

## Data Modeling

SAP Comm by default provides lots of tables (Cart ... Order ... Catalog ... Product ... User ... Promotions ... Categories...).

**Q1** = Do you think that – all these tables are enough for all the client needs (or) You might need to create some more custom tables?

= We might need to create some more tables.

**Q2** = Let's say ... SAP Comm is given Product able with 40 Cols. Do you think that – all these 40 Columns are enough for all client needs (or) You might need to add some custom cols?

= We might need to add some more columns.

**Q3** = Let's say ... SAP Comm is given Product able with 40 Cols. Do you think that – all these 40 Columns are used for client needs (or) We might not need some cols?

= We might not need some cols [Let's say 5 cols].

**Note** = How to create new table ... How to create new Col ... How SAP Comm is going to handle the DB ... How to create relations .....

**Ans** for all above Q = **SAP Comm Data Modeling**

**Best Practice** = Do not delete any “SAP Comm” provided tables or columns. You can just leave those if not used for your client.

“SAP Comm” data modeling helps an organization in maintaining their database and help to manage database connections and queries.

## Scenario = Java Vs “SAP Comm” Vs DB

SAP Comm	Java	DBMS
Item Type	Java Class	Table
Attribute	Class Members (Variables)	Columns
Instance	Object	Row
Service Layer Classes	Methods	Functions (Nearly)
Model Classes	POJO Classes	---
Generic Item	Object Class	---
Extends (Keyword to inherit a table)	Extends (Keyword to inherit a class)	---
Atomic types	Primitive data types	
Deployment Table="TableName"	---	Create Table TableName
PK	---	Primary Key

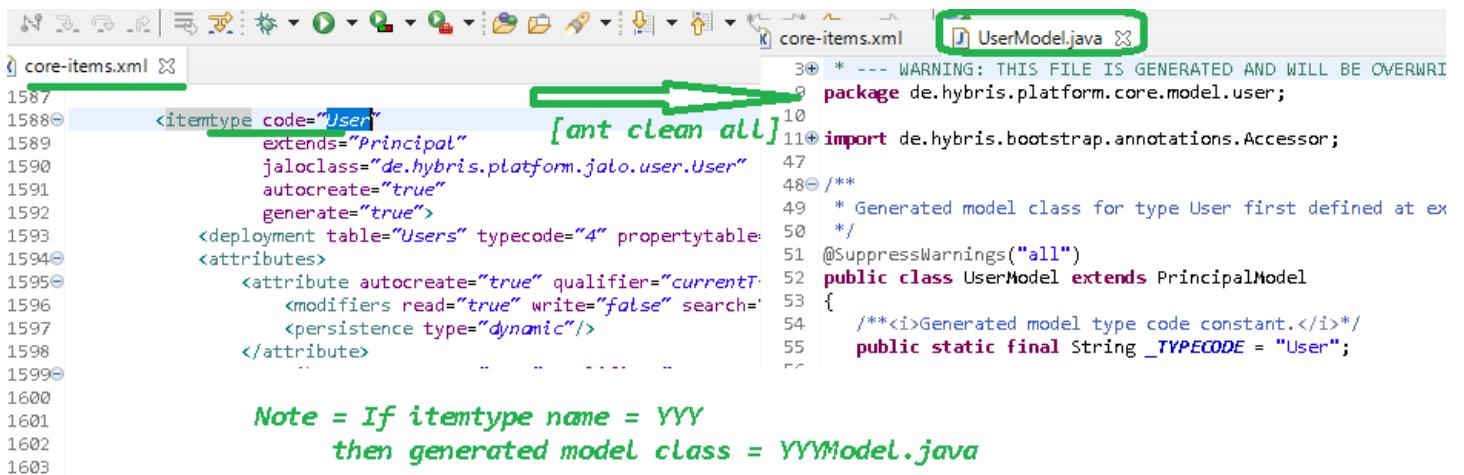
## Scenario = Model Classes

In Java – POJO classes [Plain Old Java Objects].

Java – POJO classes == SAP Comm – Model classes.

SAP Comm generates Model classes for each defined item type.

**Q = When model classes are generated? = build [ant clean all].**



```
core-items.xml
1587
1588<itemType code="User" [ant clean all]
1589    extends="Principal"
1590    jaloClass="de.hybris.platform.jalo.user.User"
1591    autoCreate="true"
1592    generate="true">
1593    <deployment table="Users" typeCode="4" propertyTable="UserProperty">
1594        <attributes>
1595            <attribute autoCreate="true" qualifier="currentT...
1596                <modifiers read="true" write="false" search="...
1597                <persistence type="dynamic"/>
1598            </attribute>
1599        ...
1600
1601
1602
1603
UserModel.java
3+ * --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN
9 package de.hybris.platform.core.model.user;
10
11 import de.hybris.bootstrap.annotations.Accessor;
12
13 /**
14  * Generated model class for type User first defined at ex...
15  */
16 @SuppressWarnings("all")
17 public class UserModel extends PrincipalModel
18 {
19     /**<i>Generated model type code constant.</i>*/
20     public static final String _TYPECODE = "User";
21 }
```

Note = If itemType name = YYY  
then generated model class = YYYModel.java

**Q = What model classes contains?**

Model class provides getter and setter method for attributes.

```
*core-items.xml
itemtype code="User"
    extends="Principal"
    jaloclass="de.hybris.platform.jalo.user.User"
    autoreate="true"
    generate="true"
<deployment table="Users" typecode="4" propertytable="UserProps"/>
<attributes>
    <attribute autoreate="true" qualifier="currentTime" type="java.util.Date">
    <attribute autoreate="true" qualifier="currentDate" type="java.util.Date">
    <attribute autoreate="true" qualifier="password" type="java.lang.String">

```

**Col Names**

**Q = What is there in \*Model.java?**

For each attribute in \*-items.xml ... there will be

- 1) Variable
- 2) getXXX()
- 3) setXXX()

```
*core-items.xml   *UserModel.java
3@ * --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN! ---
9 package de.hybris.platform.core.model.user;
10
11@ import de.hybris.bootstrap.annotations.Accessor;
12
13@ SuppressWarnings("all")
14 public class UserModel extends PrincipalModel
15 {
16     public static final String CURRENTTIME = "currentTime";
17
18     public static final String CURRENTDATE = "currentDate";
19
20     public static final String PASSWORD = "password";
21
22     @Accessor(separator = "currentDate", type = Accessor.Type.GETTER)
23     public Date getCurrentDate()
24     {
25         return getPersistenceContext().getDynamicValue(this,CURRENTDATE);
26     }
27
28     @Accessor(separator = "currentTime", type = Accessor.Type.GETTER)
29     public Date getCurrentTime()
30     {
31         return getPersistenceContext().getDynamicValue(this,CURRENTTIME);
32     }
33 }
```

**Q = What \*Mode.java files contains?**

1) Let's say we have 8 attributes in \*-items.xml then

In \*Model.java – We will have

==== 8 Variables

==== 8 setXXX Methods

==== 8 getXXX Methods

That means – For each attribute there will be 1 variable ... 1 setXXX method & 1 getXXX method.

2) If the attribute <persistence type="dynamic" ==> then this attribute will **not be created** in DB as a **Col** and will **not be in \*Model.java**.

3) If the attribute <persistence type=" property" ==> then this attribute will **be created** in DB as a **Col** and **will be in \*Model.java**.

4) If the attribute type="localized:java.lang.String" ==> then this attribute will be created in DB.

In \*Model.java – We will have: -

- 1) 1 variable
- 2) 2 setXXX() =
- 3) 2 getXXX() =

**Step 1**

```
<itemtype code="UserRight"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.security.UserRight"
  autocreate="true"
  generate="true">
<deployment table="UserRights" typecode="29"/>
<attributes>
  <attribute autocreate="true" qualifier="code" type="java.lang.String"></attribute>
  <attribute autocreate="true" qualifier="name" type="localized:java.lang.String">
    <modifiers read="true" write="true" search="true" removable="true" optional="true"/>
    <persistence type="property"/>
    <custom-properties></custom-properties>
  </attribute>
</attributes>
</itemtype>
```

**UserRight[8796093153309]**

**Step 3 = After "Update / INIT"**

Name	en	fr
es_CO		ru
in		zh_TW
int		ja

**Step 2 [ant clean all]**

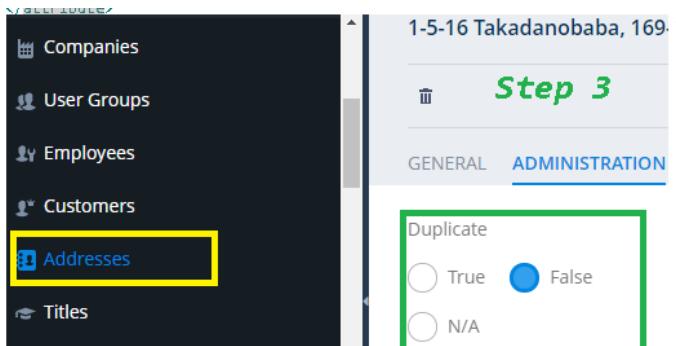
```
public class UserRightModel extends ItemModel
{
  /**
   * <i></i>Generated constant</i> - Attribute key of <code>UserRight</code>
  public static final String NAME = "name";
  @Accessor(qualifier = "name", type = Accessor.Type.GETTER)
  public String getName()
  {
    return getName(null);
  }
  @Accessor(qualifier = "name", type = Accessor.Type.GETTER)
  public String getName(final Locale loc)
  {
    return getPersistenceContext().getLocalizedValue(NAME, loc);
  }
  @Accessor(qualifier = "name", type = Accessor.Type.SETTER)
  public void setName(final String value)
  {
    setName(value,null);
  }
  @Accessor(qualifier = "name", type = Accessor.Type.SETTER)
  public void setName(final String value, final Locale loc)
  {
    getPersistenceContext().setLocalizedValue(NAME, loc, value);
  }
}
```

**Step 1**

```
<itemtype code="Address"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.user.Address"
  autocreate="true"
  generate="true">
<deployment table="Addresses" typecode="23" propertytable="AddressProps"/>
<attributes>
  <attribute qualifier="duplicate" type="java.lang.Boolean">
    <modifiers read="true" write="true" search="false" optional="true"/>
    <persistence type="property"/>
    <model>
      <getter default="true" name="duplicate">
        <nullDecorator>Boolean.valueOf(false)</nullDecorator>
      </getter>
    </model>
  </attribute>
</attributes>
```

**Step 2**

```
public class AddressModel extends ItemModel
{
  public static final String DUPLICATE = "duplicate";
  @Accessor(qualifier = "duplicate", type = Accessor.Type.SETTER)
  public void setDuplicate(final Boolean value)
  {
    getPersistenceContext().setPropertyValue(DUPLICATE, value);
  }
  @Accessor(qualifier = "duplicate", type = Accessor.Type.GETTER)
  public Boolean getDuplicate()
  {
    final Boolean value = getPersistenceContext().getPropertyValue(DUPLICATE);
    return value != null ? value : Boolean.valueOf(false);
  }
}
```



**Step 1**

```
<itemtype code="User"
  extends="Principal"
  jaloClass="de.hybris.platform.jalo.user.User"
  autocreate="true"
  generate="true">
<deployment table="Users" typecode="4" propertytable="UserProps"/>
<attributes>
```

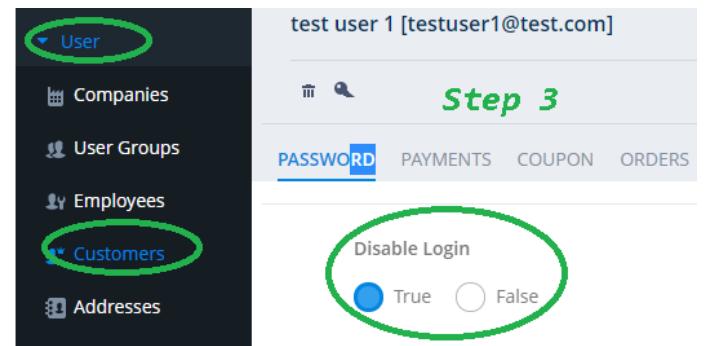
**Step 2**

```
public class UserModel extends PrincipalModel
{
  public static final String LOGINDISABLED = "loginDisabled";
  @Accessor(qualifier = "loginDisabled", type = Accessor.Type.SETTER)
  public void setLoginDisabled(final boolean value)
  {
    getPersistenceContext().setPropertyValue(LOGINDISABLED, toObject(value));
  }
  @Accessor(qualifier = "loginDisabled", type = Accessor.Type.GETTER)
  public boolean isLoginDisabled()
  {
    return toPrimitive((Boolean)getPersistenceContext().getPropertyValue(LOGINDISABLED));
  }
}
```

**Step 2**

**setXXX()**  
**isXXX()**

```
/*
51  */
52  @SuppressWarnings("all")
53  public class UserModel extends PrincipalModel
54  {
  public static final String LOGINDISABLED = "loginDisabled";
  @Accessor(qualifier = "loginDisabled", type = Accessor.Type.SETTER)
  public void setLoginDisabled(final boolean value)
  {
    getPersistenceContext().setPropertyValue(LOGINDISABLED, toObject(value));
  }
  @Accessor(qualifier = "loginDisabled", type = Accessor.Type.GETTER)
  public boolean isLoginDisabled()
  {
    return toPrimitive((Boolean)getPersistenceContext().getPropertyValue(LOGINDISABLED));
  }
}
```

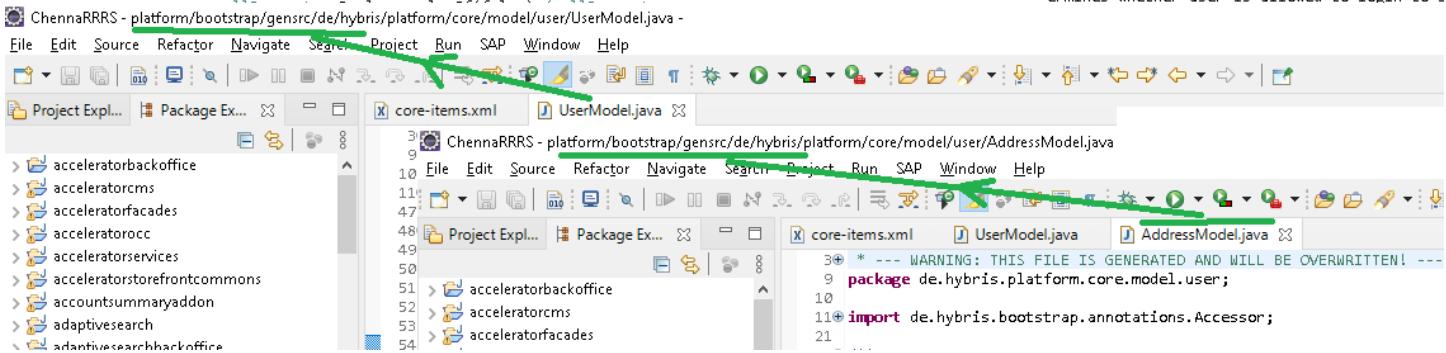


```

46
47<database-schema database="oracle" primary-key="primary key" null="" not-null="not null" >
48    <type-mapping type="java.lang.Boolean" persistence-type="number(1,0)" />
49    <type-mapping type="boolean" persistence-type="number(1,0) DEFAULT 0" />
50
51
52

```

**Note = All \*Model.java files will be created in “platform” folder.**



**Q = What is the purpose of the \*Model.java (or) \*Model.class file?**

By using \*Model classes – We can perform DB operations.

By using \*Model classes – We can interact with DB tables.

**Example =**

Insert Record = \*Model Classes

Delete Record = \*Model Classes

Get the Records = \*Model classes

User [Itemtype] === UserModel

Cart [Itemtype] === CartModel

==== Using \*Model classes only we can perform **CURD operations** on Tables.

## Database

### Emp (Table): -

Empld	EmpName	EmpSal
101	Chenna	10\$
102	ABC	15\$

POJO Class (Java) == Model Class (Hybris)

Contains: -

- 1) For every DB Table Col there will be 1 variable
- 2) For every DB Table Col there will be 1 setXXX() & 1 getXXX()

## Java Side

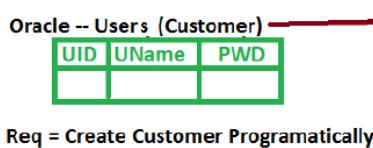
### Emp.java (POJO Class): -

```
=====
public class Emp
{
```

```
    String Empld;
    String EmpName;
    String EmpSal;
    setEmpld(==){==}
    getEmpld(){==}
    setEmpName(==){==}
    getEmpName(){==}
    setEmpSal(==){==}
    getEmpSal(){==}
```

```
}
```

**Scenario** = Assume that, In Oracle DB we have table = User (Customer) & what happen when you try to create customer from SAP Comm?



Step 1 = Create Customer Obj

Backoffice -- Customers

1st Way = CustomerModel cm = new CustomerModel(); (Legacy)

2nd Way = modelService.create(CustomerModel.class) -- Best way  
CustomerModel cm = modelService.create(CustomerModel.class);

Step 2 = Enter the Data (or) Set the Data



cm.setUid("123"); cm.setUname("abcd"); cm.setPwd("aaa");

Step 3 = Create Customer (or) Perform DB Operation



modelService.save(cm);

Note: - Using Model classes only we can perform the DB operations (CURD).

If we have XXX itemtype then the generated Model class = XXXModel.java

modelService.save(cm);

Note: - After completion of 3rd Step... What is the output?

Oracle -- Users (Customer)

UID	UName	PWD
123	abcd	aaa

Record is created.

Q: What are the general Steps?

Step 1 = Create Obj

XXXModel obj = modelService.create(XXXModel.class)

Step 2 = Set the Data

obj.xxx(==); obj.yyy(==);

Step 3 = Perform the DB operation

modelService.save(obj);

**Conclusion** = With this example – we can able to understand lots of code.

# Scenario = Create Emp Table with 3 Cols

Req1 -- Create Emp Table with 3 Cols

Oracle (Emp)

EmpID    EmpName    EmpSal

**Step 1 =**

Hybris -- \*-items.xml  
 =====  
 <itemtype code="Emps" ...>  
     <deployment table = "Emp" ...>  
         <attribute qualifier = "EmpID" ...>  
         <attribute qualifier = "EmpName" ...>  
         <attribute qualifier = "EmpSal" ...>  
 </itemtype>

Note: - If itemtype = XXX then during the build (ant clean all), there will be model class created = XXXModel.java

**Step 2 = Build (ant clean all)**

During this Model class will be generated.

EmpsModel.java

```
public class EmpsModel{  

    String EmpID; String EmpName;  

    StringEmpSal;  

    setEmpID(){==}> getEmpID(){==}  

    setEmpName(){==}> getEmpName(){==}  

    setEmpSal(){==}> getEmpSal(){==}
```

Note: - With the help of Model classes only you will be able to set the data into DB (or) You will be able to get the data from DB.

**Step 3 =**  
**Start the Server & Perform Update / INIT (1st time)**

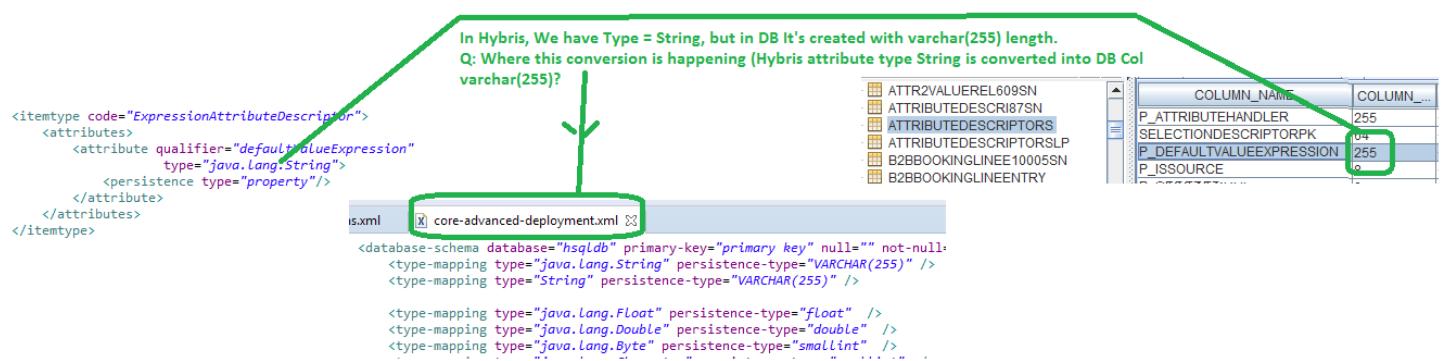
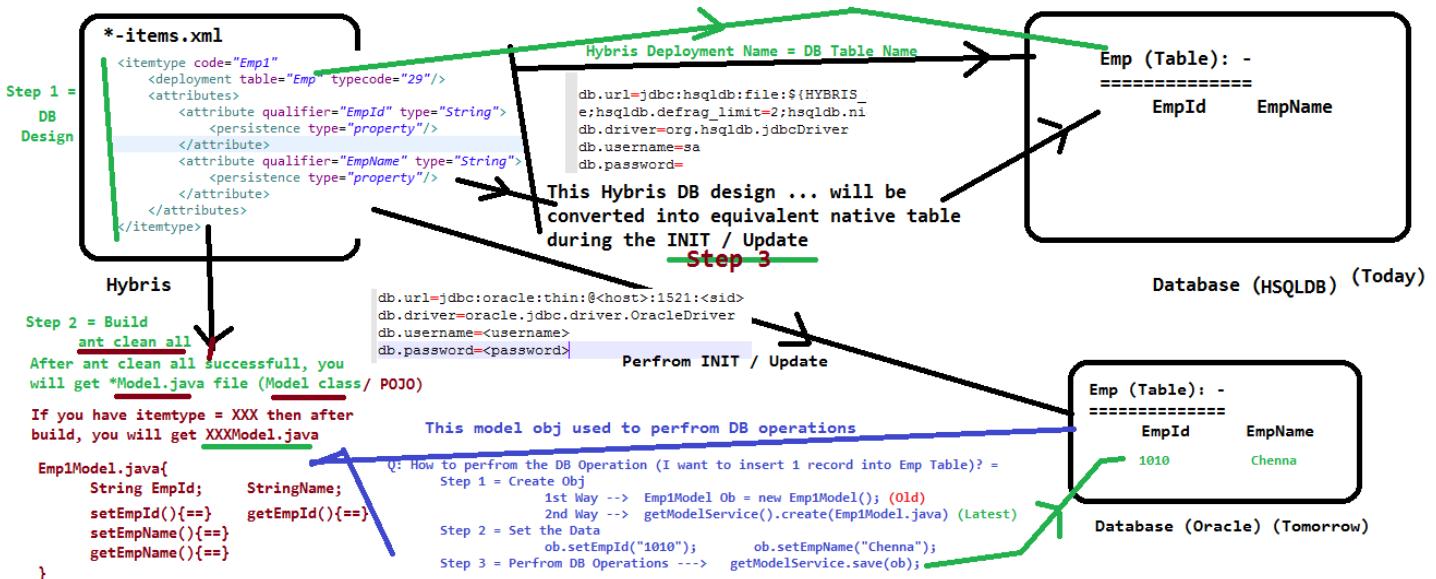
If your DB = Oracle then completing the Update ... You will see a table called "Emp" with 3 Cols (EmpID... EmpName... EmpSal)

**Results =**

Oracle (Emp)

EmpID    EmpName    EmpSal

# Scenario = SAP Comm is Database Independent.



# Scenario = Itemtype vs Deployment

**Your Hybris Data Model is converted into Equivalent Native table during INIT / Update.**

**(Deployment is Optional)**

This (Itemtype name = User) will be used for Hybris purpose.

Note: - We can have same name for itemtype & deployment.

If you have itemtype code = XXX then during the build (ant clean all) --> XXXModel.java (Model Class) will be generated.

Using this Model class only we can perform the DB Operations (Insert / Update / Delete).

**This is for Hybris Purpose**

**This is for DB Purpose**

**Step 1 = (EAD)**

```
<itemtype code="ExpressionAttributeDescriptor"
  extends="AttributeDescriptor">
  <attribute qualifier="defaultValueExpression" />
</itemtype>
```

**Step 2 = (AD)**

```
<itemtype code="AttributeDescriptor"
  extends="Descriptor" />
  <deployment "AttributeDescriptor">
    <attribute qualifier="defaultValue" />
  </deployment>
</itemtype>
```

**Step 3 = Do build (ant clean all)**

During the build Model classes will be generated.

```
EADMModel.java extends ADM      ADMModel.java
{   String dVE;           {   String dV;
  getdVE(){==}           setdV(){==}
  setdVE(){==}           getdV(){==}
}
```

Using this Model classes only we perform the DB Operations.

**Step 4 = Start the Server**

**DB Connection**

```
db.url=jdbc:hsql:file:$({HYBRIS}
db.driver=org.hsqldb.jdbcDriver
db.username=sa
db.password=
```

**Step 5 = Perform Update / INIT**

During Update / INIT, Hybris Runtime Env converts your Datamodel into equivalent native tables.

**Step 6 = Using the Model classes you can perform the DB Operations. (Insert 1 Record into TAD table)**

```
XXX.java: -
=====
Step 1 = Create Obj
  EADMModel obj = new EADMModel(); -- Sol1
  EADMModel obj = modelService.create(EADMModel.class) -- Sol2
Step 2 = Set the Data
  obj.setDVE("1001");          obj.setDV("One");
Step 3 = Perform the DB Operation (Save Record)
  modelService.save(obj)
Step 7 = Do build (ant clean all) & Start the Server
```

**Step 5 = Results**

Table (TAD)	dVE	dV
====	====	====

**Step 8 = Test the Results (Execute XXX.java)**

**Step 8 = Results**

Table (TAD)	dVE	dV
====	====	====
1001	One	One

**Note:** 1) If current item type we don't have the <deployment> tag then attributes will store in parent item type / Generic Item / Item.  
2) If you want attributes in your table then specify <deployment> tag.

**Scenario** = Let's say, your company already have table with some columns & business asked you to create some more fields. How to create & what kind of challenges we may encounter?

MaterialPlant (Table) (Oracle)		
Plant Name	Plant Desc	Plant Code
ABC11	XYZ1	P101
ABC2	XYZ2	P102

Plant Name	Plant Desc	Plant Code	valueCenter	profitCenter
ABC11	XYZ1	P101		
ABC2	XYZ2	P102		
ABC3	XYZ3	P103	Hello	Testing

Today's Req = Business asked to add another 2 fields called "valueCenter & profitCenter".

```
<attribute qualifier="profitCenter" type="java.lang.String">
  <persistence type="property"/>
</attribute>
<attribute qualifier="valueCenter" type="java.lang.String">
  <persistence type="property"/>
</attribute>
```

After Build (ant clean all)

After Update

Plant Name	Plant Desc	Plant Code	valueCenter	profitCenter
ABC11	XYZ1	P101		
ABC2	XYZ2	P102		

After 2 Days

After the entering the data, business realized that, the cols whatever created are having length 255.

Now, business came back & ask me to restrict the length = 10.

```
<attribute qualifier="profitCenter" type="java.lang.String">
  <persistence type="property"/>
    <columntype database="oracle" ><value> VARCHAR2(10) </value>
    <columntype database="mysql" ><value> VARCHAR(20) </value>
    <columntype database="sqlserver" ><value> NVARCHAR(15) </va
  </persistence>
<attribute qualifier="valueCenter" type="java.lang.String">
  <modifiers read="true" write="true" optional="true" />
  <persistence type="property">
    <columntype database="oracle" ><value> VARCHAR2(10) </value>
    <columntype database="mysql" ><value> VARCHAR(20) </value>
    <columntype database="sqlserver" ><value> NVARCHAR(10) </va
  </persistence>
```

This coding is done based the requirement.  
"ant clean all" = Build  
Update = Over

... After update ..

Expected results = We should valueCenter & profitCenter with 10 Chars length.

But There is no expected in the table? ... Why?

DB Thumb =  
If any data in Col then Increasing size is possible... Decreasing the size is not possible.

- Q: What is the Solutions?
- 1) Export ... Create table .. Import
  - 2) INIT
  - 3) Interceptors
  - 4) Validations
  - 5) DB Admin can do the changes

Req 1 = Create 2 Cols called VC & PC.  
Note:- Material Plant table is already having 2 Cols (M.No & M.Desc).  
Material Plant (Table)= Oracle

M. No	M. Dec
M101	Abc
M102	Xyz

M. No	M. Dec	VC	PC
M101	Abc	255	255
M102	Xyz		

Note:- Business started using those new Cols. And started entering the data.  
After 10 days - Business identified that, this Cols are created with 255 length & they want only 10 chars length.

Now -- Business asked to do the changes.

M. No	M. Dec	VC (255)	PC (255)
M101	Abc		
M102	Xyz		

Step 1 = Do the data modeling

```
*core Ext --> *-items.xml
<itemtype code="MaterialPlant" generate="true" autorecreate="true">
  <deployment table="MaterialPlant" typecode="12039" />
  <attributes>
    <attribute qualifier="profitCenter" type="java.lang.String">
      <persistence type="property"/>
    </attribute>
    <attribute qualifier="valuecenter" type="java.lang.String">
      <persistence type="property"/>
    </attribute>
  </attributes>
```

If persistence type ="property" then your attribute (VC) will be created in DB.

If persistence type = "dynamic" then your attribute (VC) will not be created in DB. It's just Hybris purpose.

Step 2 = Do the Build (ant clean all)... Start the Server .... hAC -- Update ==> Results

M. No	M. Dec	VC (255)	PC (255)
M101	Abc		
M102	Xyz		

Step 3 = Start making the code changes, such way that, VC & PC will have only 10 chars length.

```
*itemtype code="MaterialPlant" generate="true" autorecreate="true">
  <deployment table="MaterialPlant" typecode="12039" />
  <attributes>
    <attribute qualifier="profitCenter" type="java.lang.String">
      <persistence type="property"/>
      <columntype database="oracle" ><value> VARCHAR2(10) </value>
      <columntype database="mysql" ><value> VARCHAR(20) </value>
      <columntype database="sqlserver" ><value> NVARCHAR(10) </va
    </persistence>
    <attribute qualifier="valuecenter" type="java.lang.String">
      <modifiers read="true" write="true" optional="true" />
      <persistence type="property">
        <columntype database="oracle" ><value> VARCHAR2(10) </value>
        <columntype database="mysql" ><value> VARCHAR(20) </value>
        <columntype database="sqlserver" ><value> NVARCHAR(10) </va
      </persistence>
```

Why? = DB Rules.

If DB Col is having the data, then Increasing size is possible. Decreasing size is not possible.

Now -- What is the Solution?

Sol 1 = Ask DB admin to drop & create again. Make sure data is backed up.

Sol 2 = U take data back (ImpEx) ... Perform the INIT (if your business OK).

## Scenario = core-advanced-deployment.xml?

UserRightModel.java catalog-items.xml core-advanced-deployment.xml

```
<itemtype code="UserRight"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.security.UserRight"
  autorecreate="true"
  generate="true">
  <deployment table="UserRights" typecode="29"/>
  <attributes>
    <attribute autorecreate="true" qualifier="code" type="java.lang.String">
      <persistence type="property"/>
      <modifiers read="true" write="true" search="true" initial="true" optional="false" />
      <custom-properties>
        <property name="hmcIndexField">
          <value>thefield</value>
        </property>
      </custom-properties>
    </attribute>
    <attribute autorecreate="true" qualifier="name" type="localized:java.lang.String">
      <modifiers read="true" write="true" search="true" removable="true" optional="true" />
      <persistence type="property"/>
      <custom-properties>
        <property name="hmcIndexField">
          <value>thefield</value>
        </property>
      </custom-properties>
    </attribute>
```

Info	Content	Row Count	Columns	Primary Key	Exp
UNITSLP			COLUMN_...		
USERAUDIT			TYPE_N...		
USERGROUPROPS			IS_NULLA...		
USERGROUPS			DECIMAL_DIGI...		
USERGROUPSLP			COLUMN_...		
USERPHONENUMBER					
USERPROFILES					
USERPROPS					
USERRIGHTS					
USERRIGHTSLP					
USERS					
USERSPERMISSION					

While doing DB design in Hybris ... We mentioned code is of type "String".... But in HSQL DB we can see it's created with 255 Char length.

Q: How this conversion happened? (Hybris String --> HSQL DB Char 255)? = U can see this conversion mapping in core-advanced-deployment.xml

items.xml UserRightModel.java core-advanced-deployment.xml

```
<database-schema database="hsqldb" primary-key="primary key" null="" not-null="not null" >
  <type-mapping type="java.lang.String" persistence-type="VARCHAR(255)" />
  <type-mapping type="String" persistence-type="VARCHAR(255)" />
  <type-mapping type="java.lang.Float" persistence-type="float" />
  <type-mapping type="java.lang.Double" persistence-type="double" />
```

The diagram illustrates the flow from Hybris XML configuration to Java code generation and finally to the database schema.

- Hybris XML Configuration:** The leftmost window shows the file `core-items.xml`. A specific section is highlighted with a black box, containing the following code:
 

```

<itemtype code="Address"
          extends="GenericItem"
          jaloClass="de.hybris.platform.jalo.Address"
          autoCreate="true"
          generate="true">
      <deployment table="Addresses" typeCode="Address">
          <attributes>
              <attribute autoCreate="true" qualifiedName="PK" persistenceType="property"/>
          </attributes>
      </deployment>
      <!-- this attribute is to distinguish between Address and GenericItem -->
  
```

 A green callout box labeled "This is for Hybris Purpose" points to this section.
- Java Code Generation:** The middle window shows the generated Java class `AddressModel.java`. The same highlighted section from the XML is pasted into the code editor. A green callout box labeled "This is for DB Purpose" points to the `PK` attribute definition in the Java code.
- Database Schema:** The bottom right window shows the database table structure for `Addresses`. The table has the following columns:
 

COLUMN_...	TYPE_N...	IS_NULLA...	DECIMAL
HJMPTS	BIGINT	YES	0
CREATEDTS	TIMESTAM...	YES	<null>
MODIFIEDTS	TIMESTAM...	YES	<null>
TYPEPKST...	BIGINT	YES	0
OWNERPK...	BIGINT	YES	0
PK	BIGINT	NO	0
SEALED	TINYINT	YES	0

 A green callout box labeled "This is for DB Purpose" points to the `PK` column in the table structure.

## Scenario = Localized Attributes

```
<itemtype code="UserRight" extends="GenericItem" autocr...>
    <deployment table="UserRights" typecode="29"/>
    <attributes>
        <attribute autocr...>
            <persistence type="property"/>
        </attribute>
        <attribute autocr...>
            <qualifier>name</qualifier>
            <type>localized:java.lang.String</type>
        </attribute>
    </attributes>
</itemtype>
```

**Col** Col Name  
Itemtype Name This is for Hybris Purpose  
autocr... This is for DB Purpose

- Q: What is there in XXXModel.java file?**

  - 1) For every attribute there will be variable created.
  - 2) For every attribute there will be 1 setXXX() & 1 getXXX()
  - 3) If your attribute is of type Localized then ... U will have  
2 setXXX() methods & 2 getXXX() methods.

**Note:** - If you want above Cols (code & name) in your own table then use "deployment" tag.

Your Itemtype name & Deployment tag name can same.

If XXX deployment is having localized attributes then ... these will be stored in separate table.

**Example:** - In above case, code will be stored in "UserRights" table & name will be stored in "UserRightsLP" table.

When we do the Build (ant clean all) --  
For every itemtype, there will be Model  
class generated.

If Itemtype Name = XXX then Model class  
name = XXXModel.java.

**Using this Model class only we perform the DB Operation (Create ... Update ... Remove)**

## UserRightModel.java

— 1 —

## public class UserRightModel

{ String code: St

```
    setCode(==);      getCode();
    setName(==);     getName(loc);
    getName();
```

Let's say -- You started the Server.  
& Performed hAC - Update.

### In the DB -- UserRights

		Info	Columns	...
	COLUMN_...	TYPE_N...		
1	HUMPTS	BIGINT		
2	CREATEDTS	TIMESTA...		
3	MODIFIEDTS	TIMESTA...		
4	TYPEPKST...	BIGINT		
5	OWNERPR...	BIGINT		
6	PK	BIGINT		
7	SEALED	TINYINT		
8	P_CODE	VARCHAR		
9	ACLTS	PIGINIT		
10	USERRIGHTS			
11	USERRIGHTS29SN			
12	USERPROPS			
13	USERPROFILES			
14	USERGROUPSLP			
15	USERGROUPS5SN			
16	USERGROUPS			
17	USERPROPSPROS			
18	USERAUDIT6SN			
19	USERLOCK			

	Info	Columns	Row
	COLUMN_...	TYPE_N...	
1	ITEMPK	BIGINT	
2	ITEMTYPEPK	BIGINT	
3	LANGPK	BIGINT	
4	P_NAME	VARCHAR	

Ghada Leather Belt Women chip x + Apparel-uk (Site)

Not secure | ngstorefront/en/Categories/Access-women/Ghada-Leather-Belt-Women/p/300385272

9 In Stock

ADD TO BAG

PRODUCT DETAILS REVIEWS SHIPPING

Hello -- I am English

Ghada Leather Belt Women chip x + Apparel-de (Site)

Not secure | ngstorefront/de/Kategorien/Gürtel-Women/Ghada-Leather-Belt-Women/p/300385272

- 1 +

ZUM WARENKORB HINZUFÜGEN

PRODUKTDDETAILS EWERTUNGEN LIEFERUNG

Hello -- I am in German

Categories  
Product Variant Types  
Catalog Browser  
Catalogs  
Products  
Units  
Keywords  
Catalog Management Tools  
Classification Systems  
Multimedia  
User  
Order  
Price Settings  
Internationalization  
Marketing  
Cockpit  
Scripting  
WCMS  
Ticket System  
Base Commerce  
Deeplink URLs  
B2B Commerce

If you make a Col of t country specific values

If you make a Col of type localized, then it allows you manage the country specific values (these values used to display in Storefront).

## Scenario = Understand Localized \*LP tables data?

Studded Belt cardinal L [300416139] - Apparel Product Catalog : Online

**PK = 8796219211777**

PROPERTIES ATTRIBUTES BUNDLING CATEGORY SYSTEM PRICES MULTIMEDIA VARIANTS EXTENDED ATTRIBUTES REVIEWS

Identifier		Summary
en	Studded Belt cardinal L	en
es_CO		es_CO
in		in
pt		pt

Approval

approved

Flexible Query SQL Query Search result Execution statistics History

Direct SQL query

```
select * from productslp where itempk = '8796219211777'
```

Flexible Query SQL Query **Search result** Execution statistics History

Commit: OFF

Show 10 entries Search:

ITEMPK	ITEMTYPEPK	LANGPK	P_NAME	P_DESCRIPTION	P_MANUFACTURERTYPEDESCRIPTION	P_SEGMENT	P_ARTICLESTATUS	P_SUMMARY
8796219211777	8796103376978	8796093055008	Studded Belt cardinal L					

Showing 1 to 1 of 1 entries Previous 1 Next

Studded Belt cardinal L [300416139] - Apparel Product Catalog : Online

**PK = 8796219211777**

PROPERTIES ATTRIBUTES BUNDLING CATEGORY SYSTEM PRICES MULTIMEDIA VARIANTS EXTENDED ATTRIBUTES REVIEWS

Identifier		Summary
en	Studded Belt cardinal L	en This is Summ in EN
es_CO		es_CO
in		in
pt		pt

Approval

approved

Flexible Query SQL Query Search result Execution statistics History

**Direct SQL query**

```
select * from productslp where itempk = '8796219211777'
```

Flexible Query SQL Query **Search result** Execution statistics History

Commit: OFF

Show 10 entries

ITEMPK	ITEMTYPEPK	LANGPK	P_NAME	P_DESCRIPTION	P_MANUFACTURERTYPEDESCRIPTION	P_SEGMENT	P_ARTICLESTATUS	P_SUMMARY
8796219211777	8796103376978	8796093055008	Studded Belt cardinal L					This is Summ in EN

Showing 1 to 1 of 1 entries

Previous 1 Next

Studded Belt cardinal L [300416139] - Apparel Product Catalog : Online

PK = 8796219211777

REFRESH SAV

PROPERTIES ATTRIBUTES BUNDLING CATEGORY SYSTEM PRICES MULTIMEDIA VARIANTS EXTENDED ATTRIBUTES REVIEWS STOCK

**ESSENTIAL**

Identifier	Summary
en ↗ Studded Belt cardinal L	en This is Summ in EN
es_CO	es_CO
pt ↗ Name In PT ---	pt
fr	fr ↗ I am FR Summary
ru	ru

Approval

approved

Flexible Query SQL Query Search result Execution statistics History

**Direct SQL query**

```
select * from productslp where itempk = '8796219211777'
```

Flexible Query SQL Query **Search result** Execution statistics History

Commit: OFF

Show 10 entries

ITEMPK	ITEMTYPEPK	LANGPK	P_NAME	P_DESCRIPTION	P_MANUFACTURERTYPEDESCRIPTION	P_SEGMENT	P_ARTICLESTATUS	P_SUMMARY
8796219211777	8796103376978	8796093055008	Studded Belt cardinal L					This is Summ in EN
8796219211777	8796103376978	8796093186080						
8796219211777	8796103376978	879609317152	Name In PT ---					I am FR Summary

Showing 1 to 3 of 3 entries

Previous 1 Next

**Conclusion = For 1 Language – 1 Record is maintained.**

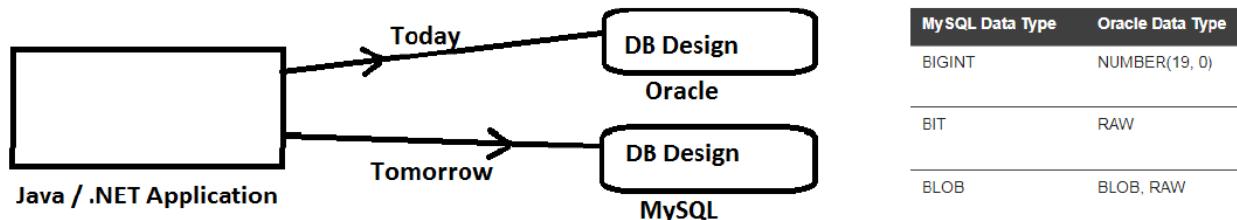
If 3 Languages are used then 3 records will be created.

**Contact Us = ChennaReddyTraining@RRRS.CO.IN**

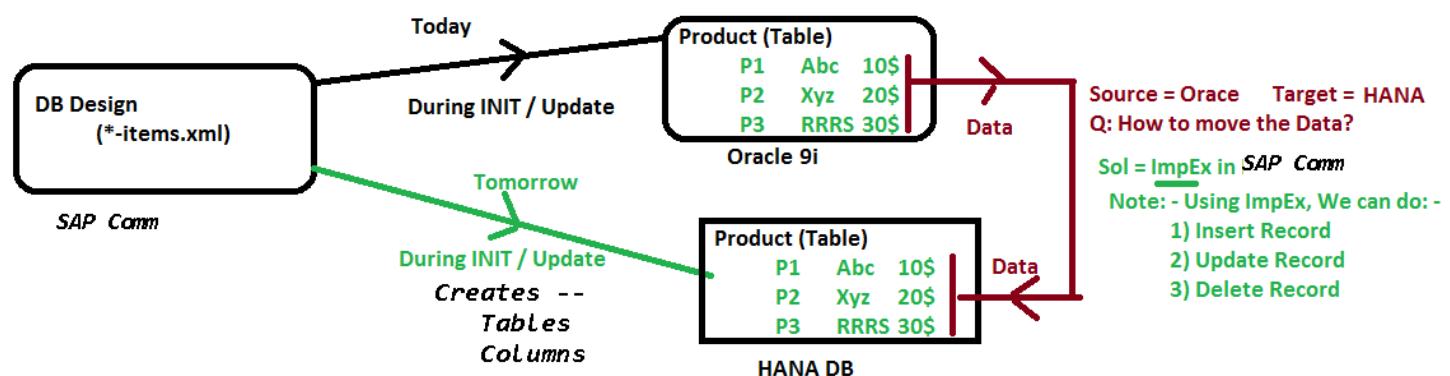
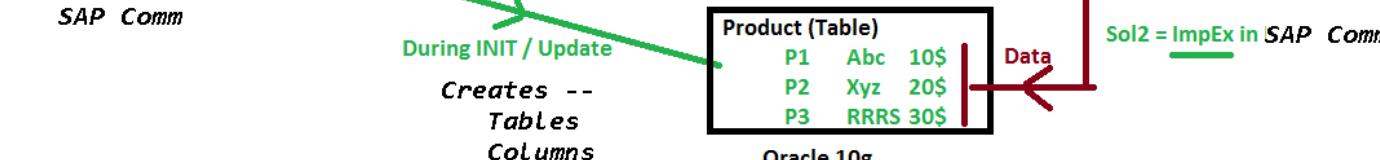
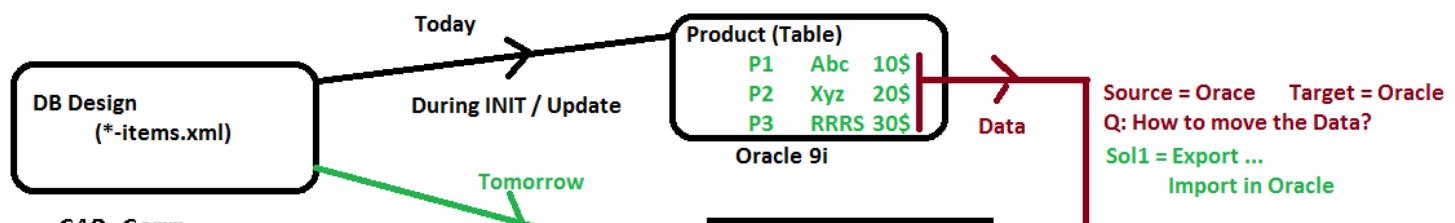
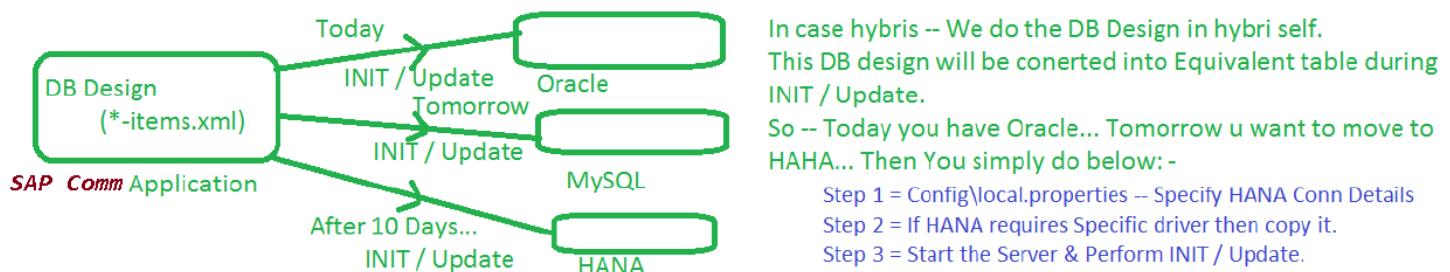
## Scenario = Data

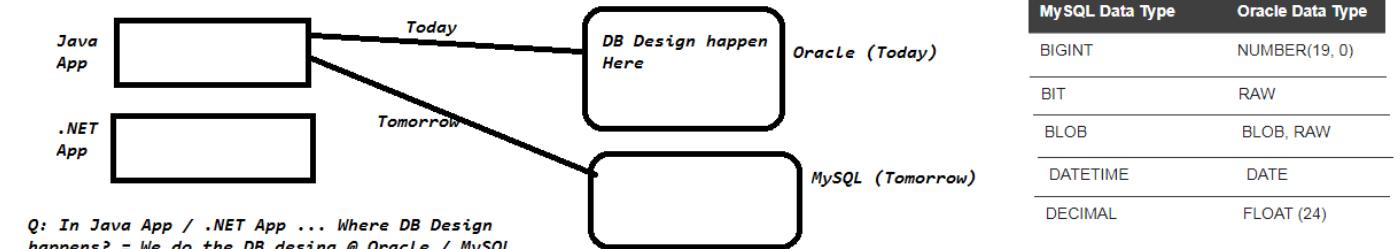
In case of “Java / .NET” application, DB design happens in native database (like – Oracle, MS SQL and ...).

In case of “SAP Comm”, DB design happens in SAP Comm itself. This DB design will be converted into equivalent native tables by “SAP Comm” while doing “INIT / Update”.

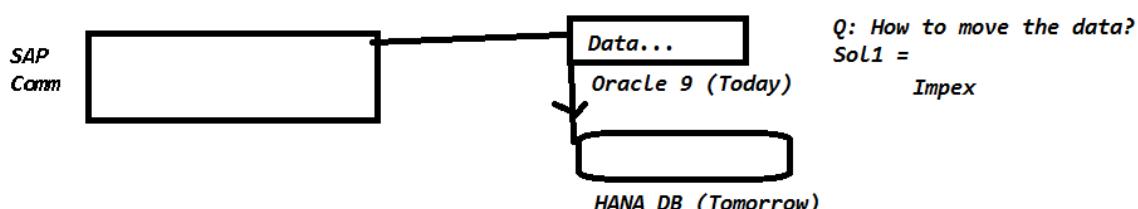
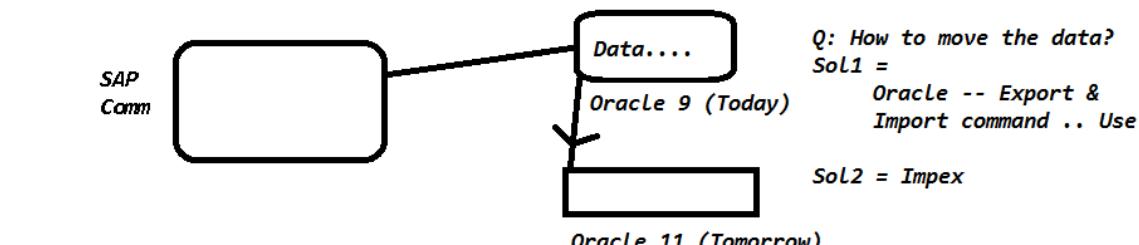
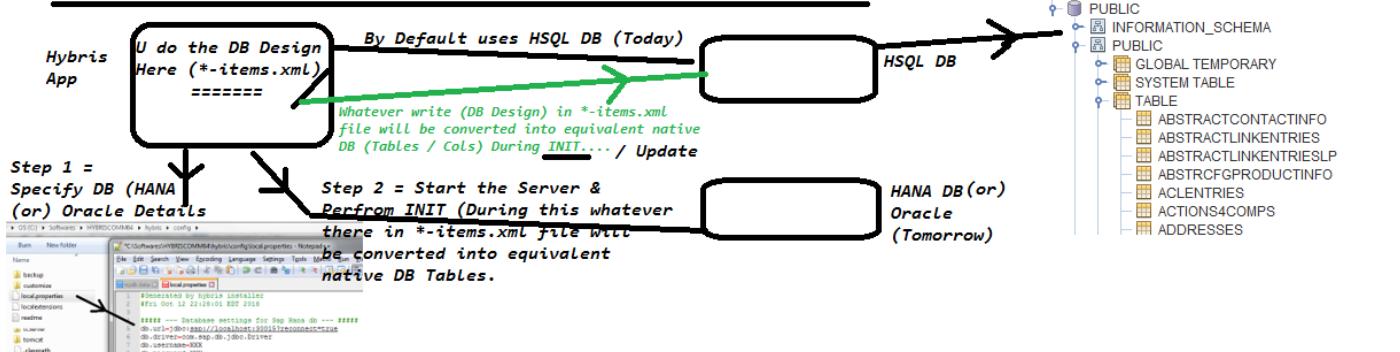


**Note:** - In case of Java / .NET, we do the DB design @ DB level. That's why moving from Oracle to MySQL / Some other DB is kind of project (3 months) ....

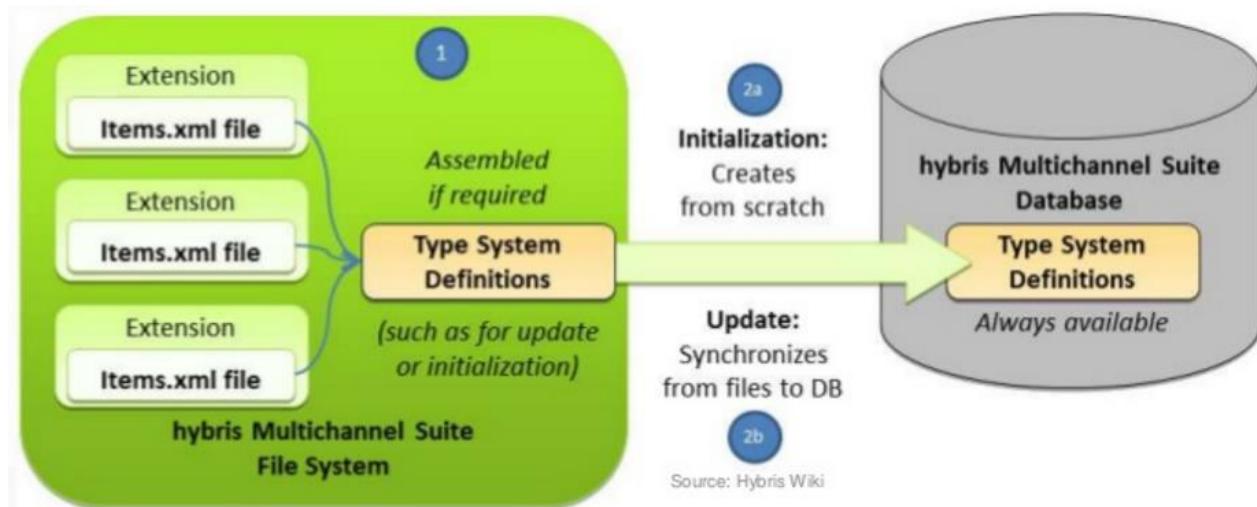




Note: - Tomorrow ... If we want to go with MySQL then we need to do DB Design again for MySQL ... Then Point Java App / .NET App to MySQL and Resolve all challenges..... It's kind of 2 / 3 resources ... 2 / 3 months project.



**Q = Who is giving / defining this XSD? = SAP Comm / FWK.**

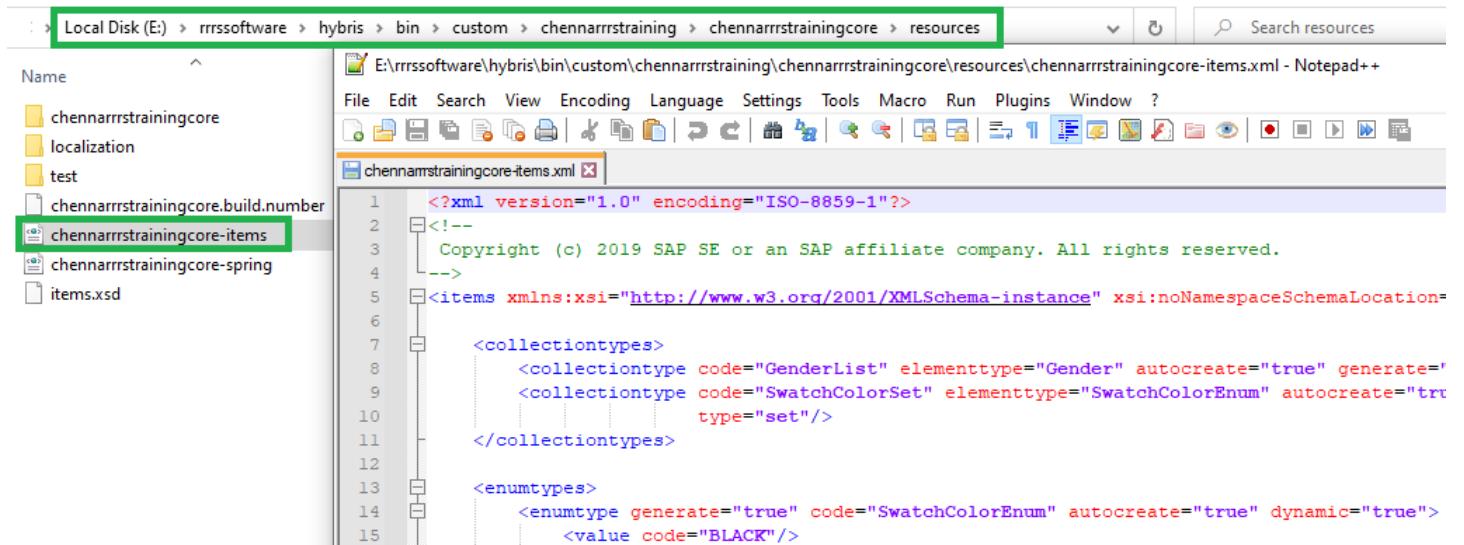


Contact Us = ChennaReddyTraining@RRRS.CO.IN

**Data Modeling** = A process of creating DB Structure & analyzing it for meeting the project requirements.

**Note** = Generally we do the data modeling in “\*core = chennarrstrainingcore” Ext.

In that Ext, File = \*core-items.xml (**chennarrstrainingcore-items.xml**).



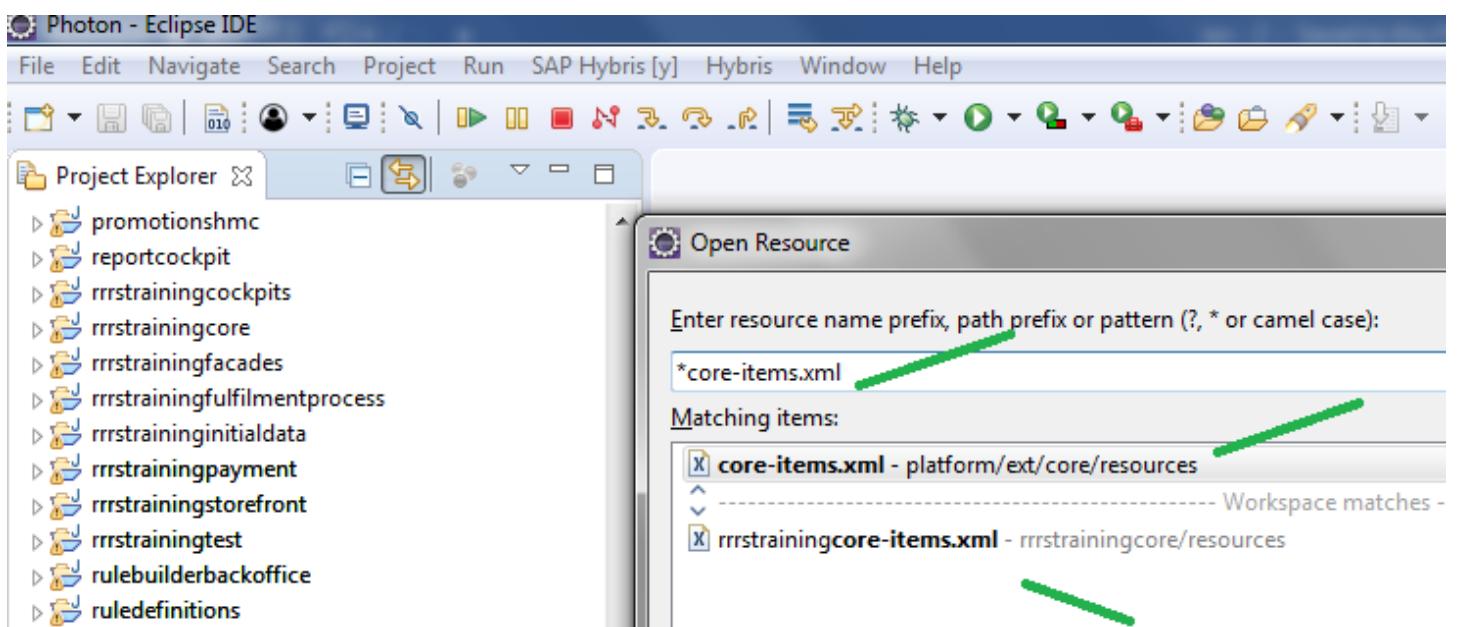
The screenshot shows a Windows file explorer window on the left and a Notepad++ editor window on the right. The file path in the file explorer is Local Disk (E) > rrrsssoftware > hybris > bin > custom > chennarrstraining > chennarrstrainingcore > resources. The Notepad++ window title is E:\rrrsssoftware\hybris\bin\custom\chennarrstraining\chennarrstrainingcore\resources\chennarrstrainingcore-items.xml - Notepad++. The code in the editor is:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--
Copyright (c) 2019 SAP SE or an SAP affiliate company. All rights reserved.
-->
<items xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="<!--
-->">
    <collectiontypes>
        <collectiontype code="GenderList" elementtype="Gender" autocreate="true" generate="true" type="set"/>
    </collectiontypes>
    <enumtypes>
        <enumtype generate="true" code="SwatchColorEnum" autocreate="true" dynamic="true">
            <value code="BLACK"/>
        </enumtype>
    </enumtypes>

```

**Q** = How to search for a file in Eclipse? = Open Eclipse .... CTRL + Shift + R

(OR) → Search → File



**Note** = As part of SAP Comm setup, we get lots predefined table (Carts, Product, Order...). For all these tables there are data model defined already.

**Q** = Where can we see the SAP Comm predefined tables data modeling?

**1) core-items.xml** = You can see data modeling for "Carts, PaymentInfo, OrderEntry, Users .....

**2) cms2-items.xml** = You can see data modeling for "ContentSlot, CMSComponentType.....

**3) catalog-items.xml** = You can see data modeling for "Catalog, Category, Product.....

**4) chennarrstrainingcore-items.xml** = We can use this for creating our own custom tables

5) .....

**Note** = SAP Comm Data Modeling can happen with Type systems.

\*-items.xml file contains 6 tags.

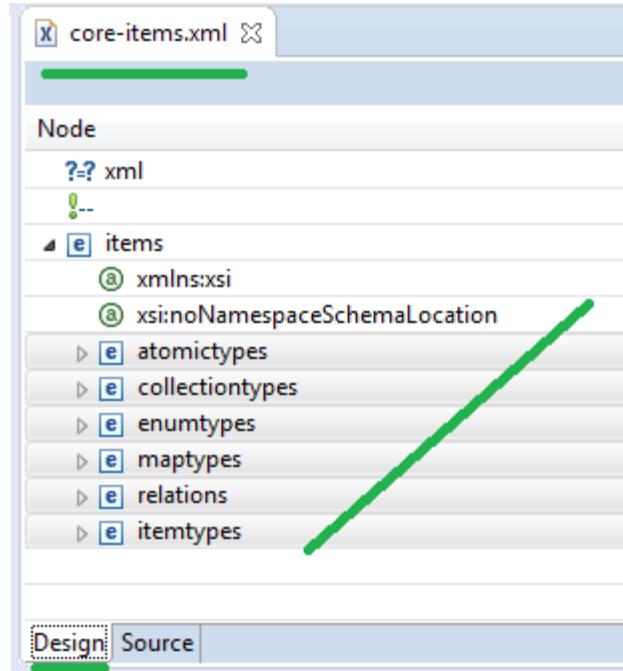
Means for data modeling we will use **these 6 tags**.

- |                 |                     |               |
|-----------------|---------------------|---------------|
| 1) Atomic types | 2) Collection types | 3) Enum types |
| 4) Map types    | 5) Relation types   | 6) Item types |

**Note** = These tags – Order is important & mandatory = **ACEMRI**

**Q** = Why we need to follow this order? =

We are doing the data modeling in **\*.xml file**. For every XML file, internally there will be **\*.xsd file**. This xsd file is having that kind of definition.



## 1) Atomictypes

**Q =** You take any language .... How many types data you handle?

20, 40 .... -20, -40 .... = Integers / Long

20.5 ... 40.5 ... -20.5... -40.5 .... = Float / Double

“a” ... “abc” ... “abc123” = Chars / Strings

True ... False ... = Boolean

**Note =** To handle these kinds of data, we use Atomic types.

Integer ... Byte ... Double .. Float .. Long ... String ... Object ...

**Note =** 20.5 ... -20.5 .... == All these are by default **double**

20.5F ... -20.5F .... == Now, these considered as **Float**.

## 2) Collection type

User Table

User ID	User Name
U01	One
U02	Two
U03	Three
U04	Four

Order Table

Order ID	Order Name	User ID
O101	ABC	U01
O102	BCD	U01
O103	CDE	U04
O104	DEF	U03
O105	EFG	U04
O106	FGH	U01

User Table

User ID	User Name	Order List
U01	One	O101,O102,O106
U02	Two	
U03	Three	O104
U04	Four	O103,O105

Q: How to find How many orders are placed by each user?

Select User.UserID, Count(Order.OrderID) from User, Order where User.UserID = Order.UserID group by Order.UserID;

Results: - U01 3  
U03 1  
U04 2

Q: Do you see any challenge with above Solution? =

Performance Issue. Bcoz, For every user, we are visiting / reading entire Order table.

Q: What is the Better Solution?

select OrderList from User;  
Apply Split Fun (or) Some Java Fun ... To find the count.

To Store these kind of data .... This Col needs to be of type Collection.  
Collection: - (1) List (or) (2) Set

If you want to allow duplicate values then go for List

If you don't want to allow duplicate values then go for Set.

```
<collectiontype code="LanguageList" elementtype="Language" autocreate="true" generate="true" type="List"/>
<collectiontype code="LanguageSet" elementtype="Language" autocreate="true" generate="true" type="set"/>

<collectiontype code="TypeCollection" elementtype="Type" autocreate="true" generate="false"/>
```

## Scenario = Collection Vs “1 – M Relation”

### Collection = 1 – M relation.

When to go for Collection & 1-M Relation?

Collection = Fast Retrieval. But 1-M = Slow retrieval.

Whenever collection size is very big then there is possibility of data truncation hence, in this case use relation.

**Note** = Too much normalization also not good. Decrease the performance.

Collection of String will contain the elements of String element type.

Collection of **Integer** will contain the elements of Integer type.

Collection of **Address** will contain the elements of Address type.

**Note:** - We know that **Collection** can be used as an **alternative** for **1 – M relation**.

	<b>Collection</b>	<b>Relation</b>
<b>Advantage</b>	Fast retrieval as it does not need any Join to get the results.	There is no chance of data truncation (no matter how big the size). It's <b>bidirectional</b> .
<b>Disadvantage</b>	If collection <b>size grows</b> there is a possibility of data truncation. It's <b>unidirectional</b> It's <b>not searchable</b> as there will be a list of PKs only present in the column.	<b>Slow</b> in retrieval due to join between the tables required
<b>When to Use</b>	When we are sure that in <b>our current &amp; future</b> requirements, we will not have many rows mapped for one side. It means <b>collection size</b> is small, Bcoz it helps to achieve <b>faster retrieval</b> .	Whenever <b>collection size is bigger</b> or there is a chance that it <b>can grow</b> bigger then <b>go with Relation</b> , bcoz there will be no data truncation. For <b>M - N</b> , we should go for Relation always.
<b>When not to Use</b>	Whenever the collection size is very big as it can lead to data truncation.	When <b>collection size is smaller</b> to compensate slow retrieval of Relation but in that case, we need to negotiate with <b>Bidirectional</b> mapping.

### 3) Enumtypes

If you have fixed number of values, then go for enum types.

Weekdays = {sun, mon, tue, wed ....}

Months = {jan, feb, mar, April ....}

```
<enumtype code="CreditCardType" autocreate="true" generate="true">
    <value code="amex"/>
    <value code="visa"/>
    <value code="master"/>
    <value code="diners"/>
</enumtype>

<enumtype code="OrderStatus" autocreate="true" generate="true" dynamic="true">
    <value code="CREATED"/>
    <value code="ON_VALIDATION"/>
    <value code="COMPLETED"/>
    <value code="CANCELLED"/>
</enumtype>
```

In C Language → Enum x = {abc, bcd, cde=-4, def, efg, fgh}

?x.abc = 0 ?x.bcd = 1 ?x.cde = -4 ?x.def = -3

**Q** = Create a col (attribute) for Gender. What is the type you consider = enum.

#### 4) Map types

This is used to store key – value Paris.

Each key is unique & values can be duplicate.

#### Emp (Table)

Empld	EmpName(Map types)
101	101 = 2020

**Q** = What happens if you insert duplicate key – values?

Nothing (No error).

Just new values will be replaced for the same key.

```
Map mymap = new HashMap();
```

```
mymap.put("1","one");
```

```
mymap.put("1","two"); = Method (m.put()) returns false.
```

```
System.out.println(mymap.get("1"))? = two.
```

#### 5) Relation types

Math's = 1-1, 1-N, N-1, M-N

Programming Lang = 1-1, 1-N, M-N

**Q** = What the relation between User table & Order table? = **1 – N**

**Q = What is the relation between User table & Address table? = 1 – N**

To define relation, we need to have “**Source & Target**” Elements.

We can define relation with “**cardinality**”.

```
<relation code="User2Addresses" generate="true" localized="false" autocreate="true">
    <sourceElement type="User" cardinality="one" qualifier="owner">
        <modifiers read="true" write="false" initial="true" search="true" removable="false" optional="false"/>
    </sourceElement>
    <targetElement type="Address" cardinality="many" qualifier="addresses">
        <modifiers read="true" write="true" search="true" optional="true" partof="true"/>
        <custom-properties>
            <property name="condition.query">
                <value>'{original} is null'</value>
            </property>
        </custom-properties>
    </targetElement>
</relation>
```

**Q = Min number of tables required to create 1 – 1 relation? = 1**

**Q = Min number of tables required to create 1 – N relation? = 2**

**Q = Min number of tables required to create M – N relation? = 3**

**Q = Explain <deployment> tag?**

If U want to keep attribute in your own table name, then specify deployment tag

Deployment tag – Will have **typecode**.

These typecode in Unique in SAP Comm System.

Range = 0 to 32767 (+Ve number only).

0 to 10000 = SAP Comm uses

If you create deployment tag then use **typecode >10000**

**Q = When we should define deployment mandatorily?**

1) Defining new item type by extending GenericItem

Example =

```
<itemtype code="MyItem" abstract="false" extends="GenericItem">
</itemtype>
```

2) Defining new item type by extending existing item type for which there is no deployment.

```
<itemtype code="AbstractOrder" extends="GenericItem" autocreate="true" generate="true" abstract="true">
<itemtype code="OneTwo" extends="AbstractOrder" autocreate="true" generate="true">
<deployment table="Carts" typecode="43"/>
```

## 6) Itemtypes

This is used to create Tables / Columns / Update existing tables (Customer... Cart... Product... Order...).

General Rule = Let's say – SAP Comm is giving 40 Tables.

**Q = Do you think that, for your project all 40 tables requires? =**

May not be. Let's say 1 table is not used for your project. Do not delete this unused table. Just leave like that.

Let's say – SAP Comm is giving 50 columns for Product table.

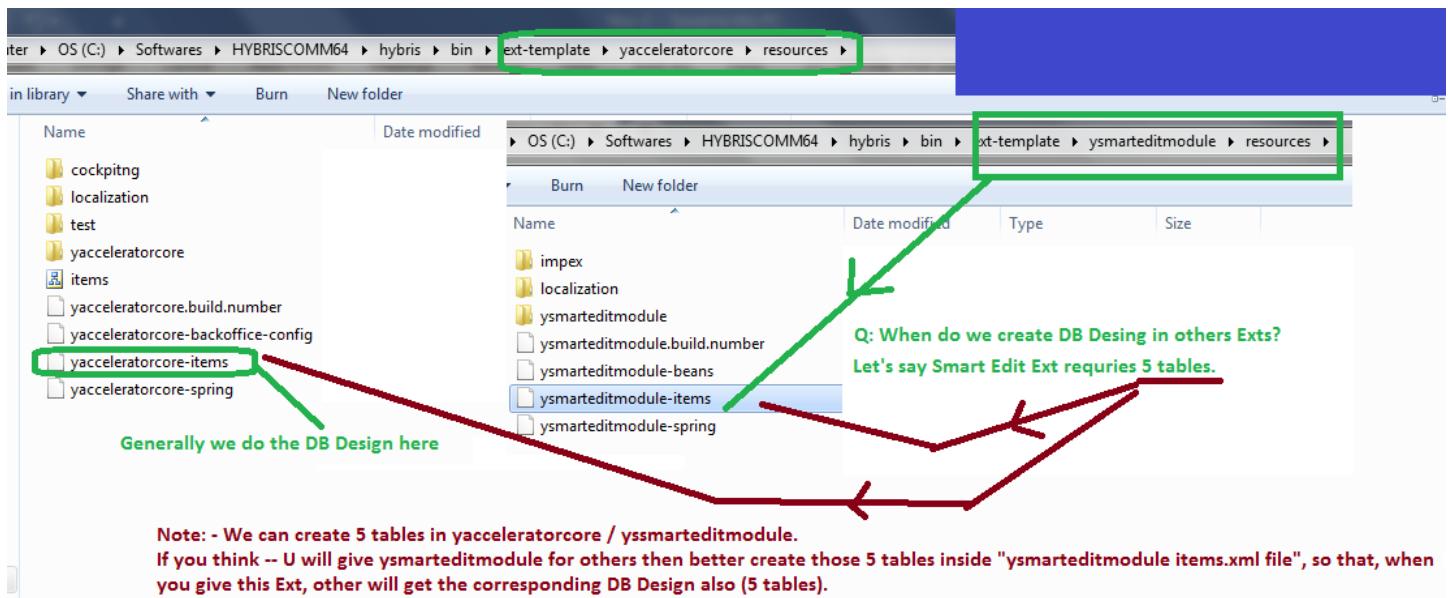
**Q = Do you think that, for your project all 50 columns requires? = May not be. Let's say 5 columns are not used for your project. Do not delete this unused Cols. Just leave like that.**

In SAP Comm – Data Modeling we do in “\*-items.xml”.

**Q = Which of the following is true about \*-items.xml?**

- Backbone for SAP Comm Data Modeling
- Every \*-items.xml file is validated against items.xsd
- It is not mandatory to have \*-items.xml in every Ext
- During the build for item type, model will be created.
- If we do something in \*-items.xml then for these DB entries will be created during INIT / Update
- Every attribute in \*-items.xml represents 1 table Col

**• All above**



**Note = Localization we do in 3 places: -**

- 1) **PIM** Localization = Example ... Product Description
- 2) **Content** Localization = Example ... Banner Description
- 3) **Programmatic** – Label localization ....

Note: - For localization values purpose, company may use 3rd party tools.

The screenshot illustrates the SAP Commerce multi-tenant and multi-language capabilities. It shows three storefronts for 'Apparel-uk (Site)', 'Apparel-de (Site)', and 'Apparel-ch (Site)'. The back-end interface on the right shows a product record with localized descriptions ('Hello -- I am English' for en and 'Hello -- I am German' for de). Below, a cart interface shows localized tax calculations. The code snippets in the bottom right show how these values are mapped from properties like 'base\_en.properties' and 'base\_de.properties' through XSLT templates like 'cartTotals.tag' to the final UI output.

## Q = Explain Important Features of “SAP Comm” Data Modeling?

1) Multi DB Support

2) DB specific attributes

```

<attribute type="localized:java.lang.String" qualifier="content">
  <persistence type="property">
    <columntype database="oracle"> <value>varchar2(4000)</value> </columntype>
    <columntype database="mysql"> <value>text</value> </columntype>
    <columntype database="sap"> <value>nvarchar(5000)</value> </columntype>
  </persistence>
  <modifiers search="false" />
</attribute>

```

3) Dynamic attributes

4) .....

Name	Date modified	Type	Size
chennarrstrainingcore			
localization			
test			
chennarrstrainingcore.build.number			
chennarrstrainingcore-items			
chennarrstrainingcore-spring			
items.xsd			

*Generally -- Here we do custom [Client Specific] Data Modeling*

**Q: Is this mandatory to do the data modeling every time in this Ext?**  
**It's not mandatory. Generally we do here.**

**Q: When to do the data modeling in other Exts (Outside of this Ext)?**

Let's say -- We have Abc Ext. To use this Ext assume that 5 tables are required.

In this case, define those 5 tables in Abc Ext only.

Example: - hMC ... We may use / we may not use.

If we don't use then we don't want create those tables.

If we use then only we want to create those tables.

```
<itemtype code="SearchRestriction"
  jaloClass="de.hybris.platform.jalo.type.SearchRestriction"
  extends="TypeManagerManaged"
  deployment="de.hybris.platform.persistence.type.SearchRestriction"
  generate="false"
  autoCreate="true">
```

*Old Version*

```
<itemtype code="UserRight"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.security.UserRight"
  autoCreate="true"
  generate="true">
<deployment table="UserRights" typeCode="29"/>
```

*New Version*

**Itemtype Name (Used in Hybris)**

```
<itemtype code="UserRight"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.security.UserRight"
  autoCreate="true"
  generate="true">
```

**autoCreate="true"** -- It creates DB entries during INIT.  
**autoCreate="false"** -- It understands that this item is already exist & will not create any Db entries.

**generate="true"** -- This generates the related classes for this item type.  
**generate="false"** -- It understands that already class exist & Just do the update of existing class.

```
<itemtype code="UserRight"
  extends="GenericItem"
  jaloClass="de.hybris.platform.jalo.security.UserRight"
  autoCreate="true"
  generate="true">
<deployment table="UserRights" typeCode="29"/>
<attributes>
  <attribute autoCreate="true" qualifier="code" type="java.lang.String">
    <persistence type="property"/>
    <modifiers read="true" write="true" search="true" initial="true" optional="false" unique="true"/>
  </attribute>
```

**Col Name**

In C = int x;	In Java = private int x;	Note: - Similarly, In Hybris when we create attributes, we specify the modifiers.
Q: How many types of Variables? 1) local 2) global 3) static 4) register	Q: How many types of modifiers? 1) private 2) public 3) protected 4) default	

Col

```

<attribute autodata="true" qualifier="code" type="java.lang.String">
    <persistence type="property"/>
        <modifiers read="true" write="true" search="true" initial="true" optional="false" unique="true"/>

```

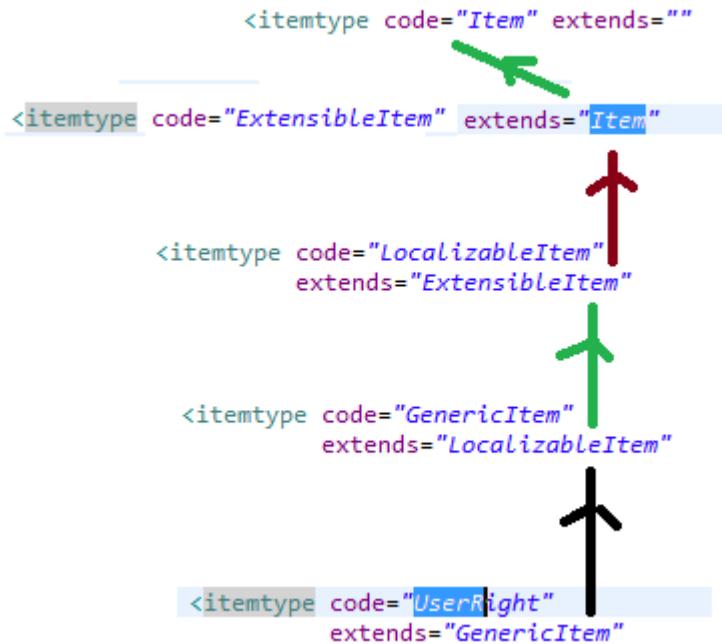
Allows attribute is searchable through queries

Mandatory

No Duplicates

**Note** = In Java – For every class – What is super class = **Object**.

In SAP Comm – For every item type – What is super item type = Generic Item / Item.



**Note** = In Oracle DB – We can create the PK.

**Q1** = Can we create more than 1 PK in same table? = Yes.

**Q2** = How many PK we can create max in 1 table? = 16

**Note** = Creating too many PK's in same table decreases the performance.

Q3 = Can we create PK with more than 1 Col = Yes. == **Composite Key**

Q4 = A Composite key can have max how many Cols? = 16

====> PK = Unique + Index

**Q** = How to create 1 Col Indexed?

```
<indexes>
  <index name="UID" unique="true">
    <key attribute="uid"/>
  </index>
</indexes>
```

Index Name  
Col Name

**Q** = How to create >1 col indexed?

```
<indexes>
  <index name="version_idx" unique="true">
    <key attribute="code"/>
    <key attribute="version"/>
  </index>
</indexes>
```

Col1  
Col2

**Q** = How to specify the default values for attribute (Col)?

```
<attribute autoreate="true" qualifier="subtotal" type="java.lang.Double" generate="true">
  <persistence type="property">
    <columntype>
      <value>java.math.BigDecimal</value>
    </columntype>
  </persistence>
  <modifiers read="true" write="true" search="true" optional="true"/>
  <defaultValue>Double.valueOf(0.0d)</defaultValue>
</attribute>
```

**<custom-properties>** = If you want to search on particular columns in hMC then put that column in custom properties. For PK columns, this will be created by default.

**Note =**

1) If you create your own itemtype & it's not extending any other Itemtypes, then make min 1 Col unique = true. == **This improve the performance.**

## 2) If relation is defined with M – N then avoid sort option == Improve Performance

The diagram illustrates the configuration of item types and their database mappings. It shows two XML files: `core-items.xml` and `*core-advanced-deployment.xml`.

**core-items.xml:**

```
<itemtype code="UserRight" extends="GenericItem" jaloClass="de.hybris.platform.jalo.security.UserRight" autoCreate="true" generate="true">
<deployment table="UserRights" typeCode="29"/>
<attributes>
    <attribute autocreate="true" qualifier="code" type="java.lang.String">
```

A green arrow points from this code to a green box labeled "After Update / INIT". A red arrow points from the same code to a red box labeled "Q: How Hybris creating 255 char length in Oracle for String?".

**\*core-advanced-deployment.xml:**

```
17 @version , $Date: 2007-05-16 16:15:55 +0200 (Mi, 16 Mai 2007) $
18 -->
19
20<model name="hybris" description="...">
21
22<database-schema database="sap" primary-key="primary key" null="" not-null="not null" >
23     <type-mapping type="java.lang.String" persistence-type="nvarchar(255)" />
24
25</database-schema>
26
27<database-schema database="oracle" primary-key="primary key" null="" not-null="not null" >
28     <type-mapping type="java.lang.String" persistence-type="varchar2(255 CHAR)" />
29 ..
```

A green arrow points from the `core-items.xml` code to the `*core-advanced-deployment.xml` file. A red X is drawn over the `core-items.xml` file, with the text "This file is having the DB specific mappings. Hybris uses this." written next to it.

**Annotations:**

- DB Col Name:** Points to the `code` attribute in `core-items.xml`.
- type = property (or) dynamic:** Points to the `persistence` and `modifiers` sections in `core-items.xml`.
- If property then you can save the data in DB Col.** Points to the `persistence` section.
- If dynamic then you will not able to save data into this Col.** Points to the `modifiers` section.

## Q = What are the different ways to create <Itemtypes>

- Define new item type **without extending any existing item type**

**Example =**

```
<itemtype code="RRRS" autoCreate="true" generate="true">
<deployment table="RRRS" typeCode="11000" />
```

- Define the new item type **by extending** it with **existing** item type

## **Example =**

```
<itemtype code="AProduct" extends="Product" autocreate="true"  
generate="true" jaloclass="com.chenna.rrrs.ApparelProduct">
```

## <attributes>

- Define the **existing** item type **again** with new attributes

## **Example =**

```
<itemtype code="Address" autocreate="false" generate="false">  
<attributes>  
    <attribute qualifier="permanentAddress" type="java.lang.Boolean">  
        <mtype code="UserRight"  
              extends="GenericItem"  
              jaloClass="de.hybris.platform.jalo.security.UserRight"  
              autocreate="true" />  
        <generate>true</generate>  
        <deployment table="UserRights" typeCode="29"/>  
<attributes>
```

**autocreate=true** at the item type level

It hints “SAP Comm” to create a new database entry for this type at initialization/update process. If we set it to false, build will fail.

We should set it to true for the first definition of item type.

**generate=true** at the item type level

It hints “SAP Comm” to generate a new jalo class for this type during build time. If we set it to false, then jalo class will not be generated however model class will always be generated.

We should set it to true for the first definition of item type.

```

<itemtype code="UserRight"
  extends="GenericItem"
  jaloclass="de.hybris.platform.jalo.security.UserRight"
  autocreate="true"
  generate="true">
<deployment table="UserRights" typecode="29"/>
<attributes>
  <attribute autocreate="true" qualifier="code" type="java.lang.String">
    <persistence type="property"/>
    <modifiers read="true" write="true" search="true" initial="true" optional="false" unique="true"/>
  </custom-properties>

```

**read = true** → attribute is readable and corresponding **getter** method will be generated for the attribute. Default value is true.

**write=true** → attribute is writable and corresponding **setter** method will be generated for the attribute. Default value is true.

**search=true** → attribute can be **searchable** by a FlexibleSearch. Default value is true.

**Optional=true** → attribute is not mandatory. Default value is true.

To make any attribute as mandatory, set optional as false

**initial = "true"** → We can give value while creation of the row itself. It can be changed only once. That is 1st time itself. Attribute is writable only at INIT time?

If 'true', the attribute will only be writable during the item creation.

Setting to 'true' is useful in combination with write='false'. Default is 'false'.

**unique = "true"** → It creates primary key column if unique = "true".

If 'true', the value of this attribute has to be unique within all instances of this type.

If multiple attributes marked as unique, then their combined values must be unique.

**Persistent** type for attributes =

Persistent type can be either property or dynamic.

If it is set as **property** then value will be stored in DB and it's called persistent attribute.

If it is set as **dynamic** then value will not be stored in DB and it's called dynamic attribute.

## **Q = Dynamic in Enum**

Dynamic in **Enum** is completely **different** from **Dynamic attributes**.

If an Enum type is dynamic then values can be added at the run time.

We can make Enum type as Dynamic by specifying `dynamic=true` in the Enum type definition.

If an Enum type is non-dynamic (by default, `dynamic="false"`) we are not allowed to add new values at runtime.

If we add any non-dynamic Enum type without values, build will fail as it does not have any effect.

So if you want to add new values at run time we have to make **dynamic="true"** for an Enum type.

We can change the **flag anytime** but enforces a **system update**.

If **dynamic="false"** → The servicelayer generates real java enums (having a fixed set of values).

If **dynamic="true"** → It generates “SAP Comm” enums which can be used without fixed values(means we can add run time values).

## Q = Defining Relation in items.xml in “SAP Comm”?

### Example = 1 – M (or) M – 1

```
<relation code="User2ContactInfos" generate="true" localized="false" autocreate="true">
    <sourceElement type="User" cardinality="one" qualifier="user">
        <modifiers read="true" write="true" search="true" optional="false"/>
    </sourceElement>
    <targetElement type="AbstractContactInfo" cardinality="many" qualifier="contactInfos" ordered="true">
        <modifiers read="true" write="true" search="true" optional="true" partof="true"/>
    </targetElement>
</relation>
```

We can see below entries in model class after build

UserModel.java

```
1. private Collection<AbstractContactInfoModel> _contactInfos;
```

AbstractContactInfoModel.java

```
1. private UserModel _user;
```

*Note = Since its a "1- M", we can observe that Collection is generated in UserModel class and just single User entity is generated in AbstractContactInfoModel class*

### Example = M – N

Consider the relation between **Product** and **Categories**.

One product can belong to many categories

And one category can contain many products.

```
<!-- category relations -->
<relation code="CategoryProductRelation" autocreate="true" generate="true" localized="false">
    <deployment table="Cat2ProdRel" typecode="143"/>
    <sourceElement qualifier="superCategories" type="Category" cardinality="many" ordered="false">
        <description>Super Categories</description>
        <modifiers read="true" write="true" search="true" optional="true"/>
    </sourceElement>
    <targetElement qualifier="products" type="Product" cardinality="many" collectiontype="list" ordered="true">
        <description>Products</description>
        <modifiers read="true" write="true" search="true" optional="true"/>
    </targetElement>
</relation>
```

We can see qualifier as **superCategories** in the **source** which is of type **Category**.

We can see qualifier as **products** in the **target** which is of type **Product**.

It means Inside a **Category** it creates a variable called **products** and inside a **Product** it creates a variable called **superCategories**.

**Generate** will have **no effect** for relation.

**Autocreate true** makes the **relation type** to be created.

We have specified **collection type** as **List** for **target**.

After the build, we can see below entries in java class

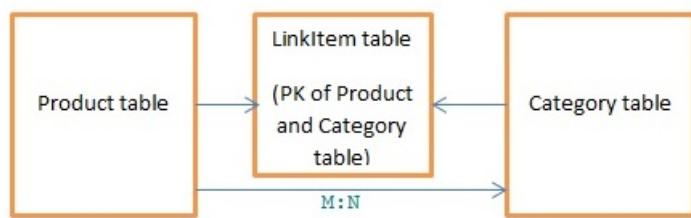
CategoryModel.java

```
1. private List<ProductModel> _products;
```

ProductModel.java

```
1. private Collection<CategoryModel> _supercategories;
```

How Relation works in backend?



One table will be created for Products.  
One table will be created for Categories.  
One extra table will be created for LinkItem (elements on both sides of the relation are linked together via instances of a LinkItem table).  
Each LinkItem instance stores the PKs of the related items for each row.

**Note** = Every row of product with associated category will have one row in the LinkItem table with PK of Catrgory and associated PK of Product.

LinkItem instances are used internally by “SAP Comm”.

We just need to call getters & setters at API level to set & get the values.

**Extra table** will be created only for many to many relations

**Q** = Explain “Redeclare” in **items.xml**?

In Java , we have a concept called variable hiding which means variable with the same name is defined in both parent and child classes.

In such cases, variable from Parent will be inherited but it will be hidden in the Child class as the Child class also has the same variable.

we can also change the variable data type in Child class keeping the same variable name.

Similar to Variable Hiding in Java, we have a concept called “Re declaring” the attributes” in SAP Comm.

Re declaring the attributes in “SAP Comm”.

Sometimes it is required to re declare the attribute in the child item type for various reasons.

It could be to change the Data type of an attribute or make an attribute as read only etc.

So We can re declare the same attributes in the Child item type to change such behavior

**Examples =**

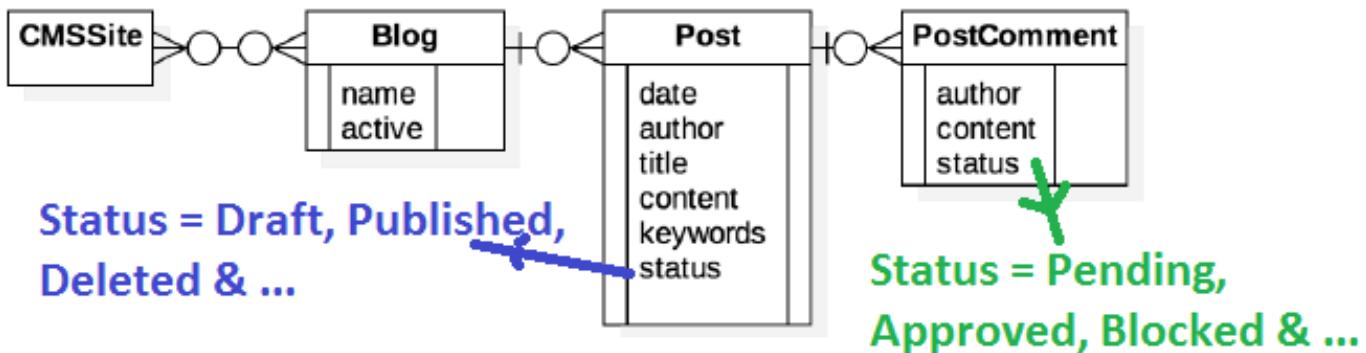
We can change the modifier from **read=true** to **read=false**

We can make attribute as **unique**.

We can add **write=false**.

We can also change the type of the attribute but only to **subtypes**.

## Business Scenario



## Solution =

### Step 1 = Create CMSSite

“SAP Comm” already giving this for us. So no need to write any code.

```
cms2-items.xml
921
922<itemtype code="CMSSite" jaloClass="de.hybris.platform.cms2.jalo.s
923 generate="true">
924 <!-- deployment table="CMSSite" typecode="1064" / -->
925<attributes>
926
927<attribute qualifier="urlPatterns" type="StringCollection"
```

### Step 2 = Create 2 Enum Types

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer** (left): Shows the project structure for **chennarrstrainingcore**, including JRE System Library [jdk-11.0.10], Referenced Libraries, src, testsrc, gensrc, classes, lib, and resources.
- chennarrstrainingcore-items.xml** (right): The XML configuration file for the project.

The XML content in the file is as follows:

```
<!-- Example 1 - Step 1 = Create ENUM Types = RRRSPostStatus & ChennaCommentStatus -->
<enumtype code="RRRSPostStatus">
    <value code="DRAFT"/>
    <value code="PUBLISHED"/>
    <value code="DELETED"/>
</enumtype>
<enumtype code="RRRSCommentStatus">
    <value code="PENDING"/>
    <value code="APPROVED"/>
    <value code="BLOCKED"/>
</enumtype>
</enumtypes>
<!-- Example 1 - Step 3 = Define Your Relationship -->
<relations>
    <relation localized="false" code="RRRSBlogCMSSite">
```

## Step 3 = Create “Blog ... Post ... PostComment” tables

```
*chennarrstrainingcore-items.xml ✘
56
57      <!-- Example 1 - Step 2 = Add your item definitions here = RRRSBlog, RRRSPost & RRRSPostComment -->
58
59<itemtype code="RRRSBlog" extends="GenericItem" autocreate="true" generate="true">
60    <deployment table="chennablog" typecode="11001" />
61    <attributes>
62        <attribute qualifier="name" type="java.lang.String">
63            <modifiers initial="true" read="true" write="false" optional="false" unique="true" />
64            <persistence type="property" />
65        </attribute>
66        <attribute qualifier="active" type="boolean">
67            <persistence type="property" />
68        </attribute>
69    </attributes>
70</itemtype>

chennarrstrainingcore-items.xml ✘
<itemtype code="RRRSPost" extends="GenericItem" autocreate="true" generate="true">
    <deployment table="chennapost" typecode="11002" />
    <attributes>
        <attribute qualifier="blog" type="RRRSBlog">
            <persistence type="property" />
        </attribute>
        <attribute qualifier="date" type="java.util.Date">
            <modifiers optional="false" />
            <persistence type="property" />
        </attribute>
        <attribute qualifier="author" type="Employee">
            <modifiers optional="false" />
            <persistence type="property" />
        </attribute>
        <attribute qualifier="title" type="localized:java.lang.String">
            <persistence type="property" />
        </attribute>
        <attribute qualifier="content" type="localized:java.lang.String">
            <persistence type="property">
                <columntype>
                    <value>HYBRIS.LONG_STRING</value>
                </columntype>
            </persistence>
            <modifiers search="false" />
        </attribute>
        <attribute qualifier="keywords" type="StringCollection">
            <persistence type="property" />
        </attribute>
        <attribute qualifier="status" type="RRRSPostStatus">
            <persistence type="property" />
            <defaultvalue>em().getEnumerationValue("RRRSPostStatus", "DRAFT")</defaultvalue>
        </attribute>
    </attributes>
</itemtype>
```

`<itemtype code="Employee"
extends="User"
jaloClass="de.hybris.platform.jalo.user.Employee"
autocreate="true"
generate="true">`

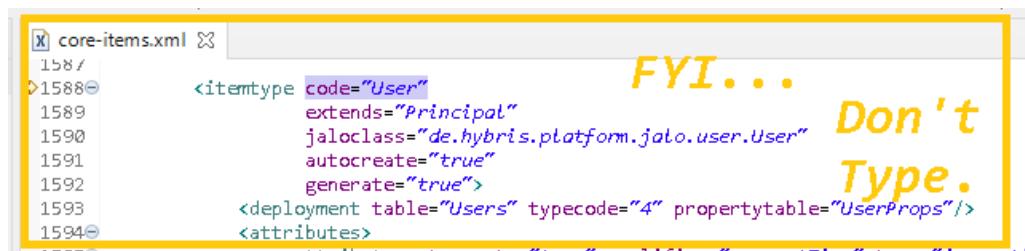
*Don't type*

*FYI...*

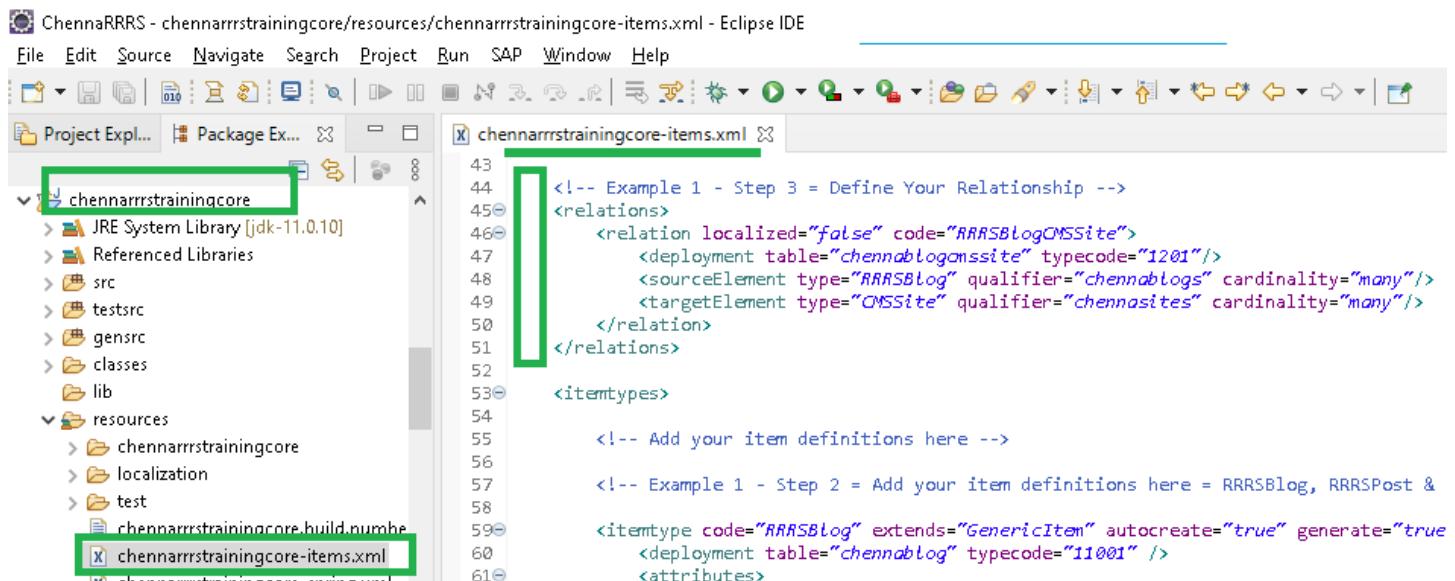
```

<itemtype code="RRRSPostComment" extends="GenericItem" autocreate="true" generate="true">
    <deployment table="rrrscomment" typecode="11003" />
    <attributes>
        <attribute qualifier="author" type="User">
            <persistence type="property" />
        </attribute>
        <attribute qualifier="content" type="java.lang.String">
            <persistence type="property" />
            <columntype>
                <value>HYBRIS.LONG_STRING</value>
            </columntype>
            </persistence>
            <modifiers search="false" optional="false" />
        </attribute>
        <attribute qualifier="status" type="RRRSCommentStatus">
            <persistence type="property" />
        </attribute>
    </attributes>
</itemtype>

```



## Step 4 = Create Relation



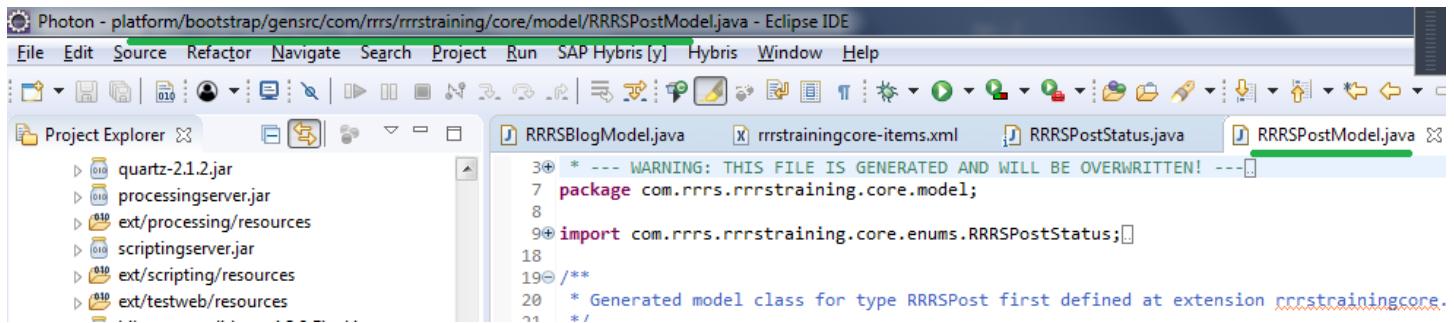
## Step 5 = Do the build [ant clean all]

E:\rrrssoftware\hybris\bin\platform>**ant clean all**

Q = What happens during the build?

1 = For every **itemtype – Model** class will be generated.

If itemtype name = XXX then generated model class = **XXXModel.java**.



The screenshot shows the Eclipse IDE interface with the title "Photon - platform/bootstrap/gensrc/com/rrrs/rrrstraining/core/model/RRRSPostModel.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, SAP Hybris [y], Hybris, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left lists several JAR files: quartz-2.1.2.jar, processingserver.jar, ext/processing/resources, scriptingserver.jar, ext/scripting/resources, and ext/testweb/resources. The code editor on the right displays the RRRSPostModel.java file:

```
3④ * --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN! ---⑤
7 package com.rrrs.rrrstraining.core.model;
8
9④ import com.rrrs.rrrstraining.core.enums.RRRSPostStatus;⑤
18
19④ /**
20  * Generated model class for type RRRSPost first defined at extension rrrstrainingcore.
21  */
```

2 = All the model classes are generated in “**platform**” folder.

With the help of model classes only we can able to the “Insert / Update / Remove / Select (**CURD**)” operation on DB – Table.

Whatever available in platform is applicable for all the Exts.

3 = If itemtype is having n attributes, then what model class will have: -

- a. For each attribute there will be **1 variable**
- b. For each attribute there will be **1 setter & 1 getter**
- c. If attribute is **localized** – there will be **2 setters & 2 getter** method

4 = If itemtype is having boolean attribute, then – there will be 1 variable, 1 setXXX() method & isXXX() method (No getXXX() method in this case).

```

lodel.java rrstrainingcore-items.xml RRRSPostStatus.java RRRSBlogModel.java rrstrainingcore-items.xml RRRSPostStatus.java
<itemtype code="RRRSBlog" extends="GenericItem" autoreate="true
  <deployment table="RRRSblog" typecode="11001" />
  <attributes>
    <attribute qualifier="name" type="java.lang.String">
      <modifiers initial="true" read="true" write="false"
      <persistence type="property" />
    </attribute>
    <attribute qualifier="active" type="boolean">
      <persistence type="property" />
    </attribute>
  </attributes>
</itemtype>

```

```

100 | @Accessor(qualifier = "active", type = Accessor.Type.GETTER
101@ public boolean isActive()
102  {
103    return toPrimitive((Boolean) getPersistenceContext().getP
104  }
105
106@ @Accessor(qualifier = "active", type = Accessor.Type.SETTER
107@ public void setActive(final boolean value)
108  {
109    getPersistenceContext().setPropertyValue(APACTIVE, toObjec
110  }
111
112}

```

**5 = For every enum type – there will be 1 enum type class generated.**

If enum type name = XXX then generated enum type class = **XXX.java**

```

RRRSBlogModel.java rrstrainingcore-items.xml RRRSPostStatus.java RRRSBlogModel.java rrstrainingcore-items.xml RRRSPostStatus.java
6  * Generated enum RRRSPostStatus declared at extension rrstrainingcore
7  */
8  @SuppressWarnings("PMD")
9  public enum RRRSPostStatus implements HybrisEnumValue
10 {
11   DRAFT("DRAFT"),
12   PUBLISHED("PUBLISHED"),
13   DELETED("DELETED");
14   public final static String TYPECODE = "RRRSPostStatus";
15   public final static String SIMPLE_CLASSNAME = "RRRSPostStatus";
16   private final String code;
17   private RRRSPostStatus(final String code)
18   {
19     this.code = code.intern();
20   }
21   public String getCode()
22   {
23     return this.code;
24   }
25   public String getType()
26   {
27     return SIMPLE_CLASSNAME;
28   }
29 }

```

```

RRRSBlogModel.java rrstrainingcore-items.xml
26
27
28
29
30
31

```

## Step 6 = Start the Hybris Server

E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

## Step 7 = Perform Platform Update

hAC = <https://localhost:9002> (admin admin1234)

Platform → Update

## Step 8 = Results

SAP CX Backoffice    Apparel Site UK | Homepage    Not secure | localhost:9002/backoffice/

**Administration Cockpit**

types

- System
- Types
- Search and Navigation
- Index Types
- Solr Facet Search Configuration
- Indexed Types
- Catalog
- Product Variant Types

RRRS

SEARCH

Identifier	Name	Extensionname
RRRSBlog	chennarrstrainingcore	
RRRSBlogCMSSite	chennarrstrainingcore	
RRRSBlogCMSSitechennablogsColl	chennarrstrainingcore	
RRRSBlogCMSSitechennasitesColl	chennarrstrainingcore	
RRRSCommentStatus	chennarrstrainingcore	
RRRSPost	chennarrstrainingcore	

**Insert Record**

Create New RRRSPostComment

MANDATORY VALUES

[content]:  
Hello -- This is my 1st Blog. Please take a time to read it.

Time created:  
May 31, 2021 2:43:50 AM

====> DONE

Display Records

RRRSPostComment

COMMON PROPERTIES XML REPRESENTATION

Record

RRRSPostComment.auth...	RRRSPostComment.content	Time creat...	Type	Time modifi...	Own...	PK
	Hello -- This is my 1st Blog. Please take a time to read it.	May 31, 2021 2:43:50 AM	RRRSPostComment	May 31, 2021		

**hybris administration console**

You're Administrator logout

Platform Monitoring Maintenance Console

Scripting Languages FlexibleSearch ImpEx Import ImpEx Export LDAP

Flexible Query SQL Query Search result

FlexibleSearch query  
select \* from {rrrspostcomment}

Commit: OFF Show 10 entries

HJMPTS	CREATEDTS	MODIFIEDTS	TYPEPKSTRING	PK	P_CONTENT	P_STATUS	ACLTS
0	2021-05-31 02:43:50.0	2021-05-31 02:44:12.018	8708147712082	8708093065070	Hello -- This is my 1st Blog. Please take a time to read it.		0

Contact Us = ChennaReddyTraining@RRRS.CO.IN

COLUMN_NAME	TYPE_NAME	IS_NULLABLE	DECIMAL_DIGITS	COLUMN_SIZE
HJMPTS	BIGINT	YES	0	64
CREATEDTS	TIMESTAMPTZ	YES	<null>	26
MODIFIEDTS	TIMESTAMPTZ	YES	<null>	26
TYPEPKST	BIGINT	YES	0	64
OWNERPK	BIGINT	YES	0	64
PK	BIGINT	NO	0	64
P_NAME	VARCHAR	YES	<null>	255
P_ACTIVE	TINYINT	YES	0	8

**Note =** If you observe above, there is no “RRRSBlogLP”, but there is “RRRSPostLP”.

**RRRSPostLP** is created because, in “**RRRSPost**” is having localized attributes.

So, localized attributes will be stored in different type = **RRRSPostLP**.

There is no ChennaBlogLP table created.  
But for ChennaPost there is ChennaPostLP table created.  
Note: - ChennaBlog is not having any localized attributes, hence ChennaBlogLP is not created.  
But, ChennaPost is having localized attributes, hence ChennaPostLP is created.  
So -- Localized attributes will be created in new table with ends with LP.

COLUMN_NAME	TYPE_NAME	IS_NULLABLE	DECIMAL_DIGITS	COLUMN_SIZE
ITEMPK	BIGINT	YES	0	64
ITEMTYPEPK	BIGINT	YES	0	64
LANGPK	BIGINT	YES	0	64
P_TITLE	VARCHAR	YES	<null>	255
P_CONTENT	VARCHAR	YES	<null>	167772

Inbox

- System
- Catalog**
- Catalogs
- Catalog Versions
- Categories
- Products

SAVED QUERIES

No queries

Description	Sales unit	Summary
Format	Piece [pieces]	en Heavy-Duty Compact
		es_CO
		in

**Localized Attribute**

## Scenario = Extend existing itemtype

The screenshot shows the SAP Administration Cockpit interface. On the left, the navigation bar includes Home, Inbox, System, Catalog (with Catalogs and Catalog Versions), Categories, and Products (circled in green). The main area displays a product creation form for "Snowboard Ski Tool Toko Side Edge Angle Pro 88 Grad [29531] - Apparel Product Catalog : Staged". The form has tabs for PROPERTIES, ATTRIBUTES, BUNDLING, and CATEGORIES. Under PROPERTIES, there are fields for Article Number (29531) and Identifier (Snowboard Ski T). A modal window titled "Column:" lists various columns from a database table, with "PRODUCTS" highlighted. To the right, a detailed table view shows columns like COLUMN\_NAME, TYPE\_N..., HJMPTS, CREATEDTS, MODIFIEDTS, etc., with several columns circled in green. A yellow "SEARCH" button is at the top right of the search bar.

“SAP Comm” already providing the **Product** table.

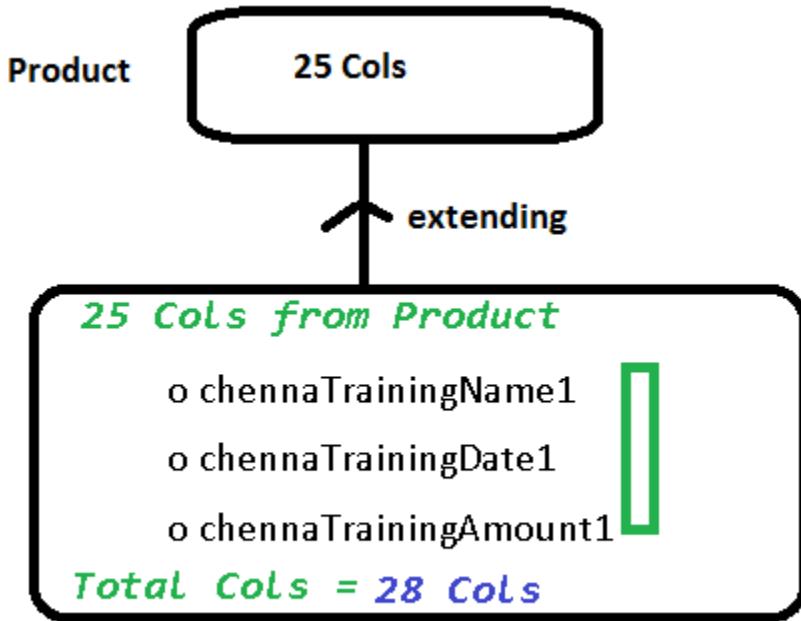
Assume that “SAP Comm” providing table is having **25 Cols**. Do you think that, this 25 Cols are sufficient for your company = May (or) May not.

Assume that your company requirement is, add **another 3 Cols**.

**Requirement** = We want to create new itemtype called **“ChennaTrainingProduct1”** by extending **“Product”** and

in new itemtype add **another 3 cols**: -

- o chennaTrainingName1
- o chennaTrainingDate1
- o chennaTrainingAmount1



## ChennaTrainingProduct1

**Solution =**

**Step 1** = Create new itemtype called “**ChennaTrainingProduct1**” by extending existing type called **Product** – and 3 new Cols called -- chennaTrainingName1, chennaTrainingDate1 & chennaTrainingAmount1

```

chennarrstrainingcore-items.xml
130 131 <!-- Example 2 = Extends Existing Product -->
132 <itemtype code="ChennaTrainingProduct1" extends="Product" autocreate="true" generate="true"
133   jaloClass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct1">
134   <attributes>
135     <attribute qualifier="chennaTrainingName1" type="localized:java.lang.String">
136       <persistence type="property" />
137       <modifiers read="true" write="true" search="true" optional="true" />
138     </attribute>
139     <attribute autoCreate="true" qualifier="chennaTrainingDate1" type="java.util.Date">
140       <persistence type="property" />
141     </attribute>
142     <attribute autoCreate="true" qualifier="chennaTrainingAmount1" type="java.lang.String">
143       <persistence type="property" />
144       <modifiers read="true" write="true" search="true" optional="true" />
145     </attribute>
146   </attributes>
147 </itemtype>
148

```

**Step 2** = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

**Step 3** = E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

**Step 4** = Perform “Platform – Update” [→ hAC → Platform → Update]

## Step 5 = Results

The screenshot shows the SAP ERP interface with the following details:

- Left Sidebar:** Contains navigation links for System, Catalog, Catalogs, Catalog Versions, Categories, Products (highlighted with a green oval), Product Variant Types, and Units. A search bar at the bottom says "Type here to search".
- Top Bar:** Shows a "Create New Product" dialog with fields for Article Number (RRRS-P), Approval (check), and Catalog version (Apparel Product Catalog : Staged). A green arrow points to the "Done" button.
- Bottom Main Area:** Shows a list of products under "[RRRS-P] - Apparel Product Catalog : Staged". One row is highlighted with a red background and contains the text "o chennaTrainingName1", "o chennaTrainingDate1", and "o chennaTrainingAmount1 === These 3 are not there". Below this is a table with columns: Article Number, Identifier, Catalog version, and Approval. The first row has values RRRS-P, empty, Apparel Product Catalog : Staged, and check.
- Buttons:** REFRESH, SAVE, and ADMINISTRATION (circled in blue).

The screenshot shows the SAP ERP interface with the following details:

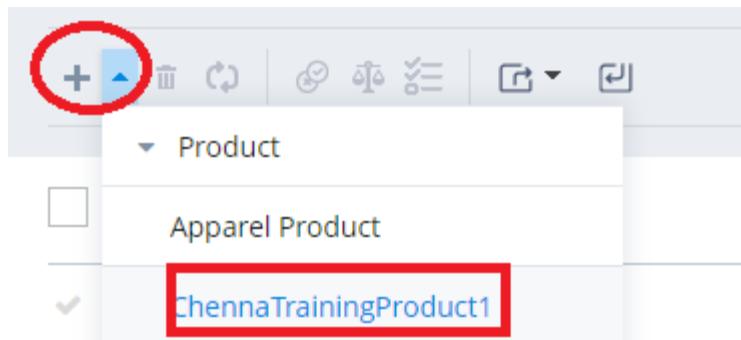
- Left Sidebar:** Contains navigation links for System, Catalog, Catalogs, Catalog Versions, Categories, Products (highlighted with a green oval), Product Variant Types, and Units. A search bar at the bottom says "Type here to search".
- Top Bar:** Shows a "Create New ChennaTrainingProduct1" dialog with fields for Article Number (RRRS-P1), Approval (approved), and Catalog version (Apparel Product Catalog : Staged). A green arrow points to the "Done" button.
- Bottom Main Area:** Shows a list of products under "[RRRS-P1] - Apparel Product Catalog : Staged". One row is highlighted with a red background and contains the text "ChennaTrainingProduct1.chennaTrainingName1", "ChennaTrainingProduct1.chennaTrainingDate1", and "ChennaTrainingProduct1.chennaTrainingAmount1". Below this is a table with columns: Article Number, Identifier, Catalog version, and Approval. The first row has values RRRS-P1, empty, Apparel Product Catalog : Staged, and approved.
- Buttons:** REFRESH, SAVE, and ADMINISTRATION (circled in blue).
- Table Rows:** The table rows for the highlighted product show "Buyer IDs" and "No value" under the Identifier column, and "ChennaTrainingProduct1.chennaTrainingName1", "ChennaTrainingProduct1.chennaTrainingDate1", and "ChennaTrainingProduct1.chennaTrainingAmount1" under the Catalog version column.

**Challenges in above Scenario =**

**1 = Column Label Names are not proper**

ChennaTrainingProduct1.chenna TrainingName1	ChennaTrainingProduct1.chennaTrainingAmount1	ChennaTrainingProduct1.chennaTrainingDate1

**2 = Itemtype Lable names are not proper**



**Scenario = How to give the proper display label names [Column names / Itemtype names]?**

**Requirement =**

**Columns =**

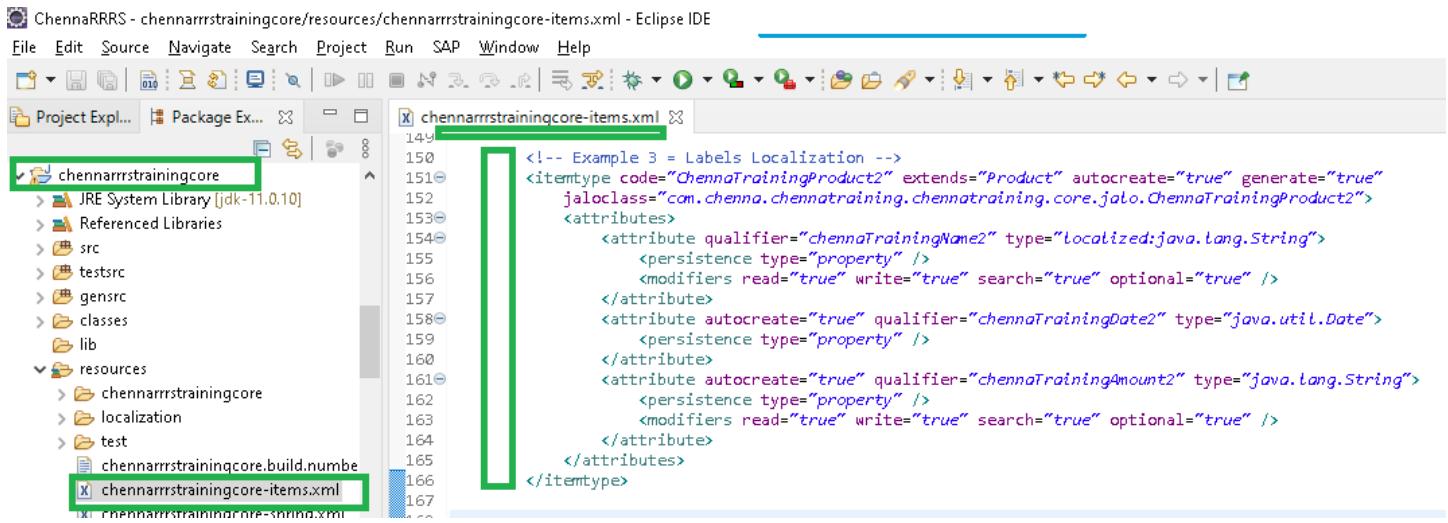
- o chennaTrainingName2 = Chenna Training Name 2
- o chennaTrainingDate2 = Chenna Training Date 2
- o chennaTrainingAmount2 = Chenna Training Amount 2

**Itemtype**

ChennaTrainingProduct2 = Chenna Training Product 2

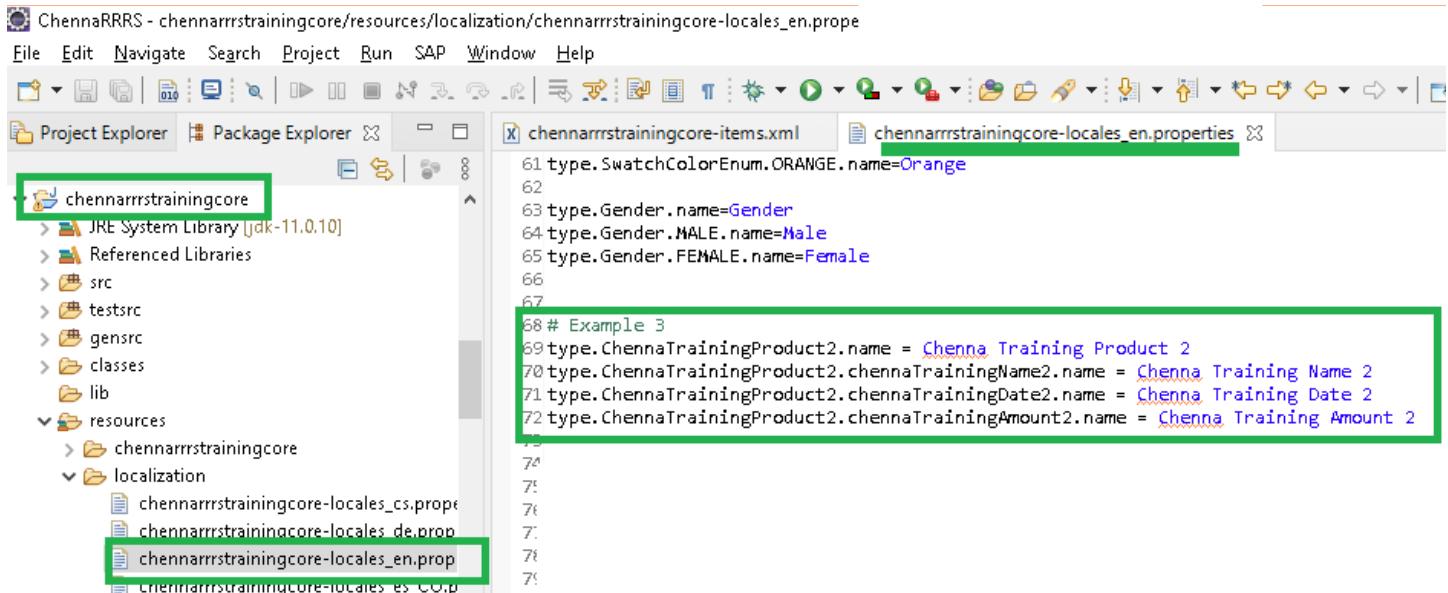
## Solution =

**Step 1** = Create new itemtype called “ChennaTrainingProduct2” by extending existing type called Product – and 3 new Cols.



```
<!-- Example 3 = Labels Localization -->
<itemtype code="ChennaTrainingProduct2" extends="Product" autoreate="true" generate="true"
  jaloClass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct2">
  <attributes>
    <attribute qualifier="chennaTrainingName2" type="localized:java.lang.String">
      <persistence type="property" />
      <modifiers read="true" write="true" search="true" optional="true" />
    </attribute>
    <attribute autoreate="true" qualifier="chennaTrainingDate2" type="java.util.Date">
      <persistence type="property" />
      <modifiers read="true" write="true" search="true" optional="true" />
    </attribute>
    <attribute autoreate="true" qualifier="chennaTrainingAmount2" type="java.lang.String">
      <persistence type="property" />
      <modifiers read="true" write="true" search="true" optional="true" />
    </attribute>
  </attributes>
</itemtype>
```

**Step 2** = Give proper displayable names



```
61 type.SwatchColorEnum.ORANGE.name=Orange
62
63 type.Gender.name=Gender
64 type.Gender.MALE.name=Male
65 type.Gender.FEMALE.name=Female
66
67
68 # Example 3
69 type.ChennaTrainingProduct2.name = Chenna Training Product 2
70 type.ChennaTrainingProduct2.chennaTrainingName2.name = Chenna Training Name 2
71 type.ChennaTrainingProduct2.chennaTrainingDate2.name = Chenna Training Date 2
72 type.ChennaTrainingProduct2.chennaTrainingAmount2.name = Chenna Training Amount 2
```

**Step 3** = Do the build = E:\rrrsssoftware\hybris\bin\platform>**ant clean all**

**Step 4** = E:\rrrsssoftware\hybris\bin\platform>**hybrisserver.bat**

**Step 5** = Perform “Platform – Update” [→ hAC → Platform → Update]

## Step 6 = Results

The screenshot shows the Oracle Commerce interface. On the left, a sidebar has 'Catalog' and 'Products' highlighted with green circles. The main area shows a product list with 'ChennaTrainingProduct1' and 'Chenna Training Product 2' selected. A modal window titled 'Create New Chenna Training Product 2' displays fields for Article Number (RRRS-P2), Approval (approved), and Catalog version (Apparel Product Catalog : Staged). A green arrow points to a 'Done' button. Below this, a product detail view for 'Chenna Training Product 2' is shown under the heading '[RRRS-P2] - Apparel Product Catalog : Staged'. The 'ADMINISTRATION' tab is selected. The administration panel contains fields for 'Chenna Training Name 2' (with a dropdown showing 'Buyer IDs' and 'No value'), 'Chenna Training Amount 2', and 'Chenna Training Date 2'. A green box highlights the administration section.

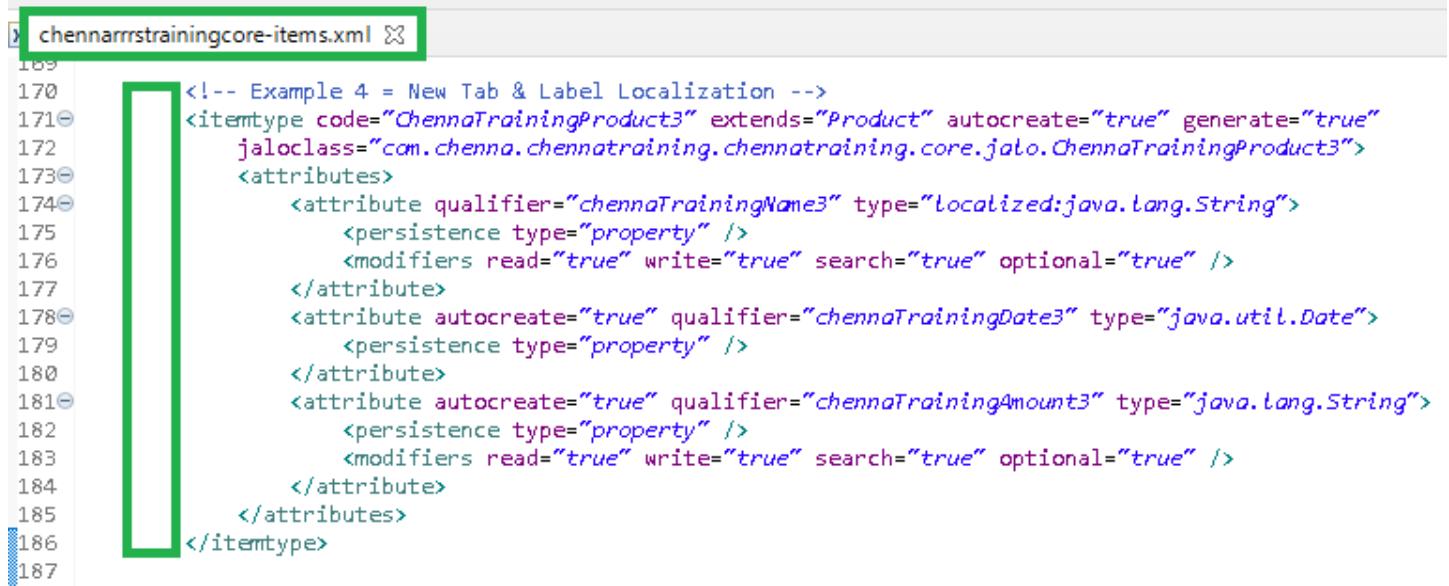
### Challenges in above Scenario =

Whenever we create **new attributes** [Cols] those are by default available inside “**Administration**” tab. How to bring these **3 new attributes** on our own tab (**Chenna RRRS Tab**).

**Requirement** = You want something like below =

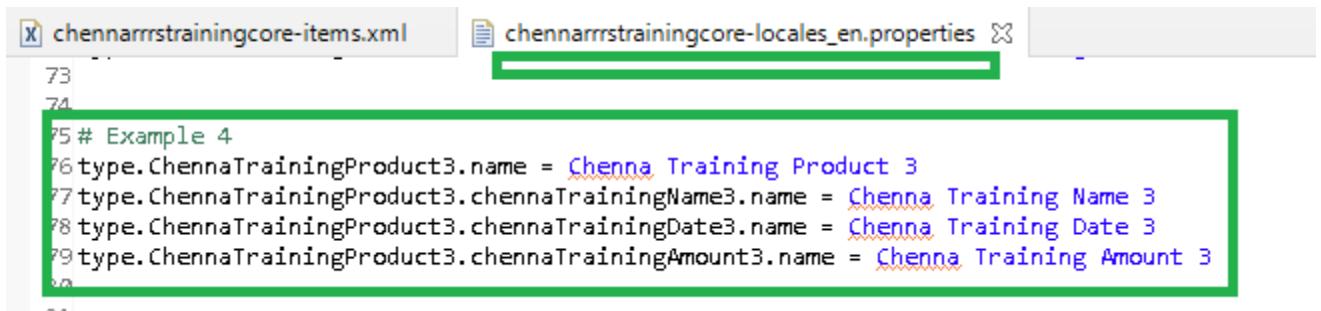
The diagram illustrates the requirement for a custom tab. It shows a standard product administration interface with tabs for PROPERTIES, ATTRIBUTES, BUNDLING, CATEGORY SYSTEM, PRICES, and VARIANTS. A green box highlights the 'CHENNA RRRS TAB' tab, which is currently inactive. Two green arrows point from the 'CHENNA RRRS TAB' tab towards the 'ATTRIBUTES' tab and the product detail area below. The product detail area shows fields for 'Chenna Training Name 3', 'Chenna Training Date 3' (with a calendar icon), and 'Chenna Training Amount 3'. A note says 'No Data for current languages'.

**Step 1 = Create new itemtype called “ChennaTrainingProduct3” by extending existing type called Product – and 3 new Cols.**



```
189
170    <!-- Example 4 = New Tab & Label Localization -->
171    <itemtype code="ChennaTrainingProduct3" extends="Product" autocreate="true" generate="true"
172        jaloClass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct3">
173        <attributes>
174            <attribute qualifier="chennaTrainingName3" type="localized:java.lang.String">
175                <persistence type="property" />
176                <modifiers read="true" write="true" search="true" optional="true" />
177            </attribute>
178            <attribute autocreate="true" qualifier="chennaTrainingDate3" type="java.util.Date">
179                <persistence type="property" />
180            </attribute>
181            <attribute autocreate="true" qualifier="chennaTrainingAmount3" type="java.lang.String">
182                <persistence type="property" />
183                <modifiers read="true" write="true" search="true" optional="true" />
184            </attribute>
185        </attributes>
186    </itemtype>
187
```

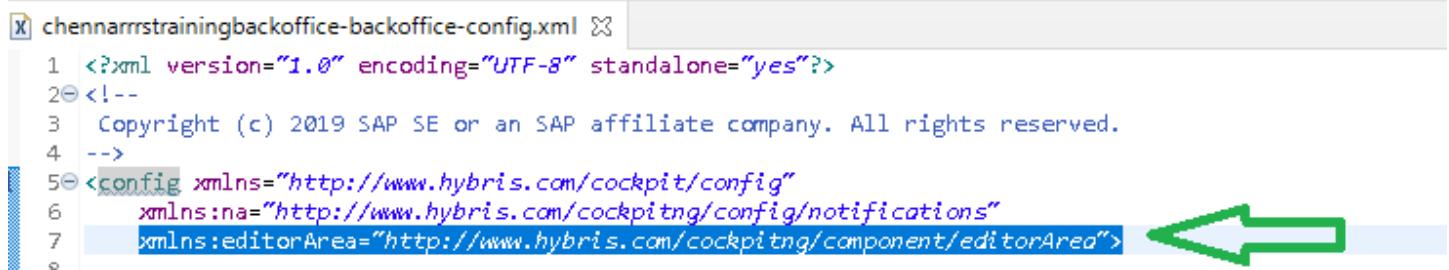
**Step 2 = Give proper displayable names**



chennarrstrainingcore-items.xml	chennarrstrainingcore-locales_en.properties
---------------------------------	---

```
73
74
75 # Example 4
76 type.ChennaTrainingProduct3.name = Chenna Training Product 3
77 type.ChennaTrainingProduct3.chennaTrainingName3.name = Chenna Training Name 3
78 type.ChennaTrainingProduct3.chennaTrainingDate3.name = Chenna Training Date 3
79 type.ChennaTrainingProduct3.chennaTrainingAmount3.name = Chenna Training Amount 3
80
```

**Step 3 = Bring newly created Cols into our own tab [Chenna RRRS Tab]**



```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <!--
3 Copyright (c) 2019 SAP SE or an SAP affiliate company. All rights reserved.
4 -->
5 <config xmlns="http://www.hybris.com/cockpit/config"
6     xmlns:na="http://www.hybris.com/cockpitng/config/notifications"
7     xmlns:editorArea="http://www.hybris.com/cockpitng/component/editorArea">
```

```

30 <!-- Example 4 -->
31 <context merge-by="type" parent="Product" type="ChennaTrainingproducts3" component="editor-area">
32   <editorArea:editorArea name="">
33     <editorArea:tab name="hmc.tab.product.cms" position="14">
34       <editorArea:section name="hmc.section.product.metadata">
35         <editorArea:attribute qualifier="productpageTitle" />
36         <editorArea:attribute qualifier="keywords" />
37       </editorArea:section>
38       <editorArea:section name="hmc.section.product.contentSlots">
39         <editorArea:attribute
40           editor="com.hybris.cockpitng.editor.localized(com.hybris.cockpitng.editor.wysiwyg)"
41           qualifier="chennaTrainingName3" />
42         <editorArea:attribute qualifier="chennaTrainingDate3" />
43         <editorArea:attribute
44           editor="com.hybris.cockpitng.editor.localized(com.hybris.cockpitng.editor.wysiwyg)"
45           qualifier="chennaTrainingAmount3" />
46       </editorArea:section>
47     </editorArea:tab>
48   </editorArea:editorArea>
49 </context>
50
51
52

```

## Step 4 = Give Proper Display Names for Backoffice UI Elements

Example = Tab name, Section name and ...

```

1# -----
2# Copyright (c) 2019 SAP SE or an SAP affiliate company. All rights reserved.
3# -----
4# in this file you has possibility to override widget labels
5# syntax:
6# <widgetID>.<labelName>=value
7#
8# Example:
9# yourCustomLabel=value
10 hmc.action.confirmpickup.success = Consignment Pickup Event Triggered Successfully
11
12# Example 4
13 hmc.tab.product.cms = Chenna RRRS Tab
14 hmc.section.product.metadata = Chenna RRRS Meta Data
15 hmc.section.product.contentSlots = Chenna RRRS Training Product Cols
16

```

**Step 5 = Do the build = E:\rrrssoftware\hybris\bin\platform>ant clean all**

**Step 6 = E:\rrrssoftware\hybris\bin\platform>hybrisserver.bat**

**Step 7 = Perfrom “Platform – Update” [→ hAC → Platform → Update]**

## Step 8 = Results

The screenshot shows the SAP Fiori interface for creating a new product. On the left, the navigation bar has 'Catalog' and 'Products' highlighted with green circles. The main area shows a list of products under 'Product' category, with 'Chenna Training Product 3' selected. A modal window titled 'Create New Chenna Training Product 3' is open, showing fields for Article Number (RRRS-P3), Approval (check), Catalog version (highlighted with a green arrow pointing to a yellow 'DONE' button), and a note about the Apparel Product Catalog being Staged.

[RRRS-P3] - Apparel Product Catalog : Staged

REFRESH SAVE

PROPERTIES ATTRIBUTES BUNDLING CATEGORY SYSTEM PRICES **CHENNA RRRS TAB** MULTIMEDIA VARIANTS EXTENDED ATTRIBUTES

[HMC.SECTION.PRODUCT.METADATA]

Keywords

CHENNA RRRS TRAINING PRODUCT COLS

Chenna Training Name 3 Chenna Training Date 3 Chenna Training Amount 3

No Data for current languages

sdf

### Challenges in above Scenario =

To create product – How many clicks required?

-- Catalog – Products – [+] -- → Product --- Chenna RRRS Product 3

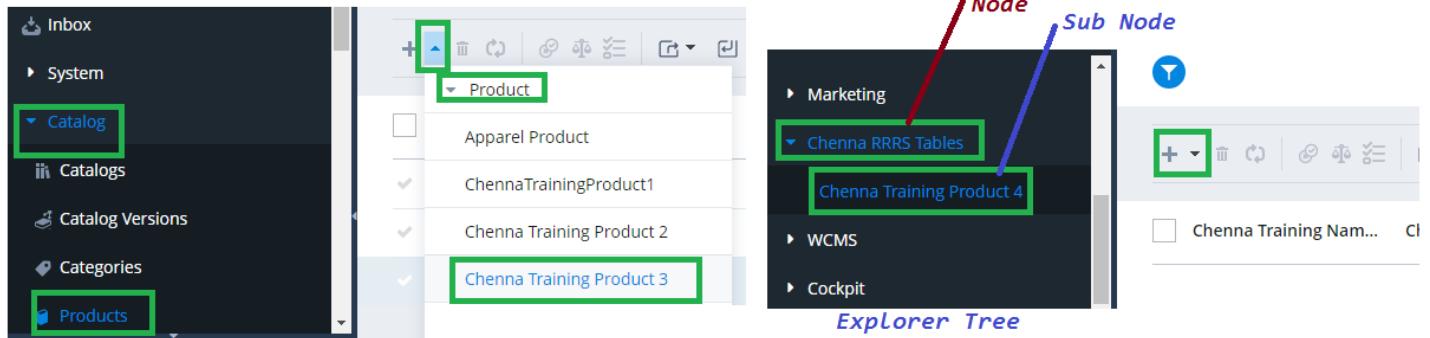
==== Total Clicks = 5

==== How to bring this into Left Navigation section to **reduces No of clicks**.

---

Contact Us = **ChennaReddyTraining@RRRS.CO.IN**

---



*This is Results Customer want.*

**Requirement** = You want something like below =

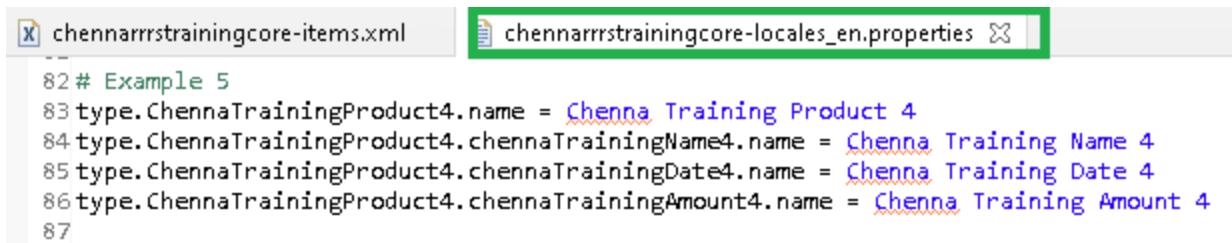
**Step 1** = Create new itemtype called “**ChennaTrainingProduct4**” by extending existing type called **Product** – and 3 new Cols.

```

<!-- Example 4 = New Tab & Label Localization -->
<itemtype code="ChennaTrainingProduct4" extends="Product" autocreate="true" generate="true"
  jaloClass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct4">
  <attributes>
    <attribute qualifier="chennaTrainingName4" type="localized:java.lang.String">
      <persistence type="property" />
      <modifiers read="true" write="true" search="true" optional="true" />
    </attribute>
    <attribute autocreate="true" qualifier="chennaTrainingDate4" type="java.util.Date">
      <persistence type="property" />
    </attribute>
    <attribute autocreate="true" qualifier="chennaTrainingAmount4" type="java.lang.String">
      <persistence type="property" />
      <modifiers read="true" write="true" search="true" optional="true" />
    </attribute>
  </attributes>
</itemtype>

```

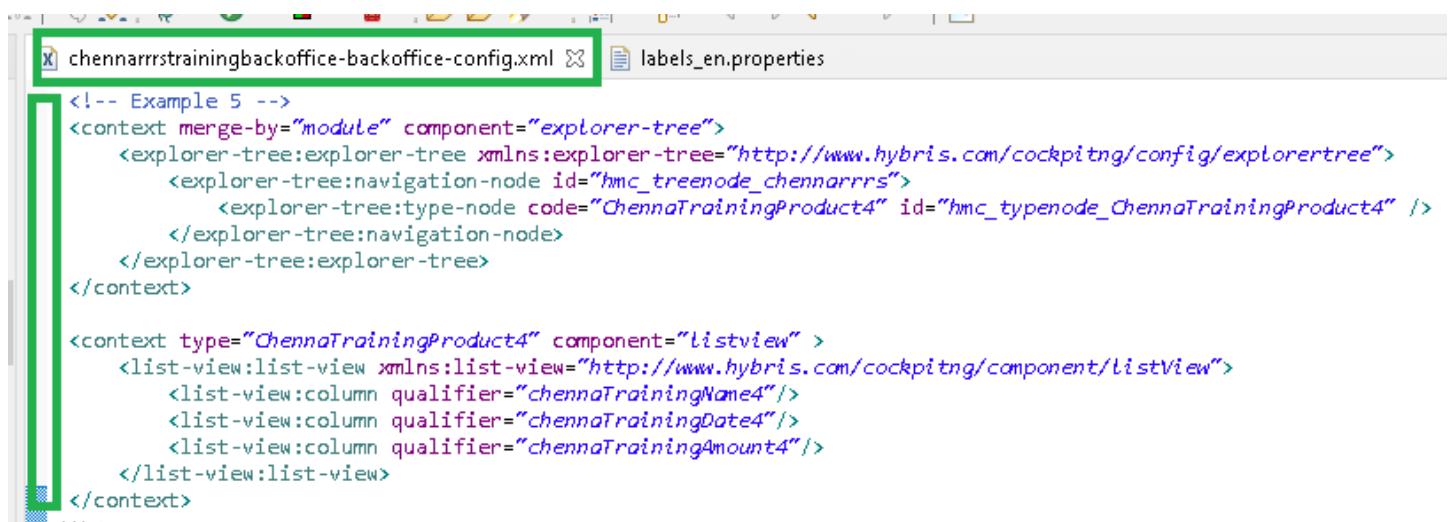
## Step 2 = Give proper displayable names



```
chennarrstrainingcore-items.xml chennarrstrainingcore-locales_en.properties
82 # Example 5
83 type.ChennaTrainingProduct4.name = Chenna Training Product 4
84 type.ChennaTrainingProduct4.chennaTrainingName4.name = Chenna Training Name 4
85 type.ChennaTrainingProduct4.chennaTrainingDate4.name = Chenna Training Date 4
86 type.ChennaTrainingProduct4.chennaTrainingAmount4.name = Chenna Training Amount 4
87
```

## Step 3 = Display ChennaTrainingProduct4 itemtype in explorer tree.

For this do the changes \*-backoffice-config.xml file.

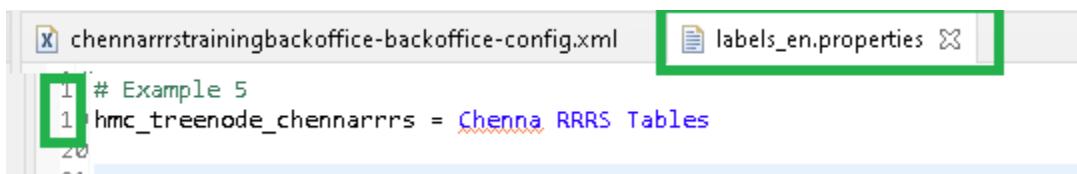


```
chennarrstrainingbackoffice-backoffice-config.xml labels_en.properties
<!-- Example 5 -->
<context merge-by="module" component="explorer-tree">
    <explorer-tree:explorer-tree xmlns:explorer-tree="http://www.hybris.com/cockpitng/config/explorertree">
        <explorer-tree:navigation-node id="hmc_treenode_chennarrrs">
            <explorer-tree:type-node code="ChennaTrainingProduct4" id="hmc_typenode_ChennaTrainingProduct4" />
        </explorer-tree:navigation-node>
    </explorer-tree:explorer-tree>
</context>

<context type="ChennaTrainingProduct4" component="listview">
    <list-view:list-view xmlns:list-view="http://www.hybris.com/cockpitng/component/listView">
        <list-view:column qualifier="chennaTrainingName4"/>
        <list-view:column qualifier="chennaTrainingDate4"/>
        <list-view:column qualifier="chennaTrainingAmount4"/>
    </list-view:list-view>
</context>
```

## Step 4 = Give Proper Display Names for Backoffice UI Elements

Example = Node name and ...



```
chennarrstrainingbackoffice-backoffice-config.xml labels_en.properties
1 # Example 5
1 hmc_treenode_chennarrrs = Chenna RRRS Tables
20
21
```

## Step 5 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

Step 6 = E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

Step 7 = Perform “Platform – Update” [→ hAC → Platform → Update]

## Step 8 = Results

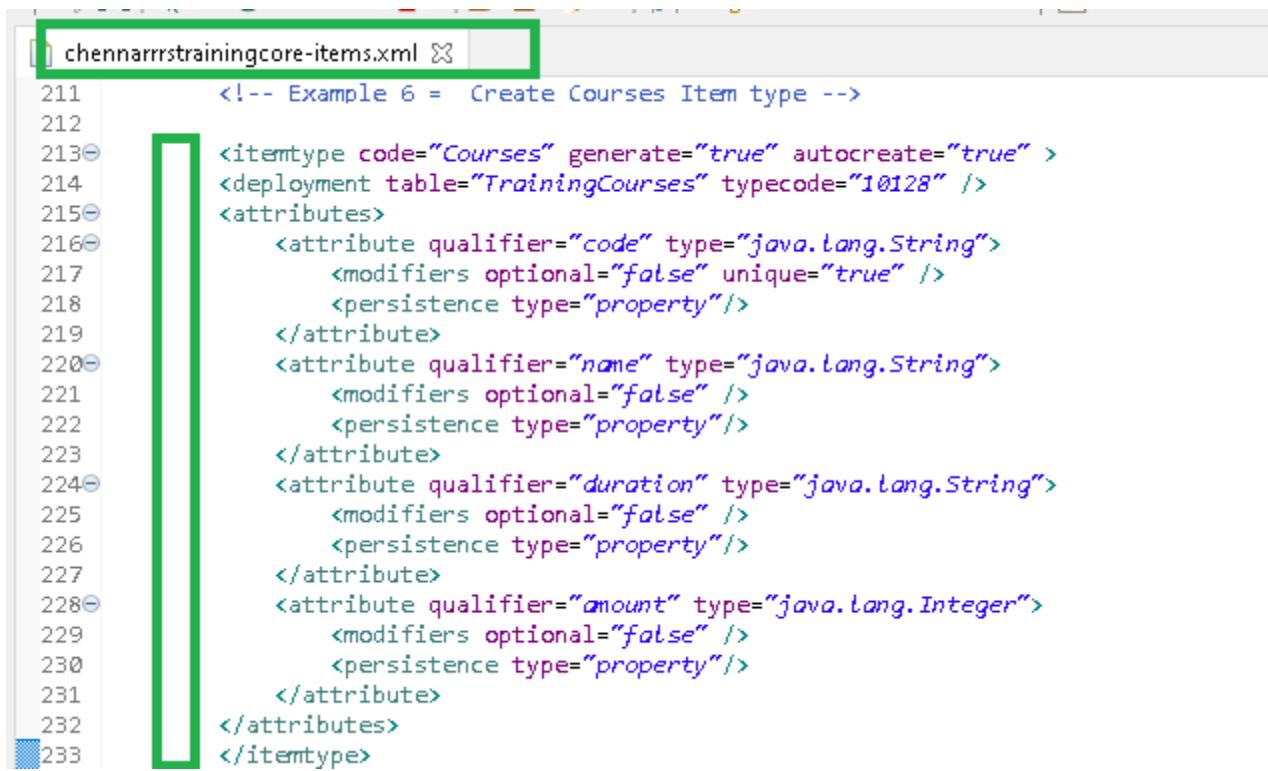
The screenshot shows the SAP Administration Cockpit interface. On the left, the navigation tree includes 'Chenna RRRS Training Courses', 'Chenna RRRS Tables' (which is expanded to show 'Chenna Training Product 4'), 'Personalization', 'Subscription', 'Entitlements', and 'Order Management'. The main area displays a table with columns: 'Chenna Training Name...', 'Chenna Training Date...', and 'Chenna Training Amount 4'. A green box highlights the 'Chenna Training Product 4' entry in the tree. A green plus sign button is visible in the toolbar above the table.

Scenario = Create itemtype called **Courses** with 4 Cols [Code, Name, Duration & Amount].

Display Courses in Backoffice Left Navigation [Explorer Tree].

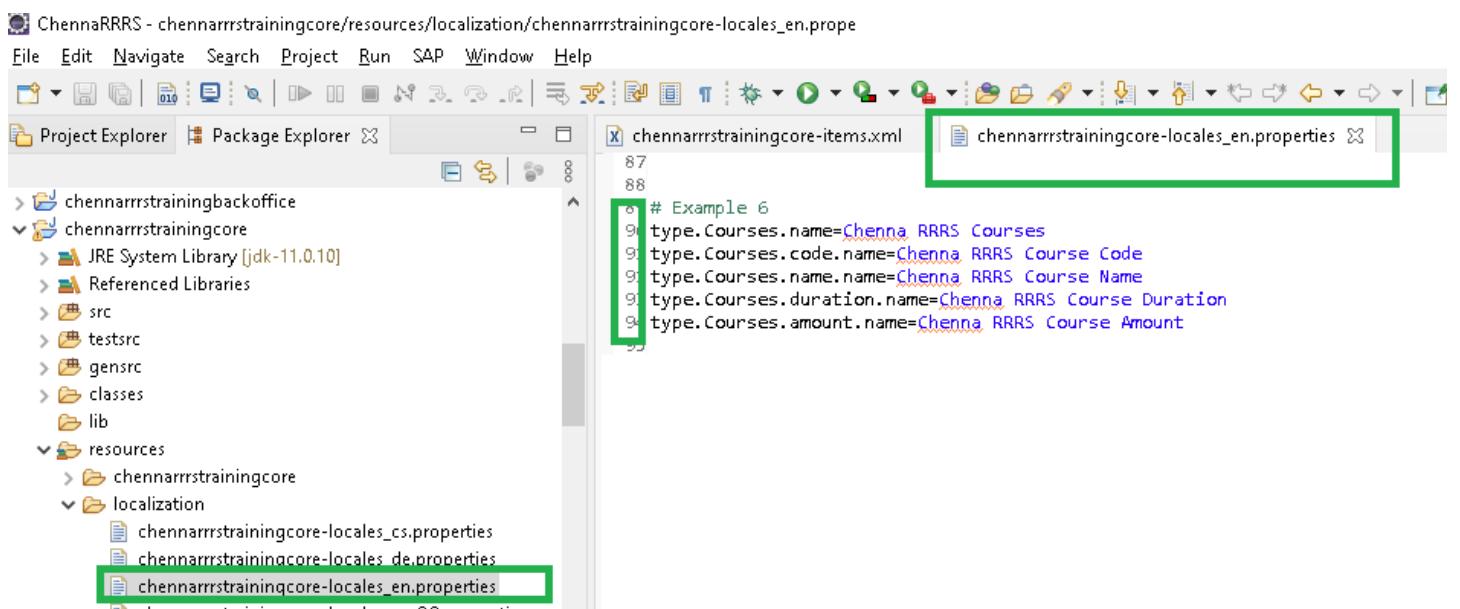
The screenshot shows the SAP Administration Cockpit interface. The navigation tree on the left includes 'B2B Commerce', 'Chenna RRRS Training Courses' (which is expanded to show 'Chenna RRRS Courses'), 'Chenna RRRS Tables', and 'Personalization'. The main area displays a table with columns: 'Chenna RRRS Course Co...' and 'Chenna RRRS Course Name'. A green box highlights the 'Chenna RRRS Courses' entry in the tree. The toolbar above the table includes a green plus sign button.

**Step 1 = Create new itemtype called “Courses” with 4 Cols [Code, Name, Duration & Amount].**



```
1 chennarrstrainingcore-items.xml
211     <!-- Example 6 = Create Courses Item type -->
212
213     <itemtype code="Courses" generate="true" autocreate="true" >
214         <deployment table="TrainingCourses" typecode="10128" />
215         <attributes>
216             <attribute qualifier="code" type="java.lang.String">
217                 <modifiers optional="false" unique="true" />
218                 <persistence type="property"/>
219             </attribute>
220             <attribute qualifier="name" type="java.lang.String">
221                 <modifiers optional="false" />
222                 <persistence type="property"/>
223             </attribute>
224             <attribute qualifier="duration" type="java.lang.String">
225                 <modifiers optional="false" />
226                 <persistence type="property"/>
227             </attribute>
228             <attribute qualifier="amount" type="java.lang.Integer">
229                 <modifiers optional="false" />
230                 <persistence type="property"/>
231             </attribute>
232         </attributes>
233     </itemtype>
```

**Step 2 = Give proper displayable names**



ChennaRRRS - chennarrstrainingcore/resources/localization/chennarrstrainingcore-locales\_en.propre

File Edit Navigate Search Project Run SAP Window Help

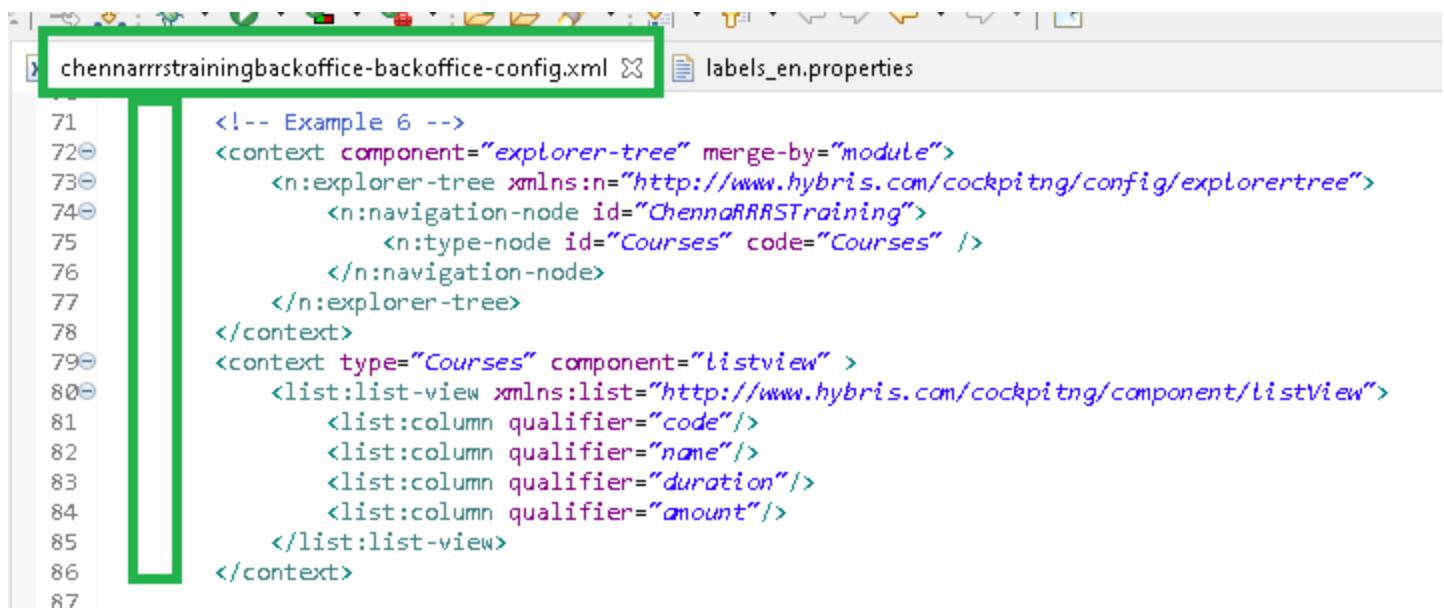
Project Explorer Package Explorer chennarrstrainingcore-items.xml chennarrstrainingcore-locales\_en.properties

```
87
88
89 # Example 6
90 type.Courses.name=Chenna RRRS Courses
91 type.Courses.code.name=Chenna RRRS Course Code
92 type.Courses.name.name=Chenna RRRS Course Name
93 type.Courses.duration.name=Chenna RRRS Course Duration
94 type.Courses.amount.name=Chenna RRRS Course Amount
```

chennarrstrainingbackoffice  
chennarrstrainingcore  
  JRE System Library [jdk-11.0.10]  
  Referenced Libraries  
  src  
  testsrc  
  gensrc  
  classes  
  lib  
resources  
  chennarrstrainingcore  
    localization  
      chennarrstrainingcore-locales\_cs.properties  
      chennarrstrainingcore-locales\_de.properties  
      **chennarrstrainingcore-locales\_en.properties**

## Step 3 = Display Courses itemtype in explorer tree.

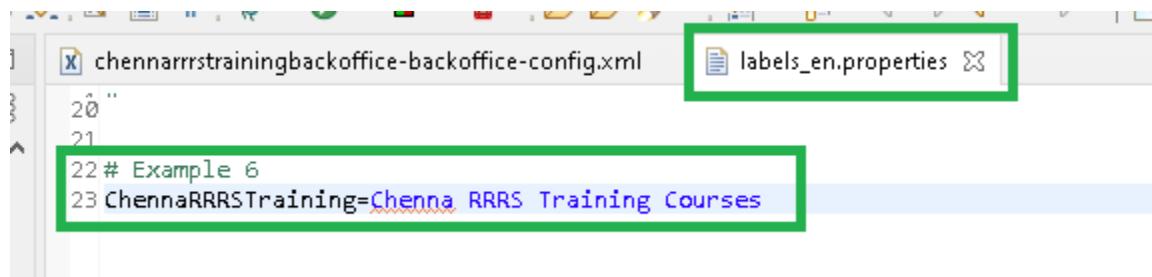
For this do the changes \*-backoffice-config.xml file.



```
71 <!-- Example 6 -->
72 <context component="explorer-tree" merge-by="module">
73   <n:explorer-tree xmlns:n="http://www.hybris.com/cockpitng/config/explorertree">
74     <n:navigation-node id="ChennaRRRSTraining">
75       <n:type-node id="Courses" code="Courses" />
76     </n:navigation-node>
77   </n:explorer-tree>
78 </context>
79 <context type="Courses" component="listview" >
80   <list:list-view xmlns:list="http://www.hybris.com/cockpitng/component/listView">
81     <list:column qualifier="code"/>
82     <list:column qualifier="name"/>
83     <list:column qualifier="duration"/>
84     <list:column qualifier="amount"/>
85   </list:list-view>
86 </context>
87
```

## Step 4 = Give Proper Display Names for Backoffice UI Elements

Example = Node name and ...



```
20 ""
21
22 # Example 6
23 ChennaRRRSTraining=Chenna RRRS Training Courses
```

## Step 5 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

Step 6 = E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

Step 7 = Perfrom “Platform – Update” [→ hAC → Platform → Update]

## Step 8 = Results

Not secure | localhost:9002/backoffice/

**Create New Chennai RRRS Courses**

**MANDATORY VALUES**

- Chenna RRRS Course Amount: 500
- Chenna RRRS Course Code: SAPCommTechoFun
- Chenna RRRS Course Duration: 120 Hrs
- Chenna RRRS Course Name: SAP Comm Tech Fun
- Time created: Jun 1, 2021 12:34:38 AM DONE

**Chenna RRRS Training Courses**

**Chenna RRRS Courses**

**Chenna RRRS Tables**

**Personalization**

**Subscription**

**Entitlements**

**Order Management**

**SAVED QUERIES**

No queries

**Chenna RRRS Course Co... Chenna RRRS Course Name Chenna RRRS Course Durati... Chenna RRRS Course Amount**

SAPCommTech	SAP Comm Tech Only	60 Hrs	500
SAPCommTechoFun	SAP Comm Techo Fun	120 Hrs	500

**Platform Monitoring Maintenance Console**

**Scripting Languages FlexibleSearch ImpEx Import ImpEx Export LDAP**

**Import content Import script**

**Import content**

```
insert_update Courses;code[unique=true];name;duration;amount
;CLang;"C Lang";60 Hrs;500
;SAPSpartacus;"SAP Sparatacus";40 Hrs;500
```

## Output =

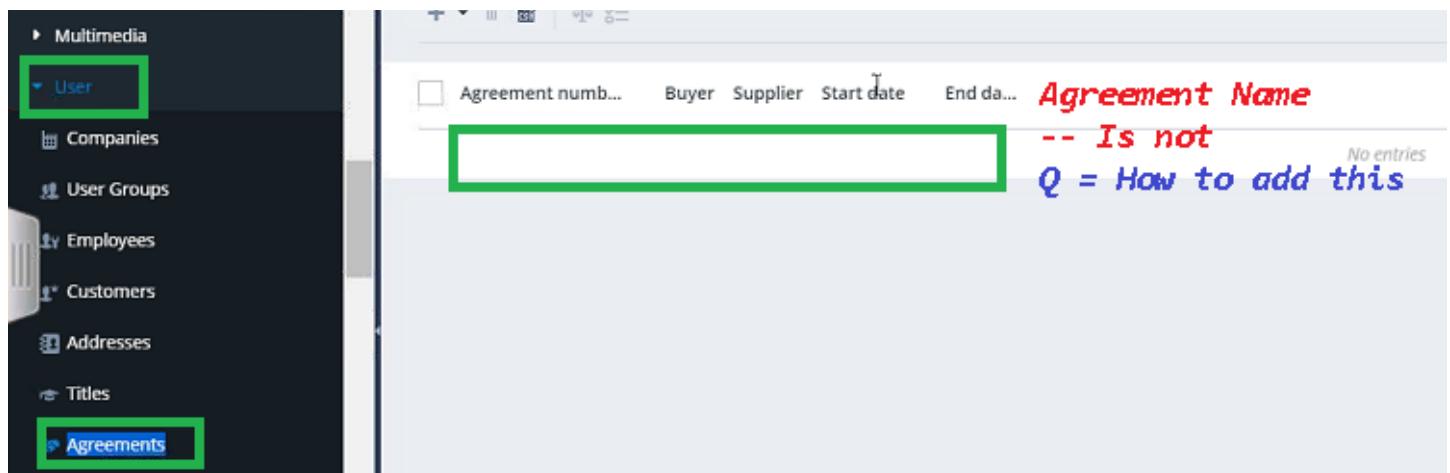
Chenna RRRS Course Co...	Chenna RRRS Course Name	Chenna RRRS Course Durati...	Chenna RRRS Course Amount
SAPSpartacus	SAP Sparatacus	40 Hrs	500
CLang	C Lang	60 Hrs	500
SAPCommTech	SAP Comm Tech Only	60 Hrs	500
SAPCommTechoFun	SAP Comm Techo Fun	120 Hrs	500

Contact Us = **ChennaReddyTraining@RRRS.CO.IN**

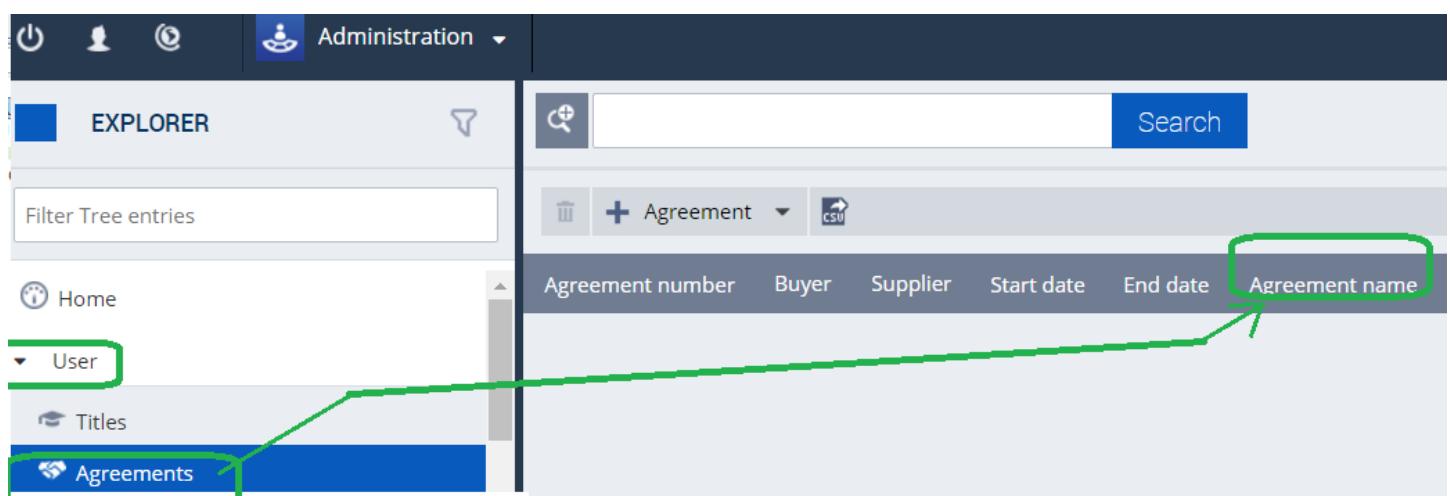
**Scenario** = How to add a **new field / new attribute** in the backoffice to the **existing node**? (or) How to customize the back office with new attribute in the existing node?

**Example** = Let's add “**Agreement Name**” in **User → Agreements**

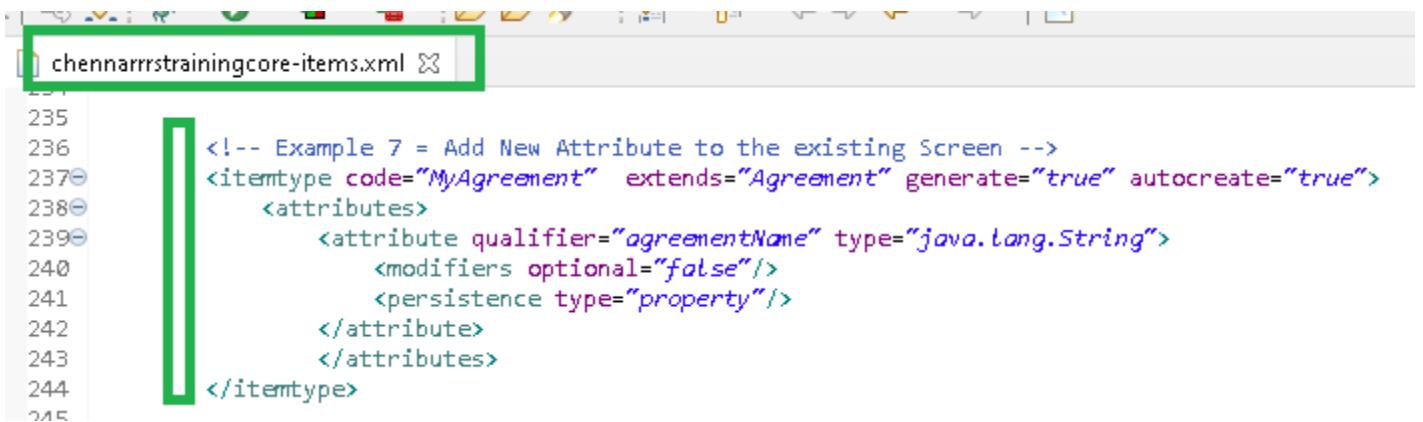
We already have “**Users & Agreements**”, but there is no “**Agreement Name**” column inside **Agreements** table. How to add it & display.



**Output =**

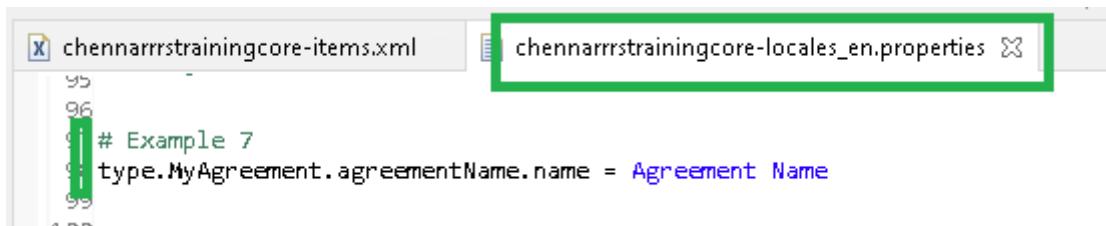


**Step 1 = Create new itemtype called “MyAgreement” by extending existing type called **Agreement** – and 1 new Cols [agreementName].**



```
235
236    <!-- Example 7 = Add New Attribute to the existing Screen -->
237    <itemtype code="MyAgreement" extends="Agreement" generate="true" autocreate="true">
238        <attributes>
239            <attribute qualifier="agreementName" type="java.lang.String">
240                <modifiers optional="false"/>
241                <persistence type="property"/>
242            </attribute>
243        </attributes>
244    </itemtype>
245
```

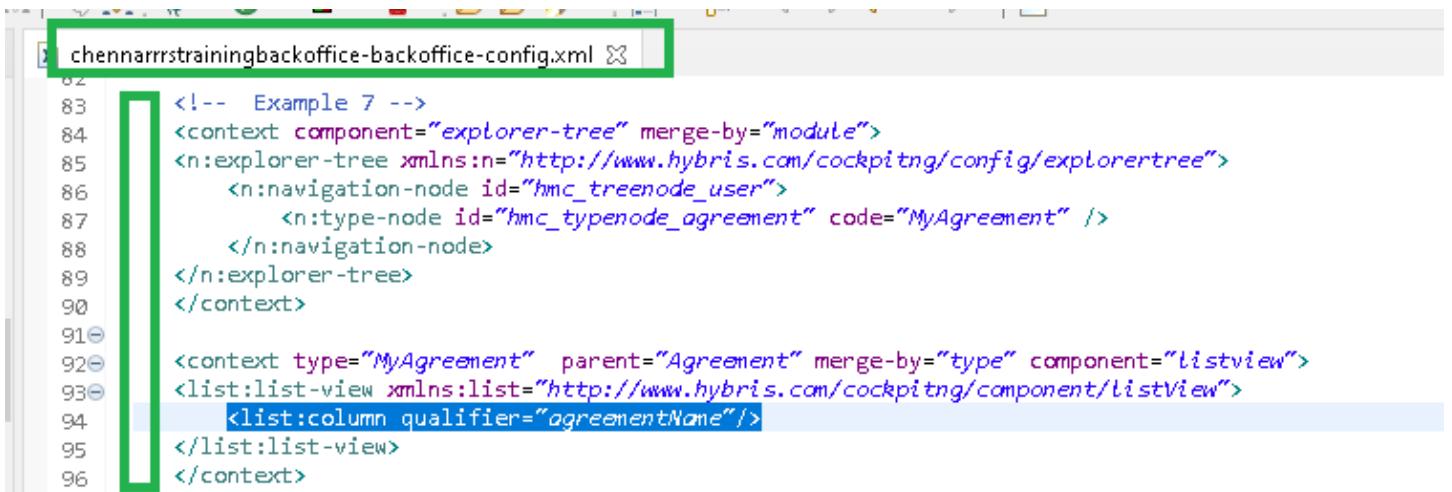
**Step 2 = Give proper displayable names**



chennarrstrainingcore-items.xml	chennarrstrainingcore-locales_en.properties
95	
96	
97 # Example 7	type.MyAgreement.agreementName.name = Agreement Name
98	
99	
100	

**Step 3 = Display **agreementName** inside Agreement Type Cols.**

For this do the changes **\*-backoffice-config.xml** file.



```
62
63    <!-- Example 7 -->
64    <context component="explorer-tree" merge-by="module">
65        <n:explorer-tree xmlns:n="http://www.hybris.com/cockpitng/config/explorertree">
66            <n:navigation-node id="hmc_treenode_user">
67                <n:type-node id="hmc_typenode_agreement" code="MyAgreement" />
68            </n:navigation-node>
69        </n:explorer-tree>
70    </context>
71
72    <context type="MyAgreement" parent="Agreement" merge-by="type" component="listview">
73        <list:list-view xmlns:list="http://www.hybris.com/cockpitng/component/listView">
74            <list:column qualifier="agreementName"/>
75        </list:list-view>
76    </context>
77
```

**Step 4 = Do the build = E:\rrrsssoftware\hybris\bin\platform>ant clean all**

**Step 5 = E:\rrrsssoftware\hybris\bin\platform>hybrisserver.bat**

**Step 6 = Perfrom “Platform – Update” [→ hAC → Platform → Update]**

**Results =**

The screenshot shows the HAC interface. On the left, a sidebar under the 'User' category lists various entities: Companies, User Groups, Employees, Customers, Addresses, Titles, and Agreements. The 'Agreements' item is highlighted with a green box. The main content area displays a table with the following columns: 'Agreement numb...', 'Buyer', 'Supplier', 'Start date', 'End da...', and 'Agreement Name'. A green box highlights the 'Agreement Name' column header. Below the table, a message says 'No entries'.

**Q = How to find the node Id's?**

The screenshot shows the HAC interface. On the left, a sidebar under the 'User' category lists various entities: Companies, User Groups, Employees, Customers, Addresses, Titles, Agreements, Consent, Consent Template, Order, and Orders. The 'Agreements' item is highlighted with a green box. The right panel displays the XML configuration file 'platformbackoffice-backoffice-config.xml'. The XML code includes several navigation nodes and type nodes, such as 'navigation-node id="hmc\_treenode\_agreement"' and 'type-node code="agreement" id="hmc\_typenode\_agreement"'. A green arrow points from the 'Agreements' item in the sidebar to the 'agreement' type node in the XML code.

## Scenario = hAC – Platform – Update Options

### Solution 1 =

The screenshot shows the hybris administration console interface. At the top, there is a header bar with the URL "Not secure | localhost:9002/platform/update". On the right side of the header, it says "You're Administrator" and has a "logout" button. Below the header is a blue navigation bar with the title "(v) hybris administration console". The navigation bar has tabs: "Platform" (which is highlighted in green), "Monitoring", "Maintenance", and "Console". Underneath this, there is a secondary navigation bar with tabs: "Tenants", "Configuration", "System", "Logging", "Extensions", "Initialization", "Update" (which is highlighted in green), and "SQL". In the main content area, there are three buttons: "Lock", "Update" (which is highlighted in green), and "Dump configuration".

### Solution 2 =

The screenshot shows the hybris administration console interface. At the top, there is a header bar with the URL "Not secure | localhost:9002/platform/update". On the right side of the header, it says "You're Administrator" and has a "logout" button. Below the header is a blue navigation bar with the title "(v) hybris administration console". The navigation bar has tabs: "Platform" (which is highlighted in green), "Monitoring", "Maintenance", and "Console". Underneath this, there is a secondary navigation bar with tabs: "Tenants", "Configuration", "System", "Logging", "Extensions", "Initialization", "Update" (which is highlighted in green), and "SQL". In the main content area, there are three buttons: "Lock", "Update" (which is highlighted in green), and "Dump configuration". To the right of these buttons, there is a modal dialog box with the title "Master" and the value "true". Inside the dialog, there are three checkboxes: "Update running system" (checked), "Create essential data" (unchecked), and "Localize types" (checked). To the right of the modal, there is a code editor window titled "essential-data\_en.impex" showing the following impex code:

```
11# Languages
12 UPDATE Language;isocode[unique=true];name[lang=$lang]
13 ;de;"German"
14 ;en;"English"
15 ;ja;"Japanese"
16 ;zh;"Chinese"
17
18# Currencies
19 UPDATE Currency;isocode[unique=true];name[lang=$lang]
20 ;EUR;"Euro"
21 ;GBP;"Pound"
22 ;JPY;"Japanese Yen"
23 ;USD;"US Dollar"
24
25# Titles
26 UPDATE Title;code[unique=true];name[lang=$lang]
27 ;dr;"Dr."
```

## Solution 3 =

Not secure | localhost:9002/platform/update

### (x) hybris administration console

You're Administrator  
logout

Platform Monitoring Maintenance Console

Tenants Configuration System Logging Extensions Initialization Update SQL S...

**Lock**

**Update**

**Dump configuration**

Database pool  
hybris

mrs

chennarrstrainingcore

**Import Users & Groups**

mrs

**Master**  
true

Update running system

Create essential data

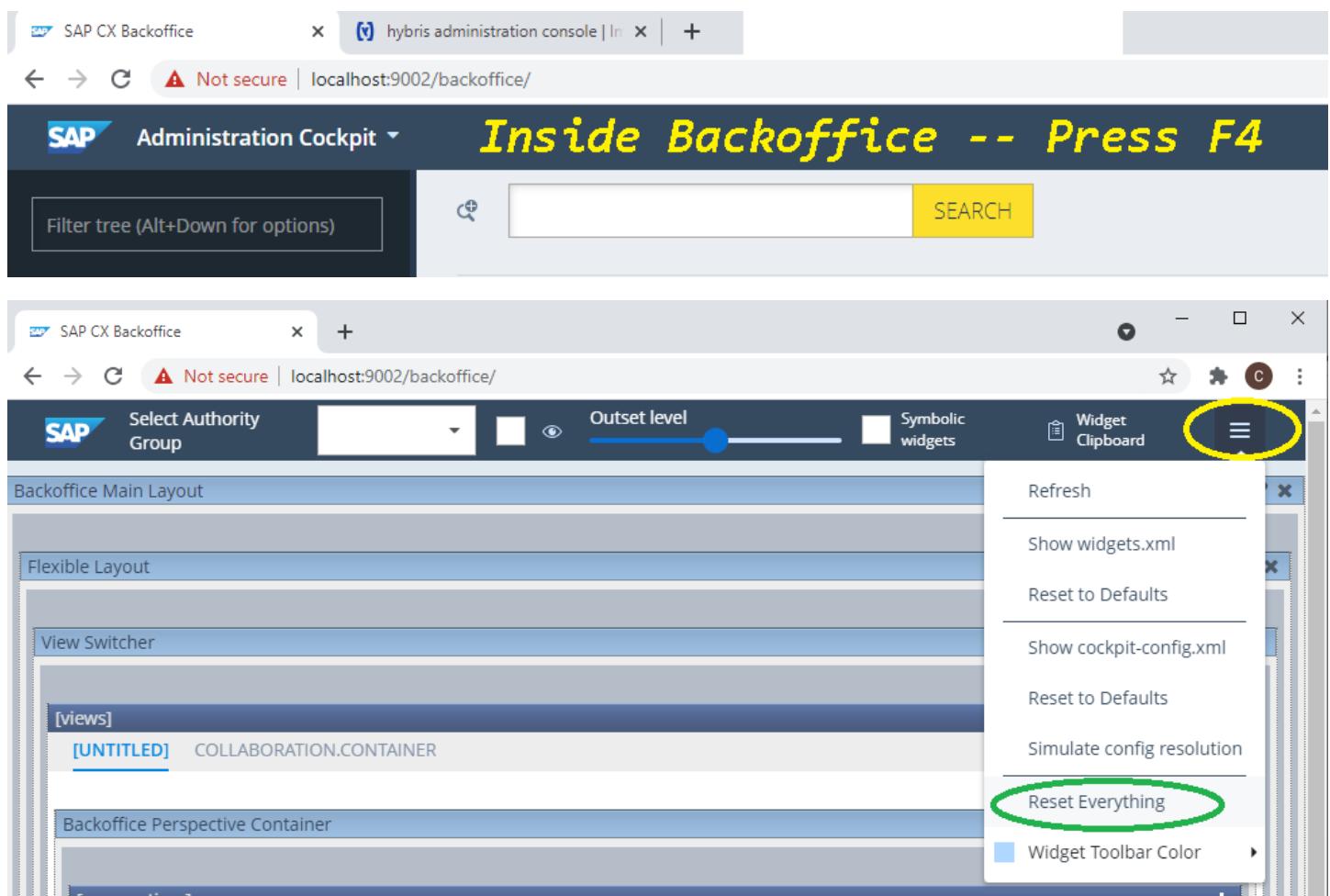
Localize types

**Note** = When we do “backoffice customization changes” – some times we don't see expected results even though the everything correct.

**Q = What to do?**

**Solution =**

**Inside backoffice – Press F4 [Orchestration Mode]**



Again, Press **F4** to come out from Orchestration Mode.

====

After this – Check the results. We should be fine.

## Backoffice –

Hybris 5.X = hMC is important.

Hybris 6.X / 1808 and above= **Backoffice** is important.

Backoffice is created with Widgets.

All the widget configuration are available in **\*wedge\*.xml** file.

You can find widgets information in below files: -

platformbackoffice-backoffice-config.xml

platformbackoffice-backoffice-widgets.xml

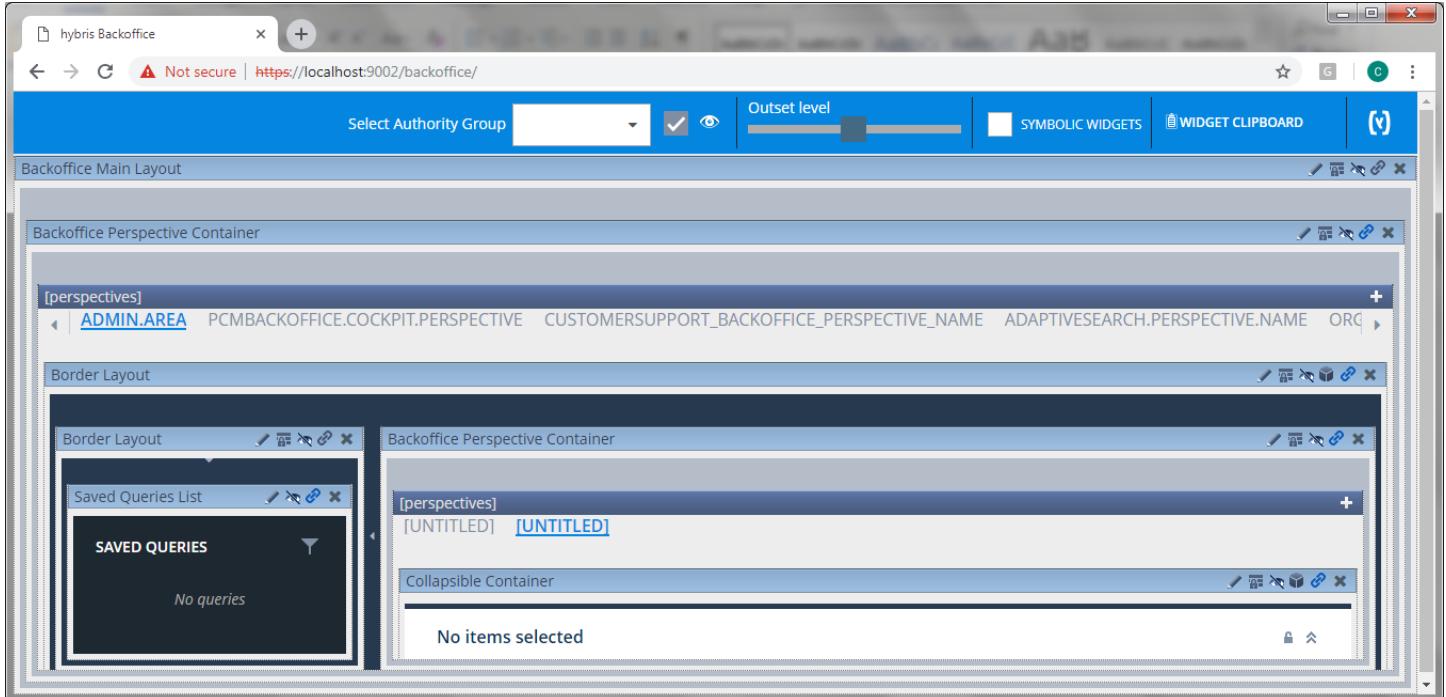
backoffice-widgets.xml

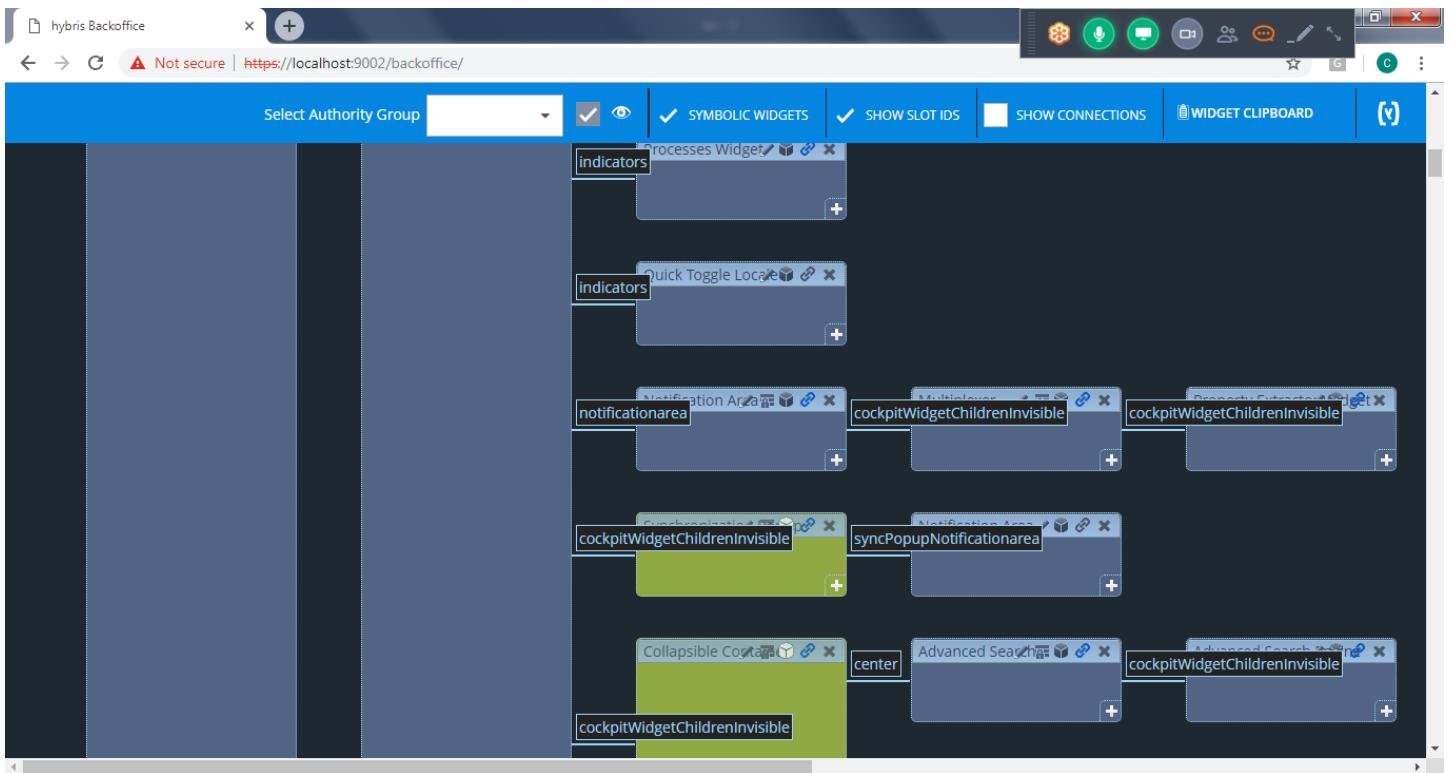
All these widgets results are combined & you have the backoffice.

1 widget can be connected with another widget.

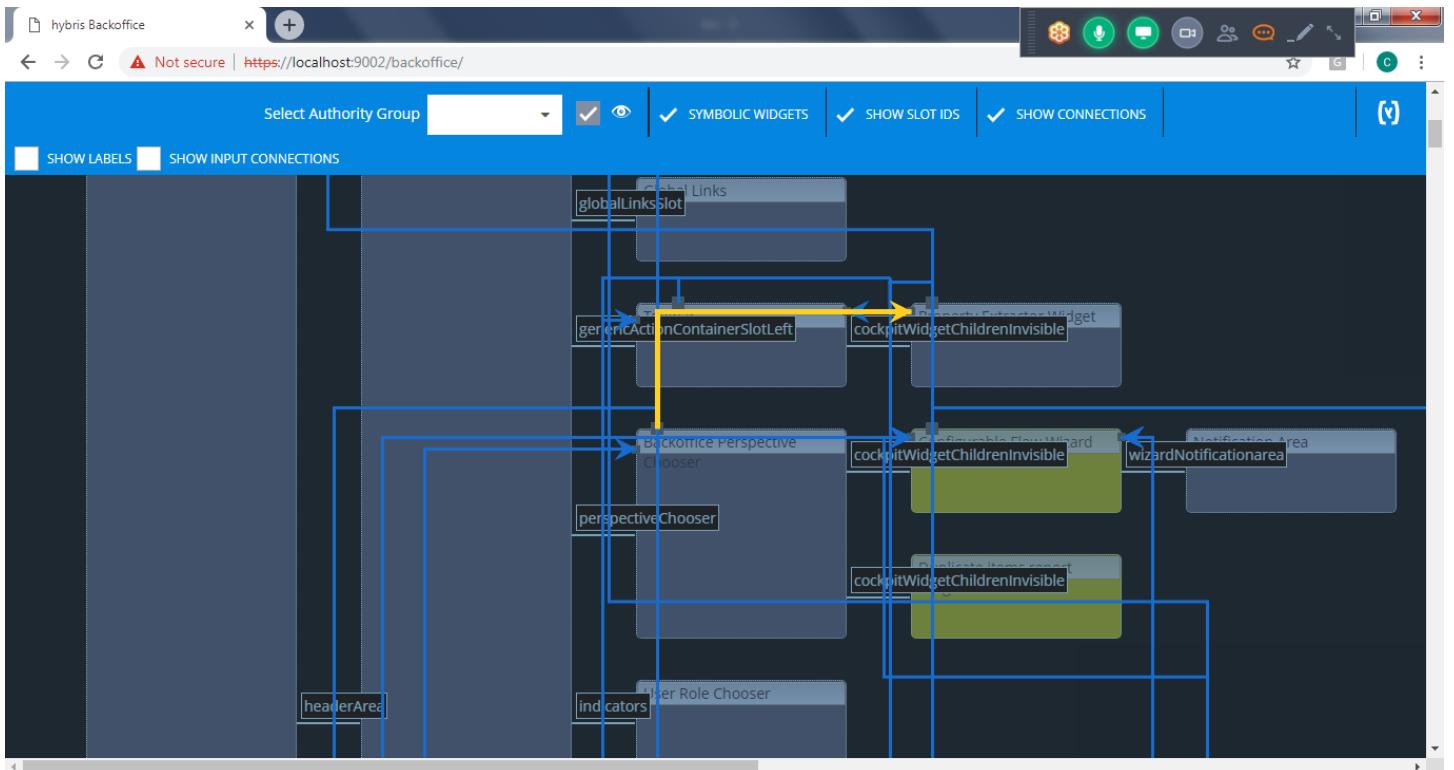
Orchestration mode – Will give full picture.

In Backoffice – Press F4 == It will take you to “Backoffice – Orchestration”





→ 1 widget is connected another widget.



hybris Backoffice

Select Authority Group

SYMBOLIC WIDGETS SHOW SLOT IDS SHOW CONNECTIONS

SHOW LABELS SHOW INPUT CONNECTIONS

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widgets>
    <widget id="mainSlot" widgetDefinitionId="com.hybris.cockpitng.backoffice.mainLayout" slotId="mainSlot" template="false">
        <widget id="notificationEA" widgetDefinitionId="com.hybris.cockpitng.backoffice.defaultEditorArea" slotId="cockpitWidget">
            <instance-settings socketEventRoutingMode="LAST_USED">
                <create onInit="false" reuseExisting="true">
                    <all-incoming-events/>
                </create>
                <close>
                    <select onInit="false">
                        <all-incoming-events/>
                    </select>
                </close>
                <setting key="height" type="String">80%</setting>
                <setting key="WidgetStyleClass" type="String"></setting>
                <setting key="WidgetStyleAttribute" type="String"></setting>
                <setting key="width" type="String">80%</setting>
                <virtual-sockets/>
            </instance-settings>
        <setting key="WidgetStyleAttribute" type="String"></setting>
        <setting key="width" type="String">80%</setting>
    </widget>
    <widget id="referenceadvancedsearchgroup" widgetDefinitionId="com.hybris.cockpitng.backoffice.referenceadvancedsearch">
        <instance-settings socketEventRoutingMode="LAST_USED">
            <create onInit="false" reuseExisting="false">
                <incoming-events>
                    <socket-event id="referenceSearchCtx"/>
                </incoming-events>
            </create>
            <close>
                <outgoing-events>
                    <socket-event id="selectedReferenceOutput"/>
                </outgoing-events>
            </close>
        </instance-settings>
    </widget>
</widgets>
```

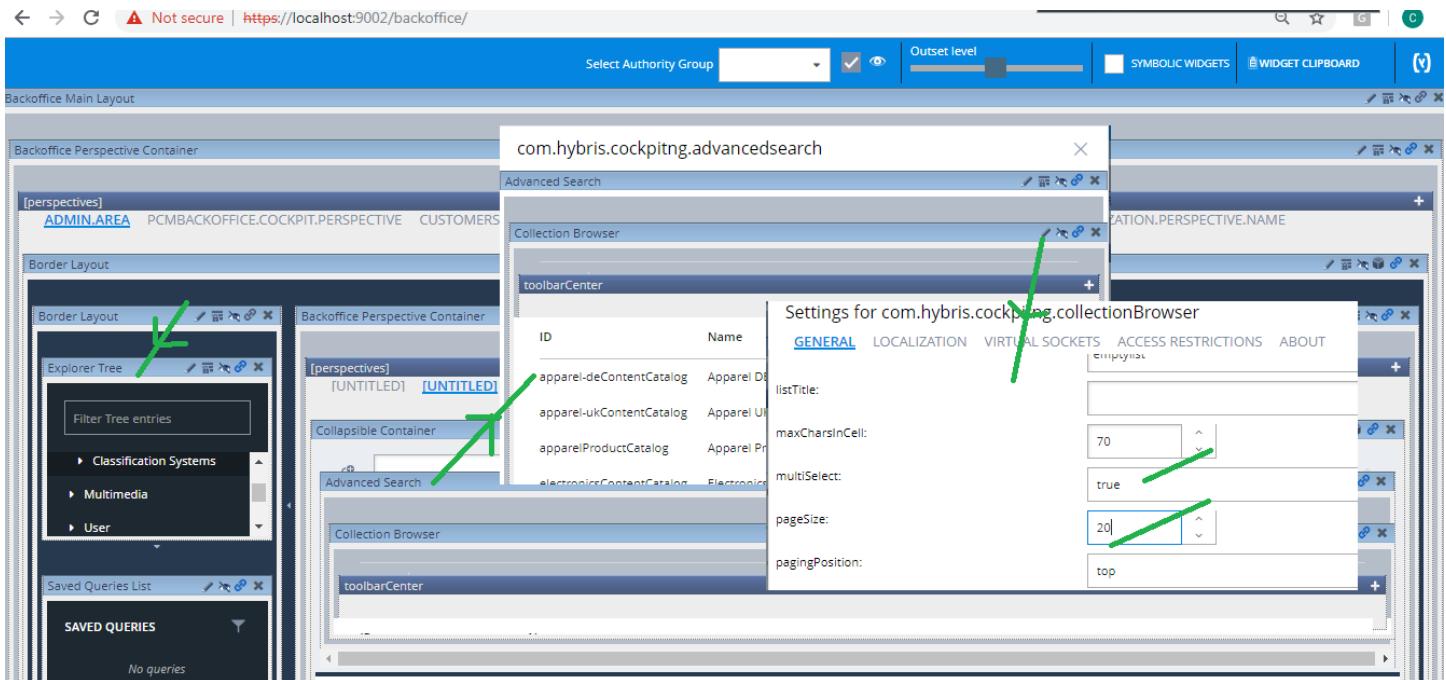
Refresh

- Show widgets.xml
- » Reset to Defaults
- Show cockpit-config.xml
- » Reset to Defaults
- Reset Everything
- Widget Toolbar Color

**Q1 = By default multi product selection allowed.**

How to Disable “Multiple Product Selection”?

In Backoffice → Press F4 → Explorer Tree → Advanced Search → Edit



Commerce

Administration

Filter Tree entries

SEARCH

Product	Article Number	Identifier	Status	Catalog version
637227	Plier Set (3 Pack)	<span>● ▲ ⚡</span>	Powertools Product Catalog : Staged	
693923	P.C. Service Kit / C.S.A. Soldering Iron	<span>● ✓ ⚡</span>	Powertools Product Catalog : Staged	
1101690	FireStorm 18 Volt 6 Tool Combo Kit	<span>● ▲ ⚡</span>	Powertools Product Catalog : Staged	
1101695	Cordless Jig Saw	<span>● ▲ ⚡</span>	Powertools Product Catalog : Staged	

3 ITEMS SELECTED

Cordless Jig Saw [1101695] - Powertools Product Catalog : Staged

**Q2 = Explore Tree is dark by default. How to make light?**

In Backoffice → Press F4 → Explorer Tree → Change the Options

Backoffice Main Layout

hybris Backoffice

Settings for com.hybris.cockpitng.widgets.common.exp

GENERAL LOCALIZATION VIRTUAL SOCKETS ACCESS RESTRICT

showNestedToolbarWidget:	true
showPrimaryActions:	true
showSecondaryActions:	true
showToolbar:	false
widgetMold:	light
widgetStyleAttribute:	light
widgetStyleClass:	dark

ADD SETTING

Explorer Tree

Border Layout

Border Layout

Saved Queries List

Filter Tree entries

Home

Inbox

System

Catalog

- Catalogs
- Catalog Versions
- Categories
- Products
- Product Variant Types

ID

- apparel-deContentCat
- ✓ apparel-ukContentCat
- ✓ apparelProductCatalog
- ✓ electronicsContentCat
- ✓ ElectronicsClassification
- ✓ electronicsProductCat

**Q3 = In Explorer Tree, by default we have Search Option.**

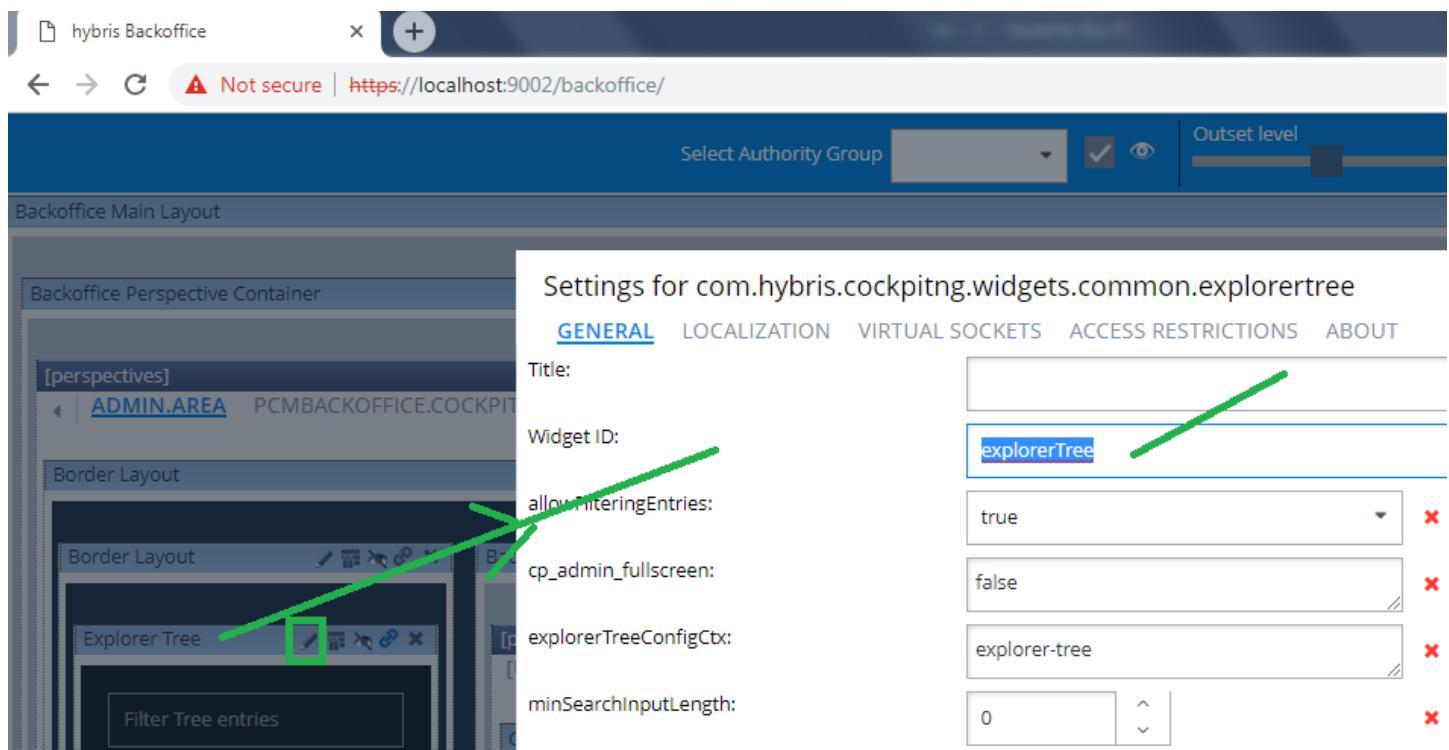
How to Disable that?

**Note** = For this, we don't have the configuration properly.

Hence, we need to customization (or) Code changes.

**Solution =**

**Step 1 = From Orchestration mode – Find the Widget ID = **explorerTree****



**Step 2 = Take the widget ID & go to “backoffice-widget.xml (or) \*-widget.xml” & do the corresponding code changes.**

Go to Eclipse → Search that widget id in “**backoffice-widgets.xml**”.

Then do the corresponding changes.

Photon backoffice/resources/backoffice-widgets.xml - Eclipse IDE

File Edit Source Navigate Search Project Run SAP Hybris [y] Hybris Window Help

Project Explorer chennatraining... ChennaBlogM... ChennaPostM... acceleratororc... catalog-item

```

243     <setting key="widgetStyleAttribute" type="String"></setting>
244     <setting key="widgetStyleClass" type="String"></setting>
245     <virtual-sockets/>
246   </widget>
247   <widget id="backofficeMainSlot" widgetDefinitionId="com.hybris.backoffice">
248     <widget id="Hmc2" widgetDefinitionId="com.hybris.cockpitng.borderlayout">
249       <widget id="explorerTree" widgetDefinitionId="com.hybris.cockpitng.widgets.actions">
250         <widget id="explorerTreeSelectionProxy" widgetDefinitionId="com.hybris.cockpitng.widgets.actions">
251           <setting key="typeNodeExpression" type="String">#root in</setting>
252             <virtual-sockets>
253               <output id="typeNode" type="com.hybris.backoffice.nav"></virtual-sockets>
254             </widget>
255           <widget id="explorerTreeConditionEvaluator" widgetDefinitionId="com.hybris.cockpitng.widgets.actions">
256             <widget id="explorerTreePropExtractor" widgetDefinitionId="com.hybris.cockpitng.widgets.actions">
257               <setting key="expression" type="String">name</setting>
258             </widget>
259           </widget>

```

File Edit Source Navigate Search Project Run SAP Hybris [y] Hybris Window Help

Project Explorer backoffice-widgets.xml

```

331   <widget id="hmc2list" widgetDefinitionId="com.hybris.cockpitng.collectionBrowser">
332     slotId="nestedWidget" template="false"
333     <setting key="listSubtitle" type="String">emptylist</setting>
334     <setting key="socketDataType_ST" type="String">java.lang.Object</setting>
335     <setting key="widgetStyleAttribute" type="String"></setting>
336     <setting key="pageSize" type="Integer">10</setting>
337     <setting key="maxCharsInCell" type="Integer">70</setting>
338     <setting key="actionSlotComponentId" type="String">listviewactions</setting>
339     <setting key="multiSelect" type="Boolean">true</setting>
340     <setting key="colConfigCtxCode" type="String">listview</setting>
341     <setting key="itemRenderer" type="String">listViewRenderer</setting>
342     <setting key="listTitle" type="String"></setting>
343     <setting key="sortableListHeader" type="Boolean">true</setting>
344     <setting key="widgetStyleClass" type="String"></setting>
345     <setting key="asyncLoading" type="Boolean">true</setting>
346     <setting key="sendItemsOnSelect" type="Boolean">true</setting>
347     <setting key="dragAndDropConfigCtx" type="String" value="dragItemsOnDComponent"/>
348   </widget>
349   <setting key="advancedSearchConfigCtxCode" type="String">advanced-search</setting>
350   <setting key="widgetStyleAttribute" type="String"></setting>
351   <setting key="addRowPosition" type="String">Bottom</setting>
352   <setting key="widgetStyleClass" type="String"></setting>
353   <setting key="enableNestedWidgetView" type="Boolean">true</setting>
354   <setting key="isNestedObjectCreationDisabled" type="Boolean">true</setting>
355   <setting key="simpleSearchConfigCtxCode" type="String">simple-search</setting>

```

File Edit Source Navigate Search Project Run SAP Hybris [y] Hybris Window Help

Project Explorer backoffice-widgets.xml organization-backoffice-widgets.xml commerce-services-backoffice-widgets.xml

```

41   <virtual-sockets/>
42   </widget>
43   <setting key="listSubtitle" type="String"></setting>
44   <setting key="socketDataType_ST" type="String">java.lang.Object</setting>
45   <setting key="widgetStyleAttribute" type="String"></setting>
46   <setting key="pageSize" type="Integer">10</setting>
47   <setting key="maxCharsInCell" type="Integer">70</setting>
48   <setting key="actionSlotComponentId" type="String">listviewactions</setting>
49   <setting key="multiSelect" type="Boolean">false</setting>
50   <setting key="colConfigCtxCode" type="String">listview</setting>
51   <setting key="itemRenderer" type="String">listViewRenderer</setting>
52   <setting key="listTitle" type="String"></setting>
53   <setting key="sortableListHeader" type="Boolean">true</setting>
54   <setting key="widgetStyleClass" type="String"></setting>
55   <setting key="asyncLoading" type="Boolean">true</setting>
56   <virtual-sockets/>
57 </widget>
```

**Note** = “In SAP – We have ABC Customer Classification”.

**Business Scenario** = ABC Classification is a frequently used analytical method to classify objects (Customers, Products or Employees) based on a particular measure (**Revenue or Profit**).

Your customers are classifying into three classes A, B and C according to the sales revenue they generate.

Threshold values used for individual ABC classes. **Example** =

All customers generating profit of **0 – 20K** belong to **class C**

All customers generating profit of **> 20K – 80K** belong to **class B**

All customers generating profit of **> 80K** belong to **class A**.

The screenshot shows the SAP Fiori launchpad with the 'ABC Reward Status Level' app selected. The main area displays three tables: 'Reward Status Level Configuration in Backoffice', 'Reward Status Levels in Backoffice', and 'Customer Rewards Status'. The first table shows reward levels: Class C (Silver, 0 - 20k), Class B (Gold, >20k - 80k), and Class A (Platinum, >80k). The second table lists reward levels: Silver, Platinum, and GOLD. The third table shows customer counts by class: Class A (34 Customers), Class B (140 Customers), and Class C (94 Customers). A green annotation on the right side of the screen reads: 'In SAP -- We have ABC Customer Classification. That means, your customer is classified into Class A / B / C based on Threshold value. Similar to Credit Cards. Silver ... Gold ... Platinum'. Another annotation at the bottom right says: 'Right now, we are managing all activities from SAP. You can say my client want these activities to be happened from SAP Comm.'

Above scenario similar to below...

The screenshot shows the SAP Fiori launchpad with the 'Chenna RRRS Training Courses' app selected. The left sidebar shows navigation options: B2B Commerce, Chenna RRRS Training Courses (highlighted with a green border), Chenna RRRS Courses, Chenna RRRS Tables, Personalization, Subscription, and Entitlements. The main area displays a table of courses with columns: Chenna RRRS Course Co..., Chenna RRRS Course Name, and Chenna RRRS Course Durati... (partially cut off). The table rows are: SAPspartacus (SAP Sparatacus, 40 Hrs), CLang (C Lang, 60 Hrs), SAPCommTech (SAP Comm Tech Only, 60 Hrs), and SAPCommTechoFun (SAP Comm Techo Fun, 120 Hrs). A green border highlights the 'Chenna RRRS Training Courses' app icon and the first row of the table.