



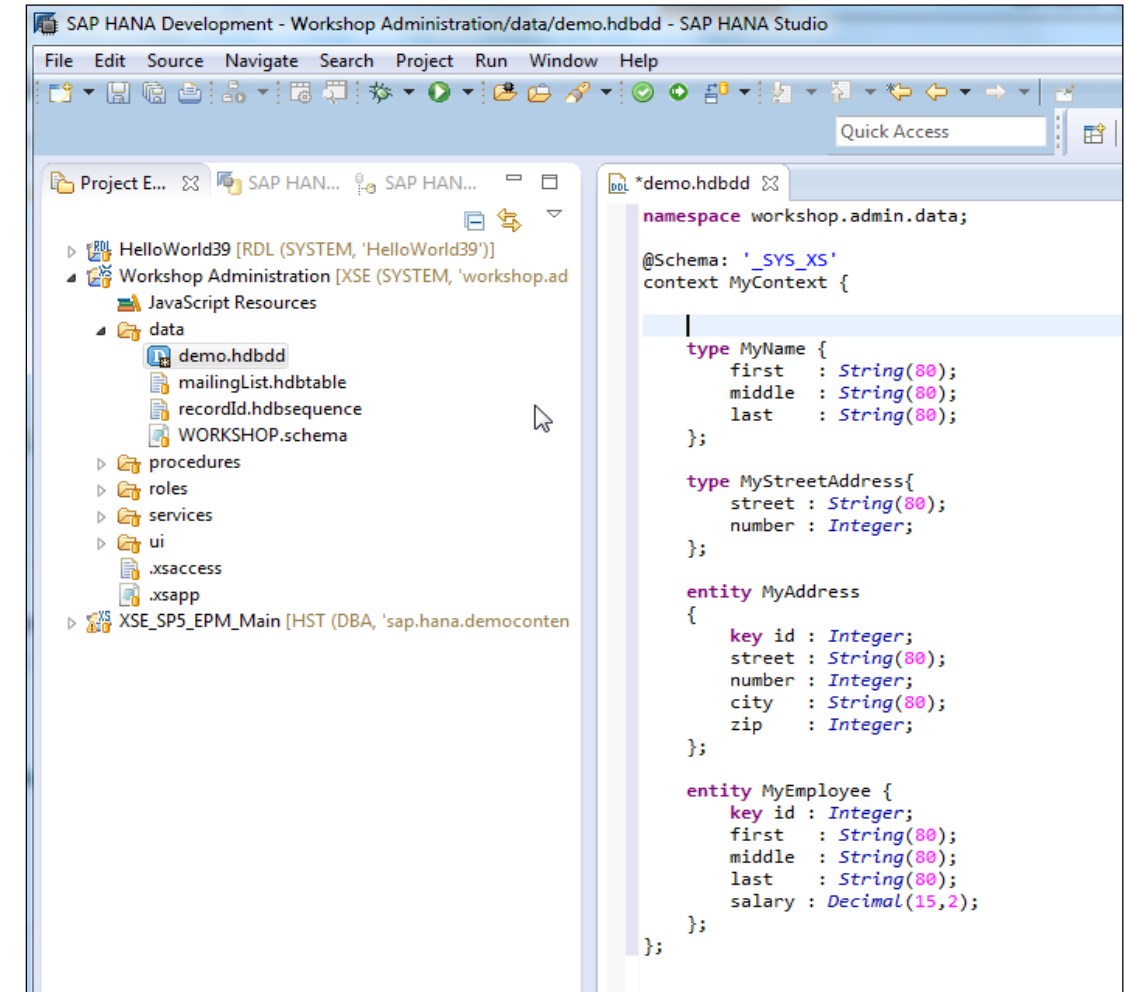
Week 2: New Programming Model

Unit 2: Evolution of Core Data Services (CDS)

Evolution of Core Data Services (CDS)

History – hdbdd since SAP HANA 1.0 SPS 06

- Development artifact extension – **hdbdd** (data dictionary)
- Defines the schema to use, reusable types, multiple entities (tables), and views all within one source file
- Multiple catalog objects are generated upon activation



Evolution of Core Data Services (CDS)

History – hdbdd since SAP HANA 1.0 SPS 06: main features

- Associations define relationships between entities (tables)
- Integrated into the database/SQL but can also 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;
```

Evolution of Core Data Services (CDS)

History – hdbdd since SAP HANA 1.0 SPS 06: main features

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

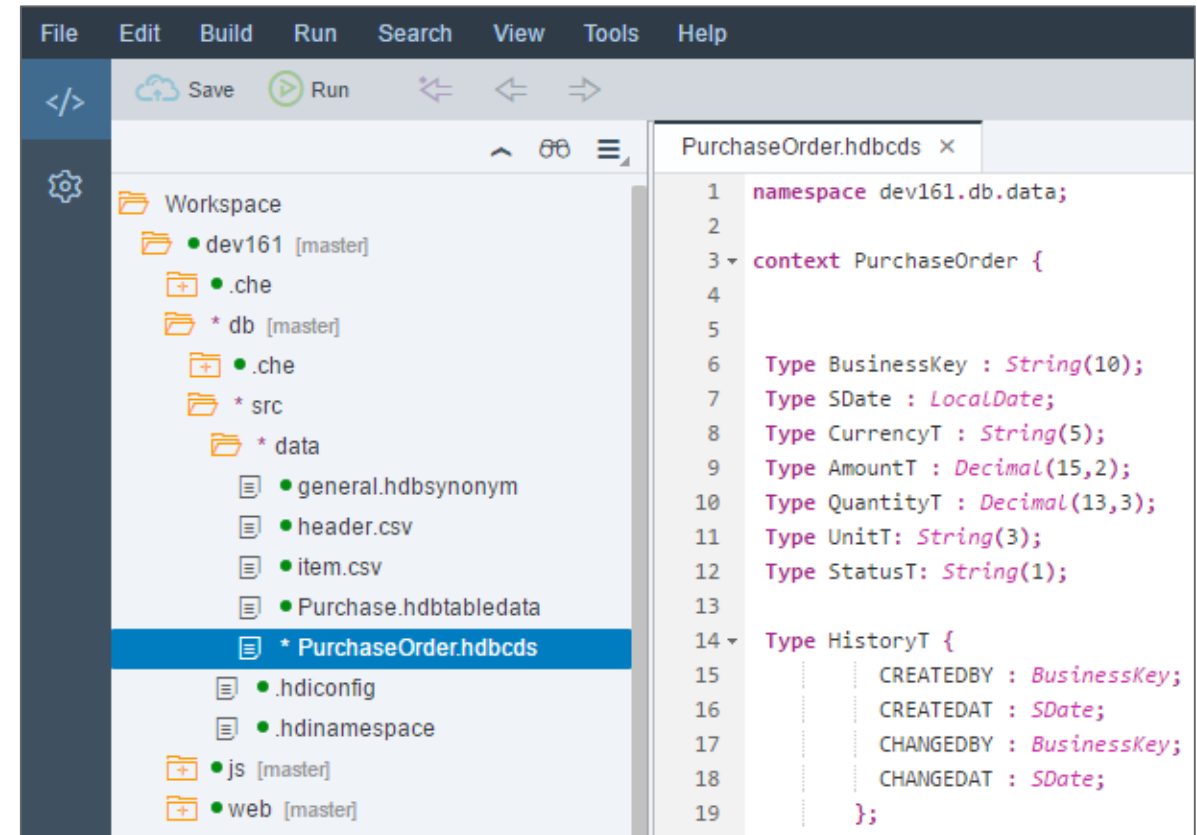
```
define view EmployeesView as SELECT from Employee {  
    /*key*/ ID, name,  
    orgunit.name as orgunitname  
};
```

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

Evolution of Core Data Services (CDS)

History – hdbcds since SAP HANA 1.0 SPS 11 within HDI

- Development artifact extension in SAP HANA SPS 06 – **hdbdd** (data dictionary) – changed to **hdbcds** in HDI with SPS 11
- Defines reusable types, multiple entities (tables), and views all within one source file
- Multiple catalog objects are generated upon build



```
1 namespace dev161.db.data;
2
3 context PurchaseOrder {
4
5
6   Type BusinessKey : String(10);
7   Type SDate : LocalDate;
8   Type CurrencyT : String(5);
9   Type AmountT : Decimal(15,2);
10  Type QuantityT : Decimal(13,3);
11  Type UnitT : String(3);
12  Type StatusT : String(1);
13
14  Type HistoryT {
15      CREATEDBY : BusinessKey;
16      CREATEDAT : SDate;
17      CHANGEDBY : BusinessKey;
18      CHANGEDAT : SDate;
19  };
```

Evolution of Core Data Services (CDS)

History – hdbcds since SAP HANA 1.0 SPS 11 within HDI

- In general, hdbcds works in HDI like it did in SAP HANA Repository. There are some incompatible changes, however.

The following annotations are not supported in HDI:

@Schema

@Catalog (index and tableType)

@SearchIndex

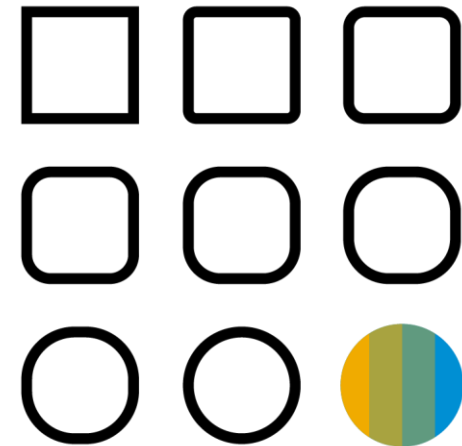
@SearchIndexes

@nokey

@GenerateTableType

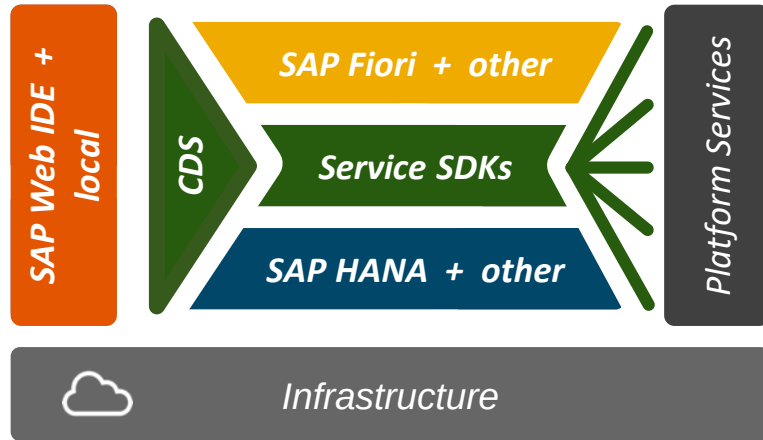
@WithStructuredPrivilegeCheck

- For all annotations that have been replaced by regular syntax in HDI, the syntax variant is also available and recommended in Repository.



Evolution of Core Data Services (CDS)

CDS usage within the new application programming model

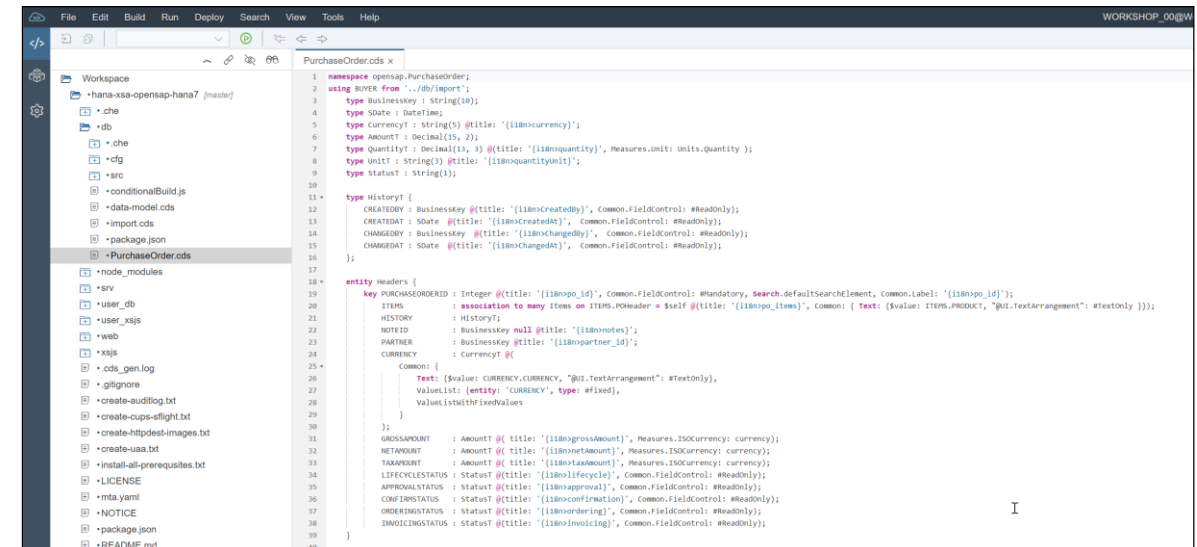


Task	Using	And let the framework take care of...
Define Data Models	CDS	- Mapping to physical schemas (e.g. SQL) - Deployment to SAP HANA + others - Foreign keys and joins - Localized data - Extensibility - Language-embedded data access
Define Services	CDS	- Model translation (e.g. to OData, Open API) - Serving requests out-of-the-box - Standard validations - Auto-filling audit data - Trace and audit logging - Authorization enforcement
Add Custom Logic	Service SDKs for Java or Node.js	- Multitenancy and isolation - Connection pooling - Validating input - Message handling - Serializing responses - Platform Integration
Consume Services (e.g. from S/4)		- Providing APIs in Java / Node.js - Mashups with local data - Handling http requests - Resilience, caching, ...
Add UIs	CDS Annotations	- Translating to OData vocabularies - Generating SAP Fiori app skeletons

Evolution of Core Data Services (CDS)

History – New CDS artifact as of SAP HANA 2.0 SPS 03 (October 2018 revision)

- Development artifact extension in SAP HANA SPS 06 – **hdbdd** (data dictionary) – changed to **hdbcds** in HDI with SPS 11 – changed again to **cds** as part of the new application programming model
- Still defines reusable types, multiple entities (tables), and views all within one source file but with modified, database-agnostic syntax
- You build **cds** artifacts on the “client side” within the development tools, not within the SAP HANA server
- When you build **cds** artifacts, database-specific design time objects are created



Evolution of Core Data Services (CDS)

CDS: user-defined types

- Individual types as well as structured types
- A structured type is a data type comprising a list of attributes, each of which has its own data type
- The attributes of the structured type can be defined manually in the structured type itself, and reused either by another structured type or an entity

```
PurchaseOrder.cds x
1 namespace opensap.PurchaseOrder;
2 using BUYER from '../db/import';
3 type BusinessKey : String(10);
4 type SDate : DateTime;
5 type CurrencyT : String(5) @title: '{i18n>currency}';
6 type AmountT : Decimal(15, 2);
7 type QuantityT : Decimal(13, 3) @(title: '{i18n>quantity}', Measures.Unit: Units.Quantity );
8 type UnitT : String(3) @title: '{i18n>quantityUnit}';
9 type StatusT : String(1);
10
11 type HistoryT {
12     CREATEDBY : BusinessKey @(title: '{i18n>CreatedBy}', Common.FieldControl: #ReadOnly);
13     CREATEDAT : SDate @(title: '{i18n>CreatedAt}', Common.FieldControl: #ReadOnly);
14     CHANGEDBY : BusinessKey @(title: '{i18n>ChangedBy}', Common.FieldControl: #ReadOnly);
15     CHANGEDAT : SDate @(title: '{i18n>ChangedAt}', Common.FieldControl: #ReadOnly);
16 };
17
```

Evolution of Core Data Services (CDS)

CDS: entities and associations

- An entity is the core artifact for persistence-model definition using the CDS syntax
- A CDS entity is a table with a set of data elements that are organized using columns and rows
- Associations define relationships between entities (tables)
- Associations are specified by adding an element to a source entity with an association type that points to a target entity, complemented by optional information defining cardinality and which keys to use

```
18 entity Headers {
19   key PURCHASEORDERID : Integer @(title: '{i18n>po_id}', Common.FieldControl: #Mandatory, Search.defaultSearchElement, Common.Label: '{i18n>po_id}');
20   ITEMS                : association to many Items on ITEMS.POHeader = $self @(title: '{i18n>po_items}', Common: { Text: {$value: ITEMS.PRODUCT, "@UI.TextArrangement": #TextOnly }});
21   HISTORY              : HistoryT;
22   NOTEID               : BusinessKey null @(title: '{i18n>notes}');
23   PARTNER              : BusinessKey @(title: '{i18n>partner_id}');
24   CURRENCY             : CurrencyT @(
25     Common: {
26       Text: {$value: CURRENCY.CURRENCY, "@UI.TextArrangement": #TextOnly},
27       ValueList: {entity: 'CURRENCY', type: #fixed},
28       ValueListWithFixedValues
29     }
30   );
31   GROSSAMOUNT          : AmountT @( title: '{i18n>grossAmount}', Measures.ISOCurrency: currency);
32   NETAMOUNT            : AmountT @( title: '{i18n>netAmount}', Measures.ISOCurrency: currency);
33   TAXAMOUNT            : AmountT @( title: '{i18n>taxAmount}', Measures.ISOCurrency: currency);
34   LIFECYCLESTATUS      : StatusT @(title: '{i18n>lifecycle}', Common.FieldControl: #ReadOnly);
35   APPROVALSTATUS       : StatusT @(title: '{i18n>approval}', Common.FieldControl: #ReadOnly);
36   CONFIRMSTATUS        : StatusT @(title: '{i18n>confirmation}', Common.FieldControl: #ReadOnly);
37   ORDERINGSTATUS       : StatusT @(title: '{i18n>ordering}', Common.FieldControl: #ReadOnly);
38   INVOICINGSTATUS      : StatusT @(title: '{i18n>invoicing}', Common.FieldControl: #ReadOnly);
39 }
40
41
42 entity Items {
43   key POHeader          : association to Headers @title: '{i18n>poService}';
44   key PRODUCT           : BusinessKey @(title: '{i18n>product}', Common.FieldControl: #Mandatory, Search.defaultSearchElement);
45   NOTEID               : BusinessKey null ;
46   CURRENCY             : CurrencyT @(
47     Common: {
48       Text: {$value: CURRENCY.CURRENCY, "@UI.TextArrangement": #TextOnly},
49       ValueList: {entity: 'CURRENCY', type: #fixed},
50       ValueListWithFixedValues
51     }
52   );
53   GROSSAMOUNT          : AmountT @( title: '{i18n>grossAmount}', Measures.ISOCurrency: currency);
54   NETAMOUNT            : AmountT @( title: '{i18n>netAmount}', Measures.ISOCurrency: currency);
55   TAXAMOUNT            : AmountT @( title: '{i18n>taxAmount}', Measures.ISOCurrency: currency);
56   QUANTITY             : QuantityT;
57   QUANTITYUNIT          : UnitT;
58   DELIVERYDATE         : SDate @(title: '{i18n>deliveryDate}');
59 }
```

Evolution of Core Data Services (CDS)

CDS: views

- CDS syntax expands to include the definition of views
- A view is a virtual table, not persisted, based on the dynamic results returned in response to a SQL statement
- Views can utilize the defined associations between entities
- Supports the use of aggregation and filtering

```
61      define view ItemView as
62      select from Items
63      {
64          POHeader.PURCHASEORDERID,
65          POHeader.PARTNER,
66          PRODUCT,
67          CURRENCY,
68          GROSSAMOUNT,
69          NETAMOUNT,
70          TAXAMOUNT,
71          QUANTITY,
72          QUANTITYUNIT,
73          DELIVERYDATE
74      };
75
```

Evolution of Core Data Services (CDS)

CDS: services and UI annotations

- CDS can also be used to define services and UI annotations
- We will go into this in much more depth in later units of this week

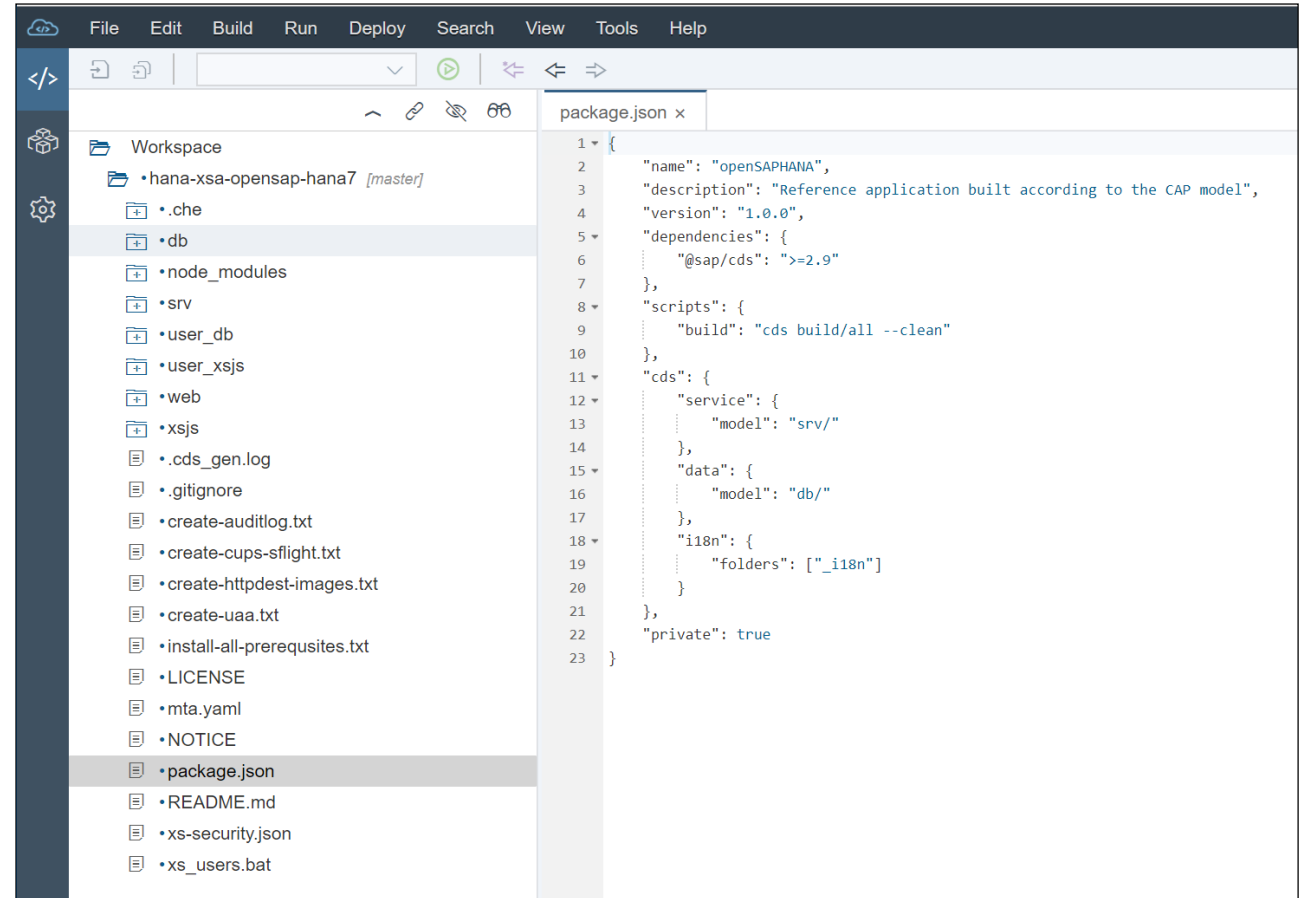
```
*my-service.cds x
1  using PURCHASE_ORDER_ITEM_VIEW as ItemsView from '../db/data-model';
2
3  service CatalogService {
4
5  extend ItemsView with {
6      PARTNERS: association to BuyerView on PARTNERS.PARTNERID = PARTNER_ID;
7  }
8
9  entity POItemsView @(
10     title: '{i18n>poService}',
11     Capabilities: {
12         InsertRestrictions: {Insertable: false},
13         UpdateRestrictions: {Updatable: false},
14         DeleteRestrictions: {Deletable: false}
15     }
16     ) as projection on ItemsView;
17 }
```

```
fiori-annotations-my-service.ods x
1  using CatalogService as cats from './my-service';
2
3  annotate cats.POItemsView with @( // header-level annotations
4  // -----
5  // List Report
6  // -----
7  // Product List
8  UI: {
9      ListItem: [
10         { $Type: 'UI.DataField', Value: PO_ITEM_ID, "@UI.Importance": #High },
11         { $Type: 'UI.DataField', Value: PRODUCT_ID, "@UI.Importance": #High },
12         { $Type: 'UI.DataField', Value: PARTNER_ID, "@UI.Importance": #High },
13         { $Type: 'UI.DataField', Value: PARTNERS.COMPANYNAME, "@UI.Importance": #Medium },
14         { $Type: 'UI.DataField', Value: AMOUNT, "@UI.Importance": #High },
15         { $Type: 'UI.DataField', Value: CURRENCY_CODE, "@UI.Importance": #Medium },
16     ],
17     PresentationVariant: {
18         SortOrder: [ { $Type: 'Common.SortOrderType', Property: PO_ITEM_ID, Descending: false }, { $Type: 'Common.SortOrderType', Property: PRODUCT_ID, Descending: false } ]
19     }
20 }
21
22 );
23
24
```

Evolution of Core Data Services (CDS)

CDS: package.json in project root

- New package.json in the project root that controls CDS settings cross-module
- Dependencies section sets overall CDS version (from NPM) and can also reference foreign cds files from other projects
- Other settings such as the service and database model module names

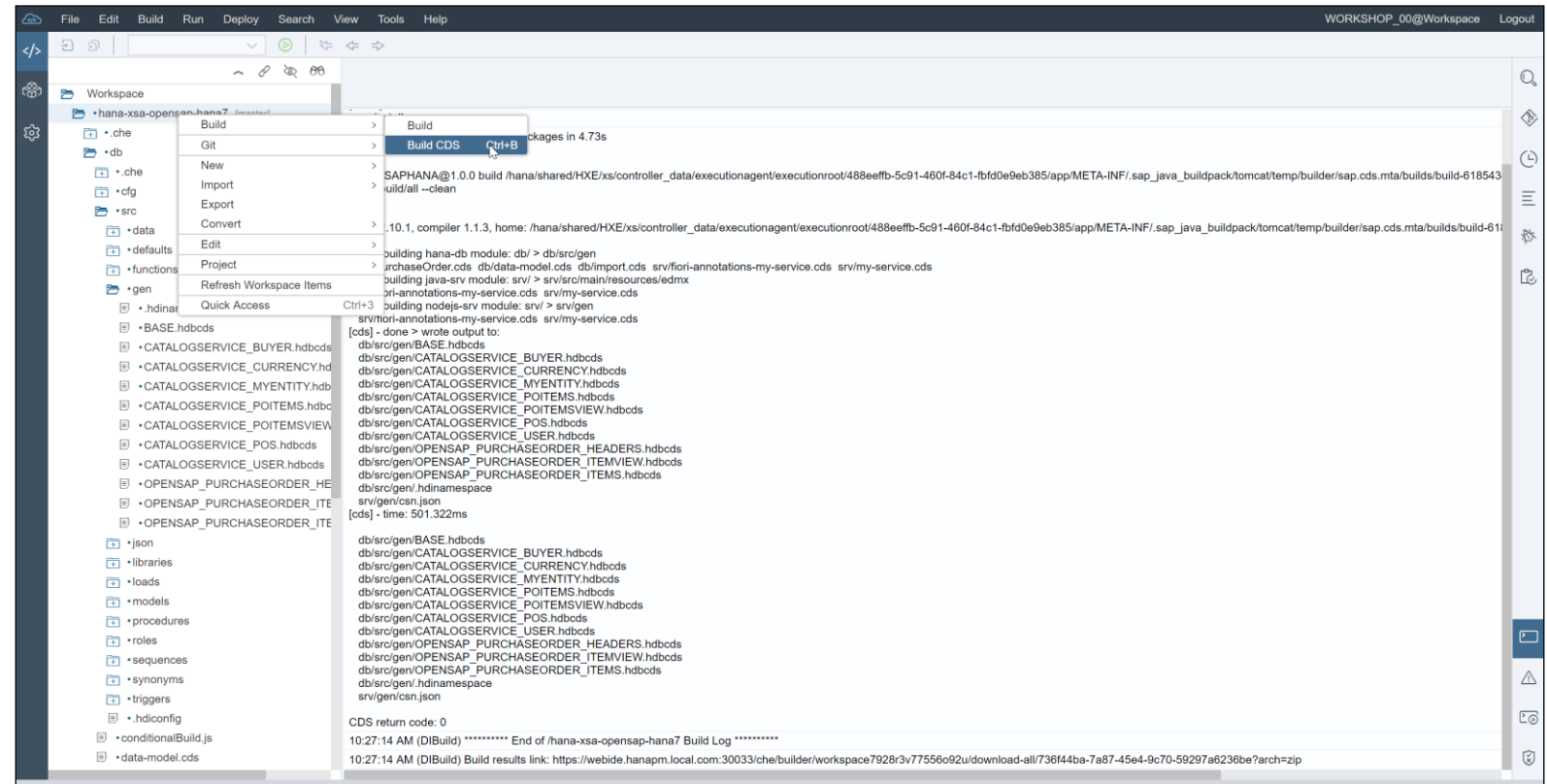


```
1 {
2   "name": "openSAPHANA",
3   "description": "Reference application built according to the CAP model",
4   "version": "1.0.0",
5   "dependencies": {
6     "@sap/cds": ">=2.9"
7   },
8   "scripts": {
9     "build": "cds build/all --clean"
10  },
11  "cds": {
12    "service": {
13      "model": "srv/"
14    },
15    "data": {
16      "model": "db/"
17    },
18    "i18n": {
19      "folders": ["_i18n"]
20    }
21  },
22  "private": true
23 }
```

Evolution of Core Data Services (CDS)

CDS build

- You build **cds** artifacts on the “client side” within the development tools, not within the SAP HANA server
- Currently, build generates **hdbcds** artifacts but in the future will generate directly to **hdbtable/hdbview**
- Builds of **cds** can be done for other target databases (for example, local testing in SQLite)



Thank you.

Contact information:

open@sap.com

Follow all of SAP



www.sap.com/contactsap

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platforms, directions, and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

See www.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.