C_ABAPD_2309$^{Q\&As}$

SAP Certified Associate - Back-End Developer - ABAP Cloud

Pass SAP C_ABAPD_2309 Exam with 100% Guarantee

QUESTION 1

Which RESTful Application Programming object can be used to organize the display of fields in an app?

A. Data model view

B. Metadata extension

C. Service definition

D. Projection view

Correct Answer: B

A metadata extension is a RESTful Application Programming object that can be used to organize the display of fields in an app. A metadata extension is a CDS view that annotates another CDS view with UI annotations, such as labels, icons, or facets. These annotations define how the data should be presented in the app, such as which fields should be shown on the object page, which fields should be editable, or which fields should be used for filtering or sorting. A metadata extension can also be used to add custom actions or validations to the app12. References: 1: Refine the Object Page with Annotations | SAP Tutorials 2: ABAP RAP : Enabling custom actions with a dialog for additional input fields | SAP Blogs

QUESTION 2

What variable type is connection full based on the following code?

DATA connection full TYPE /DMD/I_Connection.

A. Simple variable

B. Structure

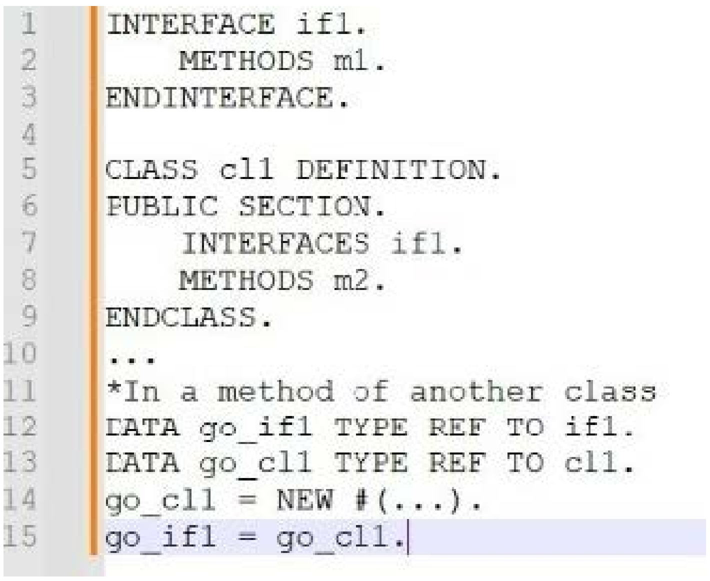
C. Internal Table

Correct Answer: B

Based on the following code, the variable type of connection_full is a structure. A structure is a complex data type that consists of a group of related data objects, called components, that have their own data types and names. A structure can be defined using the TYPES statement or based on an existing structure type, such as a CDS view entity or a CDS DDIC-based view. In this case, the variable connection_full is declared using the TYPE addition, which means that it has the same structure type as the CDS view entity /DMO/I_Connection. The CDS view entity /DMO/I_Connection is a data model view that defines a data model based on the database table /DMO/Connection. The CDS view entity /DMO/I_Connection has the following components: carrid, connid, airpfrom, airpto, distance, and fltime. Therefore, the variable connection_full has the same components as the CDS view entity /DMO/I_Connection, and each component has the same data type and length as the corresponding field in the database table /DMO/Connection. References: CDS Data Model Views - ABAP Keyword Documentation, DATA - ABAP Keyword Documentation, Structure Types - ABAP Keyword Documentation

QUESTION 3

Exhibit:



```
INTERFACE if1.
    METHODS m1.
ENDINTERFACE.

CLASS cl1 DEFINITION.
PUBLIC SECTION.
    INTERFACES if1.
    METHODS m2.
ENDCLASS.
...
*In a method of another class
DATA go_if1 TYPE REF TO if1.
DATA go_cl1 TYPE REF TO cl1.
go_cl1 = NEW #(...).
go_if1 = go_cl1.
```

What are valid statements? Note: There are 3 correct answers to this question.

A. go_if 1 may call method ml with go_ift->ml().

B. Instead of go ell = NEW #(...) you could use go ifl = NEW cll(. ...).

C. go_cll may call method ml with go_dl->ifl-ml().

D. Instead of go_cll = NEW #() you could use go_iff - NEW #(...).

E. go_ifl may call method m2 with go if->m2(...).

Correct Answer: ABE

The following are the explanations for each statement:

A: This statement is valid. go_ifl may call method ml with go_ifl->ml(). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_ifl. The class cll implements the interface ifl, which means that it provides an implementation of the method ml. The data object go_ifl is assigned to a new instance of the class cll using the NEW operator and the inline declaration operator @DATA. Therefore, when go_ifl->ml() is called, the implementation of the method ml in the class cll is executed123

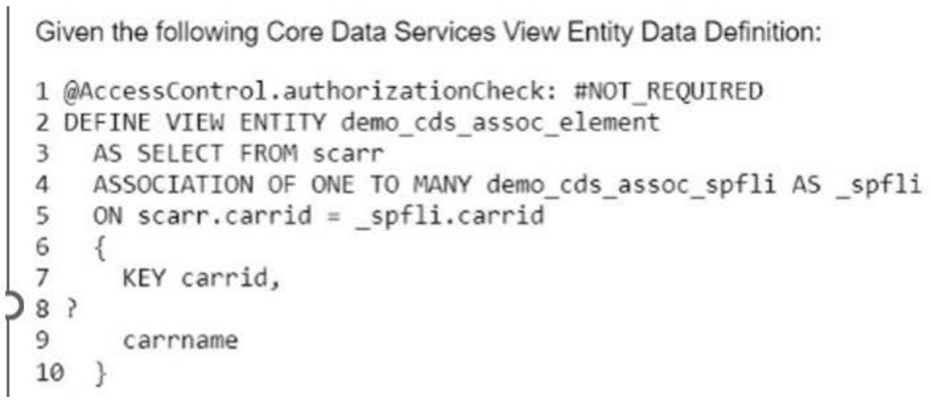
B: This statement is valid. Instead of go_cll = NEW #(...) you could use go_ifl = NEW cll(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it is compatible with the interface ifl. Therefore, go_ifl can be assigned to a new instance of the class cll using the NEW operator and the class name cll. The inline declaration operator @DATA is optional in this case, as go_ifl is already declared. The parentheses after the class name cll can be used to pass parameters to the constructor of the class cll, if any123

E: This statement is valid. go_ifl may call method m2 with go_ifl->m2(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The class cll also defines amethod m2, which is a public method of the class cll. Therefore, go_ifl can call the method m2 using the reference variable go_ifl. The method m2 is not defined in the interface ifl, but it is accessible through the interface ifl, as the interface ifl is implemented by the class cll. The parentheses after the method name m2 can be used to pass parameters to the method m2, if any123 The other statements are not valid, as they have syntax errors or logical errors. These statements are:

C: This statement is not valid. go_cll may call method ml with go_cll->ifl~ml(). This is because go_cll is a data object of type REF TO cll, which is a reference to the class cll. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_cll. However, the syntax for calling an interface method using a class reference is go_cll->ml(), not go_cll->ifl~ml (). The interface component selector ~ is only used when calling an interface method using an interface reference, such as go_ifl->ifl~ml(). Using the interface component selector ~ with a class reference will cause a syntax error123

D: This statement is not valid. Instead of go_cll = NEW #() you could use go_ifl = NEW #(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl cannot be instantiated, as it does not have an implementation. Therefore, go_ifl cannot be assigned to a new instance of the interface ifl using the NEW operator and the inline declaration operator @DATA. This will cause a syntax error or a runtime error. To instantiate an interface, you need to use a class that implements the interface, such as the class cll123 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation, NEW - ABAP Keyword Documentation

QUESTION 4



Given the following Core Data Services View Entity Data Definition:

```
1 @AccessControl.authorizationCheck: #NOT_REQUIRED
2 DEFINE VIEW ENTITY demo_cds_assoc_element
3   AS SELECT FROM scarr
4   ASSOCIATION OF ONE TO MANY demo_cds_assoc_spfli AS _spfli
5   ON scarr.carrid = _spfli.carrid
6   {
7     KEY carrid,
8 ?
9     carrname
10  }
```

The "demo_ods_assoc_spfi data source referenced in line #4 contains a field "connid" which you would like to expose in the element list. Which of the following statements would do this if inserted on line #8?

A. demo_ods_assoc_spfli.connid,

B. demo_ods_assoc_spfli-connid/

C. spfli-connid,

D. _spfli.connid/

Correct Answer: A

The statement that can be used to expose the field "connid" of the data source "demo_ods_assoc_spfli" in the element list is A. demo_ods_assoc_spfli.connid,. This statement uses the dot notation to access the field "connid" of the data

source "demo_ods_assoc_spfli", which is an association defined on line #4. The association "demo_ods_assoc_spfli" links the data source "demo_ods" with the table "spfli" using the field "carrid". The statement also ends with a comma to

separate it from the next element in the list12.

You cannot do any of the following:

B. demo_ods_assoc_spfli-connid/: This statement uses the wrong syntax to access the field "connid" of the data source "demo_ods_assoc_spfli". The dash notation is used to access the components of a structure or a table, not the fields of a data source. The statement also ends with a slash, which is not a valid separator for the element list12.

C. spfli-connid,: This statement uses the wrong data source name to access the field "connid". The data source name should be "demo_ods_assoc_spfli",not "spfli". The statement also uses the wrong syntax to access the field "connid", as explained above12.

D. _spfli.connid/: This statement uses the wrong data source name and the wrong separator to access the field "connid". The data source name should be "demo_ods_assoc_spfli", not "_spfli". The statement also ends with a slash, which is not a valid separator for the element list12. References: 1: ABAP CDS - SELECT, select_list - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - SELECT, from - ABAP Keyword Documentation - SAP Online Help

QUESTION 5

Which extensibility type does SAP recommend you use to enhance the existing UI for an SAP Fiori app?

A. Key user

B. Classic

C. Side-by-side

D. Developer

Correct Answer: D

Explanation: According to the SAP clean core extensibility and ABAP cloud topic, SAP recommends using developer extensibility to enhance the existing UI for an SAP Fiori app. Developer extensibility allows you to use the UI adaptation editor in SAP Web IDE to modify the UI layout, add or remove fields, and bind them to the data model. You can also use the SAPUI5 framework to create custom controls, views, and controllers. Developer extensibility is based on the in-app extensibility concept, which means that the extensions are part of the same application and are deployed together with the app. Developer extensibility requires developer skills and access to the source code of the app. References: SAP Learning Hub, SAP S/4HANA Cloud Extensibility - In-App Extensibility, SAP Fiori: Extensibility