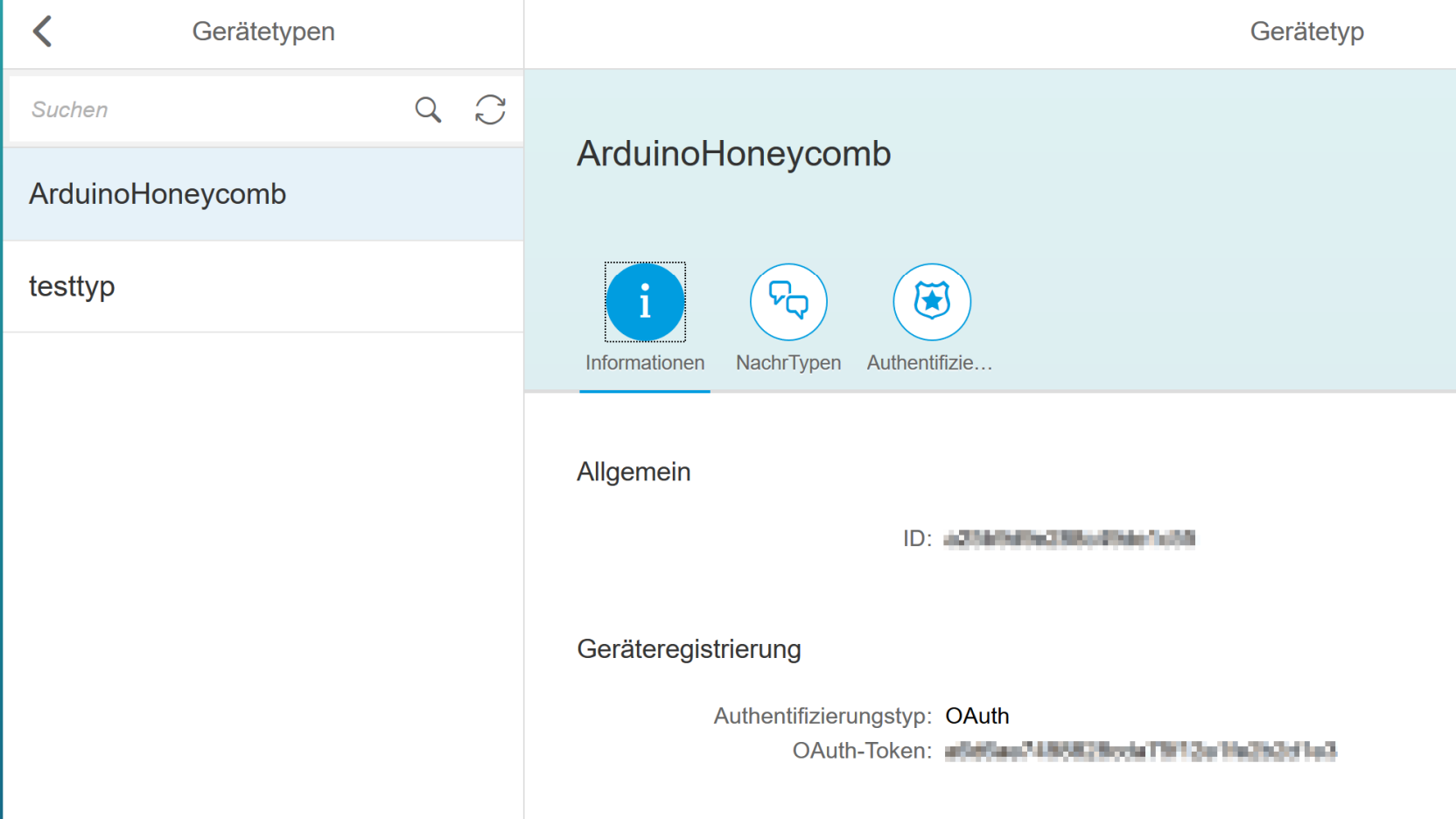
+ Nachrichtentyp hinzufügen

Zuordnungsname	Nachrichtentyp	Richtung	
<input type="text" value="Geben Sie einen Namen ein (optional)."/>	<input type="text" value="honeycombmeasure"/>	<input type="text" value="Von Gerät"/>	<input type="button" value="Löschen"/>

+

IoT Service Cockpit

Gerätetyp anlegen



Gerätetypen

Suchen

ArduinoHoneycomb

testtyp

Gerätetyp

ArduinoHoneycomb

Informationen

NachrTypen

Authentifizie...

Allgemein

ID: `...`

Geräteregistrierung

Authentifizierungstyp: OAuth

OAuth-Token: `...`

IoT Service Cockpit

Gerät anlegen

< Geräte Gerät anlegen

Suchen

testequi

Informationen

*Name:

*Gerätetyp:

Weitere Details:

Benutzerdefinierte Attribute

Schlüssel	Wert
Keine Daten	

IoT Service Cockpit

Gerät anlegen

Geräte

Gerät

Suchen

ArduinoHoneycomb1

testequi

ArduinoHoneycomb1

Informationen Authentifizie...

ID: e8278e4f53e4081f538f03218803e8f9

Gerätetyp: ArduinoHoneycomb

OAuth-Zugriffstoken

Neues OAuth-Zugriffstoken für Gerät "ArduinoHoneycomb1" angelegt.
Token: e8278e4f53e4081f538f03218803e8f9

Schließen

IoT Service Cockpit

Gerät anlegen

The screenshot displays the 'Geräte' (Devices) section of the IoT Service Cockpit. On the left, a list of devices is shown, including 'ArduinoHoneycomb1' and 'testequi'. The right pane shows the details for 'ArduinoHoneycomb1', including an 'Informationen' (Information) icon and an 'Authentifiziere...' (Authenticate...) icon. Below these icons, the device's ID, type, and authentication type are listed.

Geräte



Suchen 🔍 ↻

ArduinoHoneycomb1

testequi

Gerät

ArduinoHoneycomb1

Informationen Authentifiziere...

ID: cd5f2abd-6959-41da-b347-8071c4177bc9

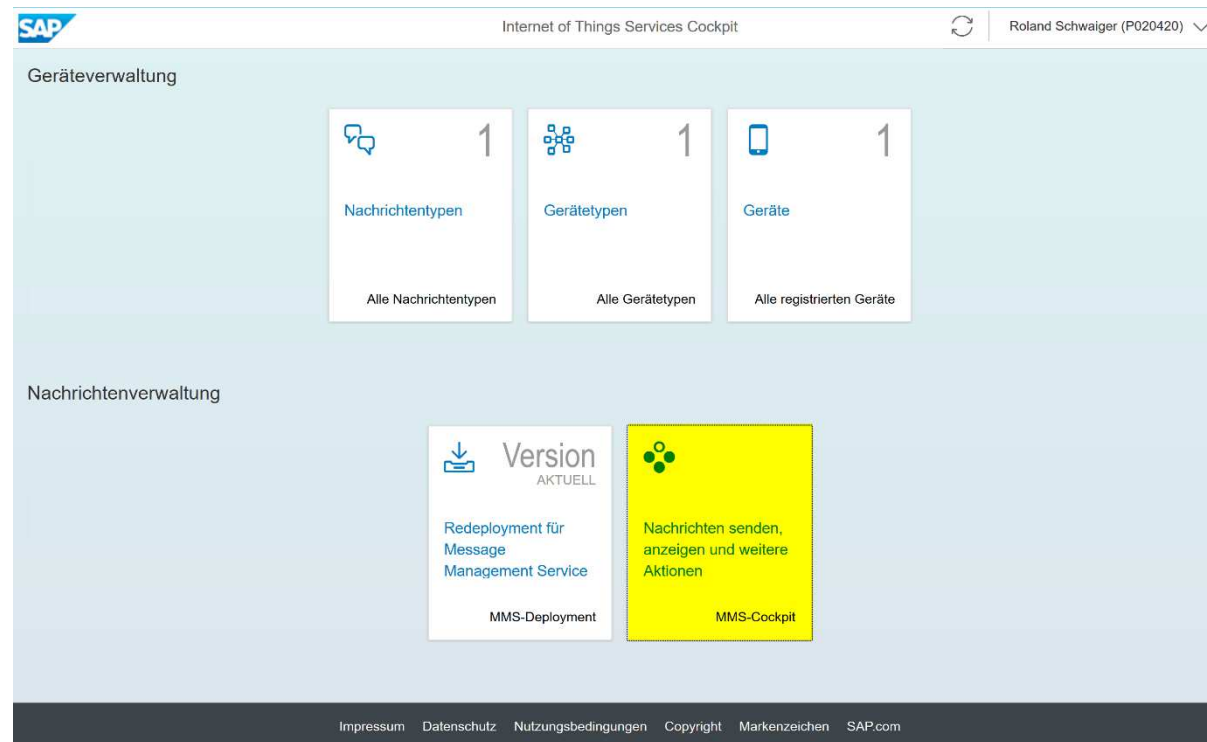
Gerätetyp: **ArduinoHoneycomb**

Authentifizierungstyp: OAuth

IoT Service Cockpit

Find Details for Arduino Sketch

- Hana Account -> Services -> IoT Service -> Go to Service



Message Management Service Cockpit

Find Details for Arduino Sketch

SAP Message Management Service Cockpit Roland Schwaiger

Zentrale Services

- Gespeicherte Nachrichten anzeigen**
Anwendungsdaten
- Nachrichten an Geräte senden**
Push-Service
- Message Management Service konfigurieren**
Konfiguration
- Zuordnungen für Verarbeitungsservices konfigurieren**
Konfiguration
- Registrierte Geräte und Gerätetypen anzeigen**
IoT Services Cockpit

Datenservices

- Datentransfer mittels HTTP**
HTTP-API
- Datentransfer mittels MQTT über TCP**
MQTT-TCP-API
- Datentransfer mittels WebSockets**
WebSocket-API
- Datentransfer mittels MQTT über WS**
MQTT-WebSocket-API

Impressum | Datenschutz | Nutzungsbedingungen | Copyright | Markenzeichen | SAP.com

Message Management Service Cockpit

Find Details for Arduino Sketch

SAP Message Management Service Cockpit Roland Schwaiger

Zentrale Services

- Gespeicherte Nachrichten anzeigen (Anwendungsdaten)
- Nachrichten an Geräte senden (Push-Service)
- Message Management Service konfigurieren (Konfiguration)
- Zuordnungen für Verarbeitungsservices konfigurieren (Konfiguration)
- Registrierte Geräte und Gerätetypen anzeigen (IoT Services Cockpit)

Datenservices

- Datentransfer mittels HTTP (HTTP-API)
- Datentransfer mittels MQTT über TCP (MQTT-TCP-API)
- Datentransfer mittels WebSockets (WebSocket-API)
- Datentransfer mittels MQTT über WS (MQTT-WebSocket-API)

Impressum Datenschutz Nutzungsbedingungen Copyright Markenzeichen SAP.com

HTTP-API Request

HTTP-API

Beschreibung

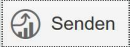
Dieser Client kann zum Testen der HTTP-API verwendet werden. Er kann Nachrichten an den Message Management Service senden, indem er HTTP-POST-Anfragen an einen gerätespezifischen HTTP-Datenendpunkt schickt. Außerdem kann der Client Bestätigungen empfangen, indem er HTTP-POST-Anfragen an einen gerätespezifischen HTTP-Bestätigungsendpunkt schickt.

Nachricht senden

Datenendpunkt:

Nachricht:
Nachr. Typ ID

GeräteID

 Senden

HTTP-API Response

Antwort vom Server

HTTP Code

 Tabelle leeren

Code	Nachricht
202	{"msg":"https://iotmms[REDACTED]trial.hanatrial.ondemand.com/com.sap.iotservices.mms/v1/api/http/ack/cd5f2abd-6959-41da-b347-8071c4177bc9/bc705193-5f3a-40c3-9a99-28a0d8509000","retryAfterHeaderValue":"60"}
200	{"msg":"1 message(s) received from device [cd5f2abd-6959-41da-b347-8071c4177bc9]"}

HTTP-API

- GeräteID
- NachrichtentypID
- Nachrichtentyp Felder
- Date = Timestamps and dates can be sent as ISO 8601 compatible string or in Unix time, represented as seconds since January 1, 1970. (For example 2016-01-14T08:00:00Z).
-

Test:

- <https://iotmms<User>trial.hanatrial.ondemand.com/com.sap.iotservices.mms/v1/api/http/da/ta/cd5f2abd-6959-41da-b347-8071c4177bc9>
- {"mode":"sync","messageType":"500bbdcbffc06ee45574","messages":[{"beekeeperid":"1","honeycombid":"1","measuredate":"2017-01-16T00:00:00Z","measuretime":",,1970-01-01T12:12:00Z","weight":"2.3","weightunit":"kg","temperature":"23.4","temperatureunit":"C","beekeepername":"hias"}]}

Response:

- 202
- {"msg":"https://iotmms<User>trial.hanatrial.ondemand.com/com.sap.iotservices.mms/v1/api/http/ack/cd5f2abd-6959-41da-b347-8071c4177bc9/bc705193-5f3a-40c3-9a99-28a0d8509000","retryAfterHeaderValue":"60"}

Zentrale Services

Gespeicherte
Nachrichten anzeigen

Anwendungsdaten

Nachrichten an
Geräte senden

Push-Service

Message
Management Service
konfigurieren

Konfiguration

Zuordnungen für
Verarbeitungsservices
konfigurieren

Konfiguration

Registrierte Geräte
und Gerätetypen
anzeigen

IoT Services Cockpit

Datenservices

Datentransfer mittels
HTTP

HTTP-API

Datentransfer mittels
MQTT über TCP

MQTT-TCP-API

Datentransfer mittels
WebSockets

WebSocket-API

Datentransfer mittels
MQTT über WS

MQTT-WebSocket-API

HTTP-API Test Result

← Anwendungsdaten

AKTUALISIEREN Letzte Aktualisierung am 16.1.2017, 12:44:46

4 Tabellen 🔗 OData-API

📊 T_IOT_500BBDCBFFC06EE45574 NEO_6N7D9ONIZ0G4J9N48R4DK3H3U	1 >
📊 T_IOT_ACKSTORE NEO_6N7D9ONIZ0G4J9N48R4DK3H3U	1 >

Received but falsy

← Anwendungsdaten

AKTUALISIEREN Letzte Aktualisierung am 16.1.2017, 12:46:30

Tabelle NEO_6N7D9ONIZ0G4J9N48R4DK3H3U.T_IOT_ACKSTORE 🔗 OData-API
(1 Zeile(n) von 1 geladen. Die neuesten Einträge kommen zuerst.)

C_DEVICEID	C_SEQUENCEID	C_STATUS	C_PHASE	C_MSG	G_CREATED
cd5f2abd-6959-41da-b347-8071c4177bc9	bc705193-5f3a-40c3-9a99-28a0d8509000	VALIDATION_FAILED	3	Parse error. Expected [measuretime] of type [DATE] but received ["0000-00-00T12:12:00Z"]	Mon Jan 16 2017 12:16:31 GMT+0100

HTTP-API Test Result

← Anwendungsdaten

AKTUALISIEREN Letzte Aktualisierung am 16.1.2017, 12:44:46

4 Tabellen 🔗 OData-API

📊 T_IOT_500BBDCBFFC06EE45574 NEO_6N7D9ONIZ0G4J9N48R4DK3H3U	1 >
📊 T_IOT_ACKSTORE NEO_6N7D9ONIZ0G4J9N48R4DK3H3U	1 >

Received OK

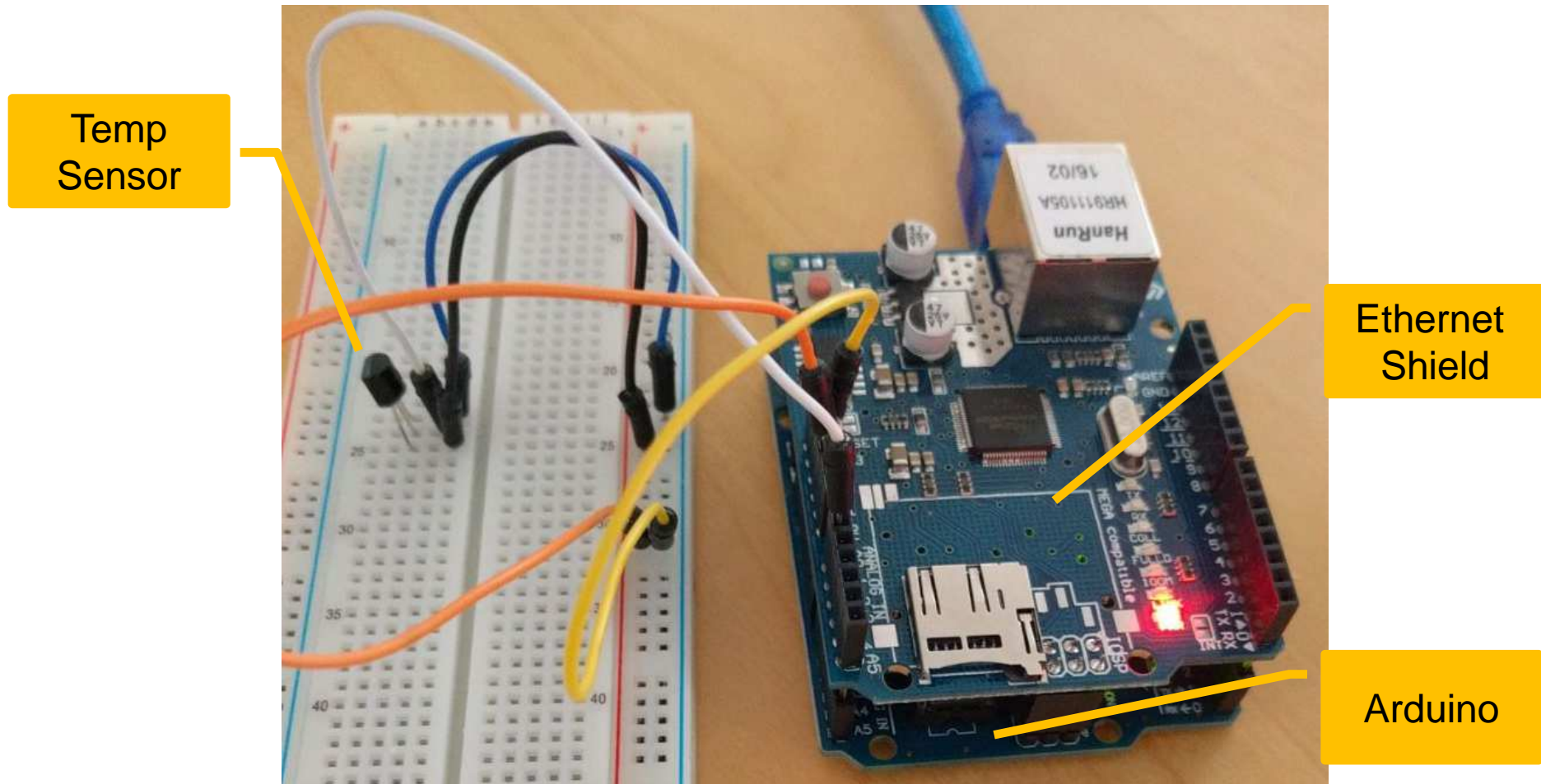
← Anwendungsdaten

AKTUALISIEREN Letzte Aktualisierung am 16.1.2017, 12:47:33

Tabelle NEO_6N7D9ONIZ0G4J9N48R4DK3H3U.T_IOT_500BBDCBFFC06EE45574
(1 Zeile(n) von 1 geladen. Die neuesten Einträge kommen zuerst.) 🔗 OData-API

G_DEVICE	G_CREATED	C_BEEKEEPERID	C_HONEYCOMBID	C_MEASUREDATE	C_MEASURETIME	C_WEIGHT	C_WEIGHTUNIT	C_TEMPERATURE	C_TEMPERATUREUNIT	C_BEEKEEPERNAME
cd5f2abd-6959-41da-b347-8071c4177bc9	Mon Jan 16 2017 12:43:00 GMT+0100	1	1	Mon Jan 16 2017 01:00:00 GMT+0100	Thu Jan 01 1970 13:12:00 GMT+0100	2.3	kg	23.4	C	hias

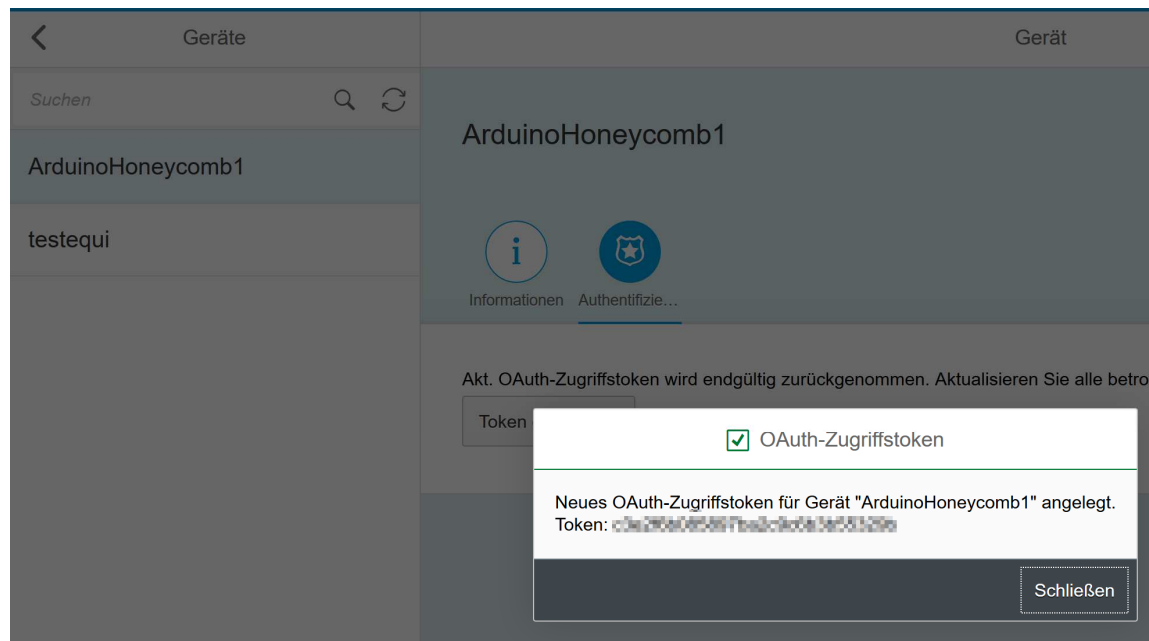
Arduino Hardware



Arduino

Sketch - Preparation

- OAuthToken: Internet of Things Services Cockpit -> Geräte -> <Gerät> -> Authentifizieren -> Token generieren



Arduino Sketch

HONEYCOMB

A smart example for IoT and SAP HANA

Honeycomb Package

- SAP HANA Web-based Development Workbench: Editor



Honeycomb XSC Application

Create Application [X]

Template: [v]

Package:

[Create] [Cancel]

Create Application [X]

Template: [v]

Package:

[Create] [Cancel]

Honeycomb

SAPUI5 - View

- Create a view `App.view.xml`

Conventions

- View names are capitalized
- All views are stored in the view folder
- All controller are stored in the controller folder
- Names of XML views always end with `*.view.xml`
- The default XML namespace is `sap.m`
- Other XML namespaces use the last part of the SAP namespace as alias (for example, `mvc` for `sap.ui.core.mvc`)

Honeycomb

SAPUI5 - View

```
<?xml version=1.0 encoding=UTF-8?>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv=X-UA-Compatible content=IE=edge />
    <meta charset=utf-8 />
    <title>Honeycomb</title>
    <!-- 1.) Load SAPUI5 (from local server), select theme and control library -->
    <script id=sap-ui-bootstrap type=text/javascript src=/sap/ui5/1/resources/sap-ui-core.js data-sap-ui-
theme=sap_bluecrystal data-sap-ui-libs=sap.m data-sap-ui-compatVersion=edge data-sap-ui-preload=async
data-sap-ui-resourceroots={ at.facet.8820038.honeycomb: ./ } />
    <script>sap.ui.getCore().attachInit(function () {
      sap.ui.xmlview({
        viewName : at.facet.8820038.honeycomb.view.App
      }).placeAt(content);
    });</script>
  </head>
  <body class=sapUiBody>
    <!-- This is where you place the UI5 -->
    <div id=content />
  </body>
</html>
```

Honeycomb

Pitfall index.html

- Resource path from Web

src=[/sap/ui5/1/resources/sap-ui-core.js](#)

- Resource path from Web IDE

src=resources/sap-ui-core.js

- Alternative:

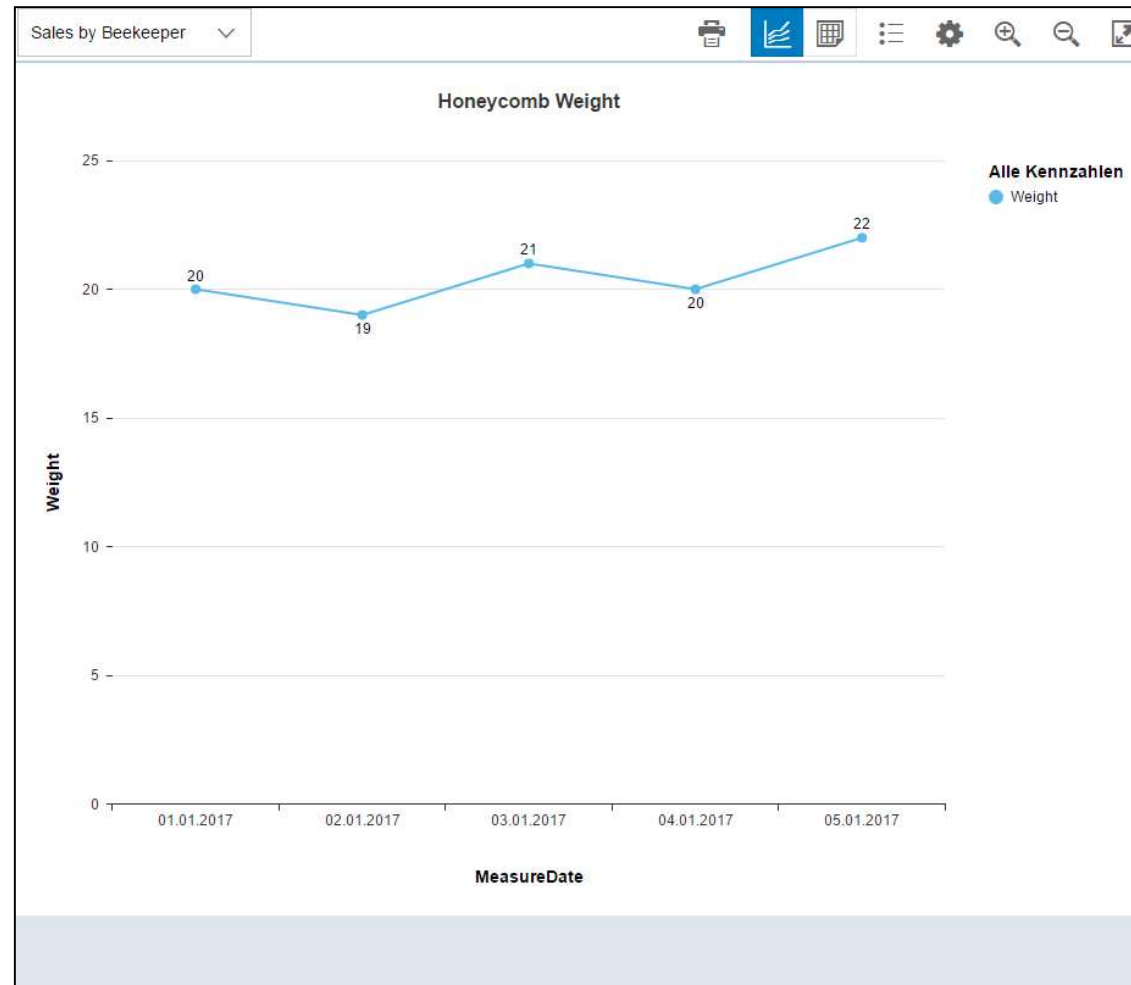
src=<https://sapui5.hana.ondemand.com/1.30.6/resources/sap-ui-core.js>

Honeycomb

SAPUI5 - View

- Now take the code from <https://sapui5.netweaver.ondemand.com/sdk/explored.html#/sample/sap.suite.ui.commons.sample.ChartContainerFixFlexLayout/code>

Honeycomb Graph



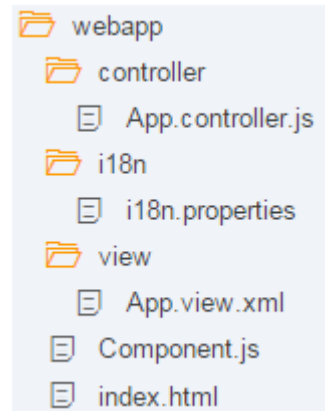
Honeycomb Table

Sales by Beekeeper ▼						
Beekeeper Name	Beekeeper ID	Honeycomb ID	Measure Date	Measure Time	Weight	Temperature
Imker Lois	1	1	01.01.2017	10:20:00	20.0	25.1
Imker Lois	1	1	02.01.2017	10:20:00	19.0	25.1
Imker Lois	1	1	03.01.2017	10:20:00	21.0	25.1
Imker Lois	1	1	04.01.2017	10:20:00	20.0	25.1
Imker Lois	1	1	05.01.2017	10:20:00	22.0	25.1

Honeycomb Refactor

- TODO: Follow the tutorial

Honeycomb Refactor



Honeycomb

manifest.json Conventions

- The descriptor file is named manifest.json and located in the webapp folder.
- Use translatable strings for the title and the description of the app.

Honeycomb Result

