WRITE DATA WITH CUSTOM DATA TYPES

Un tipo de dato personalizado (custom data type) es un objeto de datos definido por el usuario que imita estructuralmente la agrupación lógica de datos relacionados creados dentro de las tablas de subase de datos relacional. Es justo como el almacén de datos, lo que le permite estructurar y escribir datos en la base de datos

El almacen de datos (data sbor) es la conexión que comunica los datos de registros modificados desde un CDT a una base de datos relacional.

CDT vs Tipos de Registro (Record Type)

Los tipos de registro son una forma eficiente y rapidos de consultar asi como para relacionar datos de diferentes tablas. Mientras los CDT se utilizan para actualizar, insertar y eliminar datos en una tabla.

- Los CDT imitan la agrupación lógica de datos que se encuentran en una base de datos relacional y le permiten estructurar y escribir datos en la base de datos.
- Las aplicaciones generalmente se basarán en varias tablas de base de datos y por lo tanto, en multiples CDTs.
- Un almacen de datos conecta Appian con una base de datos relacional.

CREATE A CDT AND DATA STORE

5 Maneras de crear un CDT

#1- Automaticamente cuando crea una acción de registro, Quiza la forma mas sencilla e intuitiva de crear un CDT

#2- Tabla de base de datos existente. Utilice esta opción cuando tenga una tabla existente en la base de datos y necesite crear un CDT para escribir en una tabla dentro de Appian

#3- Desde cero: No es tan comun, pero puede crear un CDT desde cero. La creacion de un CDT desde cero requerirá que agregue cada campo de datos y configure los parametros, incluidos nombre, tipo, longitud, matriz y clave.

#4- CDT existente duplicado: Cuando necesite hacer un CDT similar a otro CDT. Se duplicara todas las configuraciones existentes en el CDT original incluidos las anotaciones JPA.

#5- Importar XSD: Si se tiene un archivo XSD que define su CDT. También puede crear o editar varios tipos de datos a la vez importando un solo archivo XSD.

Los CDT son diferentes a cualquier otro objeto de diseño porque estan asegurados al nivel de los objetos que los usan. Por ejemplo si una interfaz llama a un tipo de dato personalizado, se aplica la seguridad de la interfaz.

Comportamiento	Admin Sist.	Diseñador
• Crear y editar el tipo de dato	✓	✓
• Ver el tipo de dato	✓	✓
• Descargar el XSD del tipo de datos	✓	X
• Eliminar el tipo de datos	✓	X

* El almacen de datos es el responsable de conectar una aplicación a una base de datos relacional
* Los CDT se agregan al almacen de datos como entidades
* Vuelva a publicar el almacen de datos cada vez que agregue una nueva entidad

EDICION DE CDT

Si bien los CDT son editables, es importante recordar que su enfoque para editar CDT difiere de otros objetos del diseño. Dado que los CDT estan conectados a las tablas de base de datos externas.
Antes de actualizar su CDT, use la vista Dependientes para determinar donde se usa exactamente el CDT y los impactos de sus cambios

- Agregar cambios: Puede agregar un nuevo campo a su modelo de datos, simplemente agregando un nuevo campo al CDT. Una vez realizado el cambio, deberá verificar las asignaciones de tablas y volver a publicar el almacen de datos.

- Edición de campos: El método de edición más común es hacer el cambio directamente en la columna de la base de datos. Después descargar el archivo XSD. Realizar las ediciones en el archivo y crear una nueva versión del CDT a partir del archivo XSD actualizado.

CONSULTA TUS DATOS

- Query: Expresión que le permite acceder a sus datos para mostrarlos en diferentes partes de su aplicación.

- Query Editor: Una poderosa herramienta que le permite crear y probar consultas utilizando una experiencia guiada

- Expression Rule: Una expresión almacenada que devuelve un valor que puede estar influenciado por una o más entradas.

Los datos que retornan las consultas pueden ser usados para:

- Rellenar formularios
- Completar acciones
- Representar un KPI (indicadores clave de rendimiento)

* Para escribir datos en una base de datos, debe usar un tipo de dato personalizado (CDT)

▷ Puede consultar dos tipos de objetos:

* Tipo de registro (Record Type): Consulta datos de un tipo de registro usando a!queryRecordType(). Esto permite consultar datos de una base de datos, así como de Salesforce y otros servicios web cuando el tipo de registro tiene habilitado la sincronización de datos.
  - Interfaces
  - Reports
  - Record views
* Entidad de almacen de datos (Data Store Entity): Usando a!queryEntity(). Permite consultar datos directamente desde una base de datos.
  - Consulta de selección
  - Consulta de agregación

▷ Generar una consulta de selección de tipo registro

- Consulta de selección de tipo de registro mediante el editor de consultas.
  1. Seleccionar el data source (Record Type)
  2. Buscar el tipo de registro específico que quieres buscar
  3. Seleccionar los campos donde generara la consulta.
     * Si tu tipo de registro tiene relaciones también puedes seleccionar campos del registro relacionado

).data al final de la expresión de consulta.

Generar una consulta de selección de entidad de almacen de datos.

Si necesita consultar una entidad de almacen de datos (data store entity), debería seleccionar una constante que apunte a esa entidad de almacen de datos. A continuación puede elegir los campos que desea devolver. También se pueden filtrar los datos, así como aplicar paginación y clasificación

#1- Dentro de una nueva regla de expresión inicie el editor de consultas

#2- Cambie la entrada a fuente de datos a Entidad de almacenamiento de datos.

#3 Seleccione la constante que apuntara a la entidad del almacen

#4- Seleccione los campos que desea que se devuelvan

#5- Actualice las opciones de paginación y clasificación si es necesario

#6- Agregue condiciones de filtro si es necesario

#7- Click en Generar Consulta

#8- Haga click en Probar Regla y confirme que los resultados sean los esperados.

Crear una consulta de agregación de tipo de registro.

#1- Abrir el Editor de consultas

#2- Seleccionar el tipo de registro

#3- Seleccionar Record Type, como la fuente de datos

#4- Click Agregar Registros, Aggregate

*). data al final de la expresión

a) aggregation(fields) 'Recibe dos parametros

* agrupaciones (groupings)
* Medidas (Measures)

APLICAR FILTROS

Un filtro permite aplicar un conjunto de condiciones para que la consulta sólo devuelva los datos que necesita.

Ejemplos de filtros adicionales

- Filtrar valores null
- Devolver datos entre dos fechas
- Filtrar más de un campo

Operador a!queryLogicalExpression()

Se puede usar los operadores como AND, OR y AND_ALL directamente en el Editor de expresiones

Filtrar registros relacionados con a!relatedRecordData()

Si estas consultando un tipo de registro que tiene una relación de uno a varios, puedes usar el parametro relatedRecordData, en la consulta para filtrar los datos devueltos por esa relación de uno a varios

Los parametros de a!relatedRecordData() son

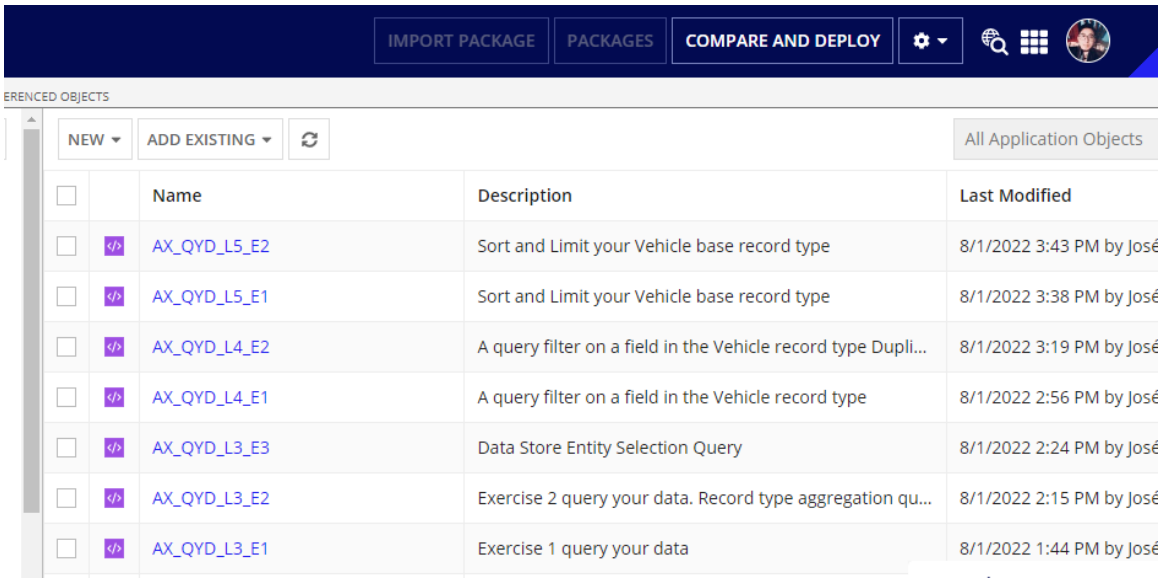
* relación
* clasificación
* limite
* Filtros.

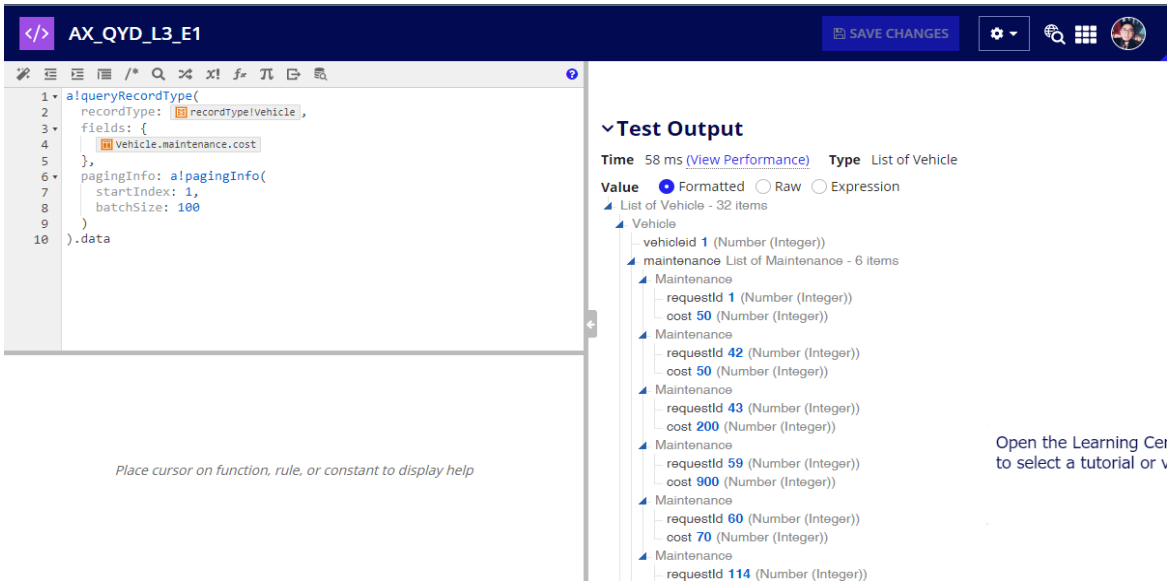
ORDENAR Y LIMITAR DATOS

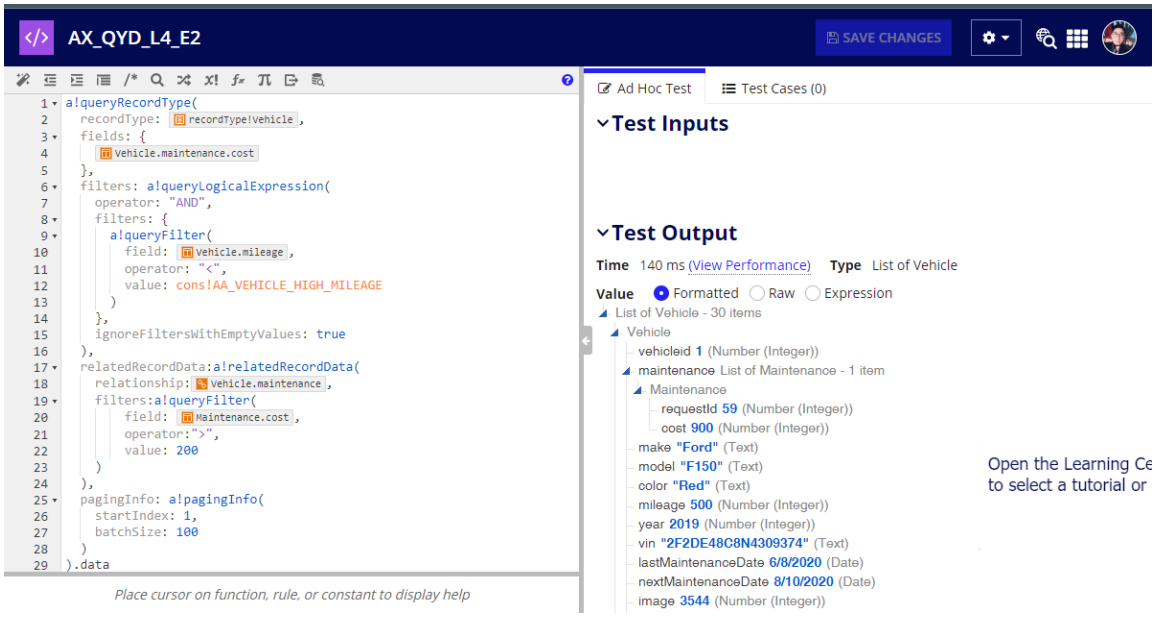
Esto permitira organizar los resultados de su consulta de una manera poderosa para presentar datos utilizables, a los usuarios sin necesidad de que interactuen con una cuadrícula o gráfico para tomar sus propias decisiones de clasificación.

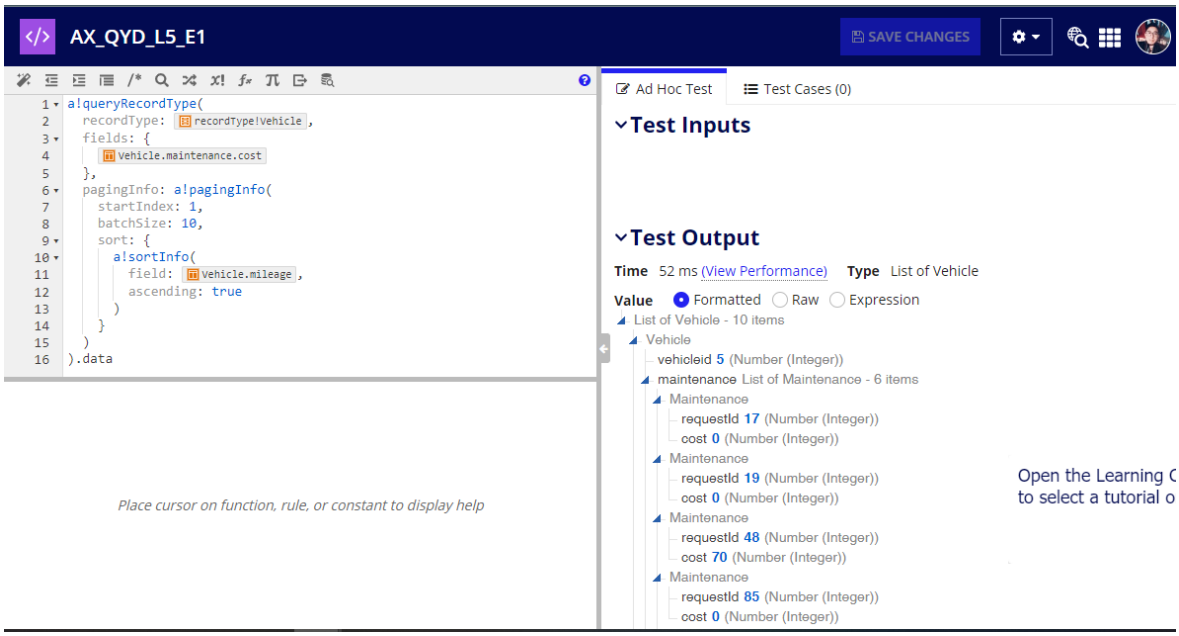
Ejemplos de clasificación adicionales

- Ordenar varios campos en una consulta de selección
- Ordenar varios campos en una consulta de agregación.


IMPORT PACKAGE
PACKAGES
COMPARE AND DEPLOY
ERENCED OBJECTS
NEW
ADD EXISTING
All Application Objects
Name
Description
Last Modified
AX_QYD_L5_E2
Sort and Limit your Vehicle base record type
8/1/2022 3:43 PM by José
AX_QYD_L5_E1
Sort and Limit your Vehicle base record type
8/1/2022 3:38 PM by José
AX_QYD_L4_E2
A query filter on a field in the Vehicle record type Dupli...
8/1/2022 3:19 PM by José
AX_QYD_L4_E1
A query filter on a field in the Vehicle record type
8/1/2022 2:56 PM by José
AX_QYD_L3_E3
Data Store Entity Selection Query
8/1/2022 2:24 PM by José
AX_QYD_L3_E2
Exercise 2 query your data. Record type aggregation qu...
8/1/2022 2:15 PM by José
AX_QYD_L3_E1
Exercise 1 query your data
8/1/2022 1:44 PM by José


AX_QYD_L3_E1
SAVE CHANGES
1 a!queryRecordType(
2 recordType: recordType!Vehicle,
3 fields: {
4 Vehicle.maintenance.cost
5 },
6 pagingInfo: a!pagingInfo(
7 startIndex: 1,
8 batchSize: 100
9)
10).data
Place cursor on function, rule, or constant to display help
Test Output
Time 58 ms (View Performance) Type List of Vehicle
Value Formatted Raw Expression
List of Vehicle - 32 items
Vehicle
vehicleId 1 (Number (Integer))
maintenance List of Maintenance - 6 items
Maintenance
requestId 1 (Number (Integer))
cost 50 (Number (Integer))
Maintenance
requestId 42 (Number (Integer))
cost 50 (Number (Integer))
Maintenance
requestId 43 (Number (Integer))
cost 200 (Number (Integer))
Maintenance
requestId 59 (Number (Integer))
cost 900 (Number (Integer))
Maintenance
requestId 60 (Number (Integer))
cost 70 (Number (Integer))
Maintenance
requestId 114 (Number (Integer))
Open the Learning Ce
to select a tutorial or


AX_QYD_L4_E2
SAVE CHANGES
a!queryRecordType(
recordType: recordType!Vehicle,
fields: {
Vehicle.maintenance.cost
},
filters: a!queryLogicalExpression(
operator: "AND",
filters: {
a!queryFilter(
field: Vehicle.mileage,
operator: "<",
value: cons!AA_VEHICLE_HIGH_MILEAGE
)
},
ignoreFiltersWithEmptyValues: true
),
relatedRecordData:a!relatedRecordData(
relationship: Vehicle.maintenance,
filters:a!queryFilter(
field: Maintenance.cost,
operator:">",
value: 200
)
),
pagingInfo: a!pagingInfo(
startIndex: 1,
batchSize: 100
)
).data
Place cursor on function, rule, or constant to display help
Ad Hoc Test
Test Cases (0)
Test Inputs
Test Output
Time 140 ms (View Performance)
Type List of Vehicle
Value Formatted Raw Expression
List of Vehicle - 30 items
Vehicle
vehicleid 1 (Number (Integer))
maintenance List of Maintenance - 1 item
Maintenance
requestId 59 (Number (Integer))
cost 900 (Number (Integer))
make "Ford" (Text)
model "F150" (Text)
color "Red" (Text)
mileage 500 (Number (Integer))
year 2019 (Number (Integer))
vin "2F2DE48C8N4309374" (Text)
lastMaintenanceDate 6/8/2020 (Date)
nextMaintenanceDate 8/10/2020 (Date)
image 3544 (Number (Integer))
Open the Learning Ce
to select a tutorial or


AX_QYD_L5_E1
SAVE CHANGES
a!queryRecordType(
recordType: recordType!Vehicle,
fields: {
Vehicle.maintenance.cost
},
pagingInfo: a!pagingInfo(
startIndex: 1,
batchSize: 10,
sort: {
a!sortInfo(
field: Vehicle.mileage,
ascending: true
)
}
)
).data
Place cursor on function, rule, or constant to display help
Ad Hoc Test
Test Cases (0)
Test Inputs
Test Output
Time 52 ms (View Performance)
Type List of Vehicle
Value Formatted Raw Expression
List of Vehicle - 10 items
Vehicle
vehicleid 5 (Number (Integer))
maintenance List of Maintenance - 6 items
Maintenance
requestId 17 (Number (Integer))
cost 0 (Number (Integer))
Maintenance
requestId 19 (Number (Integer))
cost 0 (Number (Integer))
Maintenance
requestId 48 (Number (Integer))
cost 70 (Number (Integer))
Maintenance
requestId 85 (Number (Integer))
cost 0 (Number (Integer))
Open the Learning C
to select a tutorial o