LOGALI

Teoría

CDS – Tipos y Enumeraciones

ABAP Cloud – Modelado con CDS

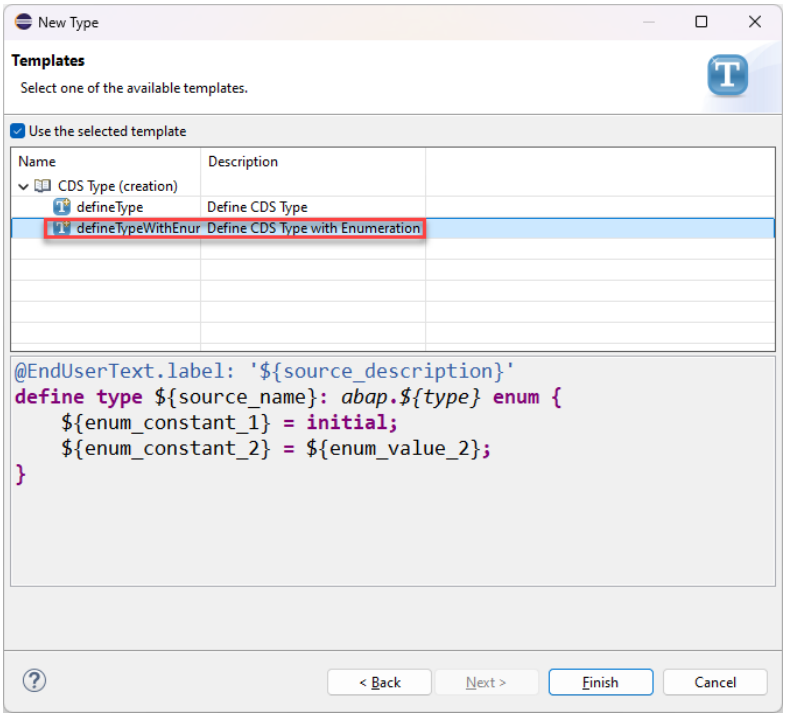
Contenido

14. CDS – Tipos y Enumeraciones

14.1. Enum Type – Definición

Los tipos enum en Core Data Services (CDS) de SAP permite establecer un conjunto de valores predefinidos para un campo, garantizando que solo se puedan asignar estos valores permitidos. Esta funcionalidad es similar a la de los dominios en el diccionario de datos, proporcionando consistencia y simplificando el mantenimiento de los datos. Los tipos enum en CDS se definen con restricciones específicas, como la necesidad de un valor inicial obligatorio y la uniformidad en los tipos de datos utilizados. Esta técnica mejora la calidad y fiabilidad de los datos en aplicaciones empresariales complejas al evitar la asignación de valores no válidos.

Para crear un tipo enum, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Type** y seleccionar la plantilla **defineTypeWithEnum** que se encuentra en la carpeta **CDS Type (creation).**



Puntos importantes a considerar:

- **Tipos Predefinidos**: Los tipos permitidos para la enumeración incluye char (con una longitud máxima de 8), y enteros (int1, int2, int3, int4). No se permite el uso de tipos más grandes como int8.
- **Valor Inicial Obligatorio:** Es obligatorio definir un valor inicial utilizando la palabra reservada **initial**. Este valor debe ser el primer valor en la lista de posibles valores de la enumeración.
- **Buenas Prácticas:** Aunque se puede definir el valor inicial en cualquier posición, se recomienda por buenas prácticas definirlo en la primera posición
- **Tipos Uniformes**: Todos los valores definidos en la enumeración deben ser del mismo tipo especificado al principio de la definición. No se permite mezclar tipos diferentes dentro de una misma enumeración.
- **Texto descriptivo :** Es posible agregar un texto descriptivo a los elementos del tipo a través de la anotación **@EndUserText.label: 'label'.**

Beneficios:

- **Consistencia:** Garantiza que solo se asignen valores permitidos a los campos, eliminando errores de datos.
- **Mantenimiento Simplificado:** Facilita el mantenimiento al centralizar la definición de valores permitidos en un solo lugar.

Sintaxis:

```
@EndUserText.label: 'Type - enum test'
define type enum_type_name : abap.type() enum {
@EndUserText.label: 'label'
  enum_constant_1 = initial;
@EndUserText.label: 'label'
  enum_constant_2 = enum_value_2;
}
```

14.2. Enum Type – Uso

Los tipos de enumeración (enum) en CDS permiten definir un conjunto de valores predefinidos para un campo, asegurando que solo se puedan asignar estos valores permitidos. Esta característica es similar a los dominios en el diccionario de datos, pero se maneja directamente en CDS. Los enums son especialmente útiles para garantizar la validez y consistencia de los datos en diversos escenarios empresariales.

Implementación de los tipos enum:

Una vez definido, el tipo enum está disponible globalmente en el servidor de aplicaciones y se pueden utilizar en diversos elementos SAP como:

- **Entidades CDS:** Al proyectar columnas en vistas CDS. En este caso es necesario colocar el nombre del tipo enum seguido de punto (.) y se desplegará una lista con los campos posibles a utilizar definidos en el tipo. Con la diferencia que empezaran con el símbolo # seguido del nombre de la opción debido a la sintaxis en el uso dentro de las entidades CDS.

  Ejemplo:

```
@AbapCatalog.viewEnhancementCategory: [#]
@AccessControl.authorizationCheck: #
@EndUserText.label: ''
@Metadata.ignorePropagatedAnnotations: true/false
@ObjectModel.usageType:{
        serviceQuality: #,
       sizeCategory: #,
       dataClass: #
}
define view entity entity_name as select from table_name

{

       components,

       component as alias,

       zenum_type.#Option as AliasName

}
```

- **Filtrado**: Es posible filtrar los datos en las vistas CDS utilizando los valores del enum. Sin embargo, debido a algunas restricciones, es necesario utilizar el casting para asegurar que los tipos sean compatibles.

  Ejemplo:

```
@AbapCatalog.viewEnhancementCategory: [#]
@AccessControl.authorizationCheck: #
@EndUserText.label: ''
@Metadata.ignorePropagatedAnnotations: true/false
@ObjectModel.usageType:{
        serviceQuality: #,
       sizeCategory: #,
       dataClass: #
}
define view entity entity_name as select from table_name

{

       components,

       component as alias,

       zenum_type.#Option as AliasName

}

where

component  = cast((zenum_type.#Option as abap.type() ));
```

- **Código Abap:** para acceder a los valores del tipo enum es necesario colocar el nombre del tipo seguido de guión (-) seguido del nombre de la opción con el valor requerido.

  Ejemplo:

```
lv_variable = zenum_type-option.
```

Restricciones y Buenas Prácticas:

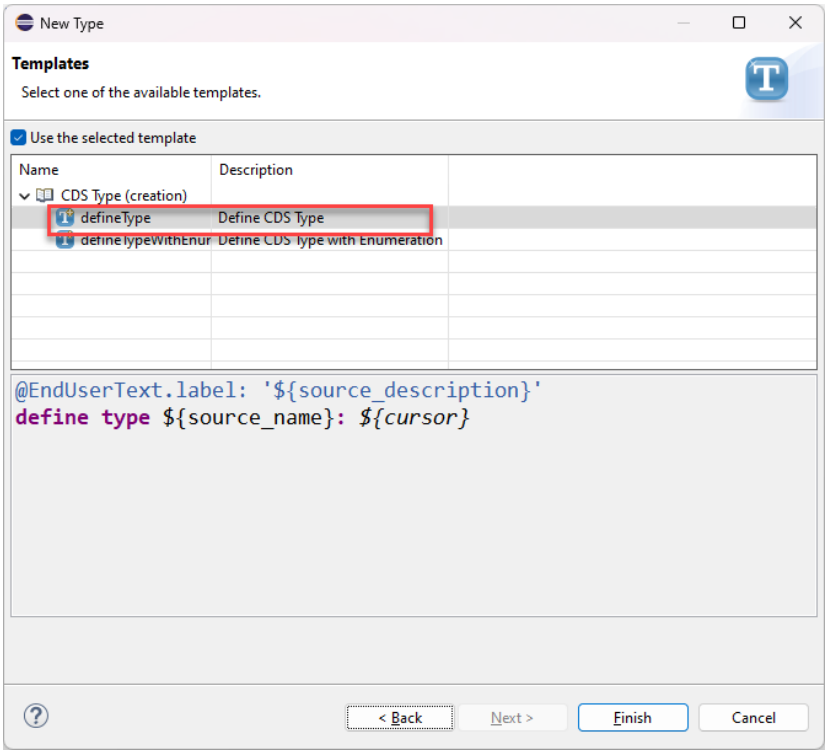
- **Tablas de base de datos:** En tablas de base de datos, no es posible el uso directo de los tipos enum. Aunque como dato adicional **los define table entity** (Recordando que las tablas de base de datos se declaran con la instrucción define table)**,** ubicados en la carpeta core data service, los cuales son concepto que se estudiará en otros módulos son compatibles con el uso de los tipos enum para la asignación para las columnas.
- **Limitaciones en Vistas de Proyección:** En las vistas de proyección, no es posible el uso directo de los tipos enum.
- **Inicialización y Valor Inicial:** Es obligatorio definir un valor inicial utilizando la palabra reservada **initial**. Este valor debe ser el primer valor en la lista de posibles valores de la enumeración.
- **Compatibilidad de Tipos:** Los tipos definidos en la enumeración deben coincidir con el tipo de datos de las columnas en las vistas CDS. En algunos casos, puede ser necesario aplicar un casting para garantizar la compatibilidad.

14.3. CDS Type

La definición de tipos CDS permite centralizar y tipos de datos, mejorando la coherencia y la mantenibilidad de los modelos de datos. Estos tipos pueden incluir tipos predefinidos, elementos de datos del diccionario de datos, y tipos de enumeración (enum) que restringen los valores permitidos a un conjunto predefinido. La capacidad de definir y utilizar tipos en CDS proporciona una herramienta poderosa para la gestión de datos en SAP. Esto asegura la consistencia, mejora la mantenibilidad y permite la reutilización de estructuras de datos predefinidas. Aunque existen algunas restricciones actuales, la dirección futura apunta a una mayor integración y flexibilidad en el uso de tipos en CDS.

Definición de Tipos en CDS:

Para crear un tipo CDS, se realiza a través de la carpeta de proyecto luego New en la opción **Other ABAP Repository Object** ubicar la carpeta **Core Data Services** y luego seleccionar **Type** y seleccionar la plantilla **defineType** que se encuentra en la carpeta **CDS Type (creation).**



En la declaración del tipo de datos del tipo cds pueden ser utilizados los tipos predefinidos (por ejemplo, abap.char), referenciar elementos de datos existentes en el diccionario de datos o incluso un tipo enum. Esto permite reutilizar definiciones de datos ya establecidas y asegurar la consistencia.

Sintaxis:

```
@EndUserText.label: 'Type - Cds Type'
define type cds_type_name: data_type.
```

Uso de Tipos en Entidades CDS:

Una vez definido, el tipo CDS puede ser utilizado en las definiciones de entidades CDS. Esto incluye la proyección de columnas y la utilización en elementos virtuales.

Ejemplos de uso en CDS con virtualización:

```
@AbapCatalog.viewEnhancementCategory: [#NONE]
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS - CDS with virtualization'
@Metadata.ignorePropagatedAnnotations: true
@ObjectModel.usageType:{
  serviceQuality: #X,
  sizeCategory: #S,
  dataClass: #MIXED
}
define view entity entity_name
 as select from data_source_name
{
 key component            as AliasName,
    component             as AliasName,
    @EndUserText.label: 'Label name'
    @ObjectModel.virtualElementCalculatedBy: 'ABAP:class_name'
    cast( value as cds_type_name) as AliasName
}
```

Ejemplos de uso en proyecciones con virtualización:

```
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'CDS - CDS with virtualization on projection'
@Metadata.ignorePropagatedAnnotations: true
define root view entity entity_name
 provider contract transactional_query
 as projection on cds_root_source
{
 key component            as AliasName,
    component             as AliasName,
    @EndUserText.label: 'Label name'
```

```
    @ObjectModel.virtualElementCalculatedBy: 'ABAP:class_name'
 virtual component : cds_type_name
}
```

Código Abap: se pueden utilizar en cualquier tipo de declaración de elementos como cualquier elemento de datos.

```
DATA: lv_variable TYPE cds_type_name.
```

Restricciones y Consideraciones:

- **Compatibilidad de Tipos:** Es esencial que los tipos utilizados en las definiciones de CDS sean compatibles con los datos subyacentes. En algunos casos, puede ser necesario aplicar un casting para asegurar la compatibilidad.
- **Limitaciones en Tablas de Persistencia:** Actualmente, no es posible utilizar directamente los tipos enum en las tablas de persistencia debido a restricciones del compilador. SAP planea soportar esta funcionalidad en futuras versiones mediante la definición de tablas directamente en CDS.
- **Buenas Prácticas:** Se recomienda definir tipos en CDS para estructuras de datos reutilizables y aplicar casting solo cuando sea absolutamente necesario para evitar complicaciones en el mantenimiento del código.