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

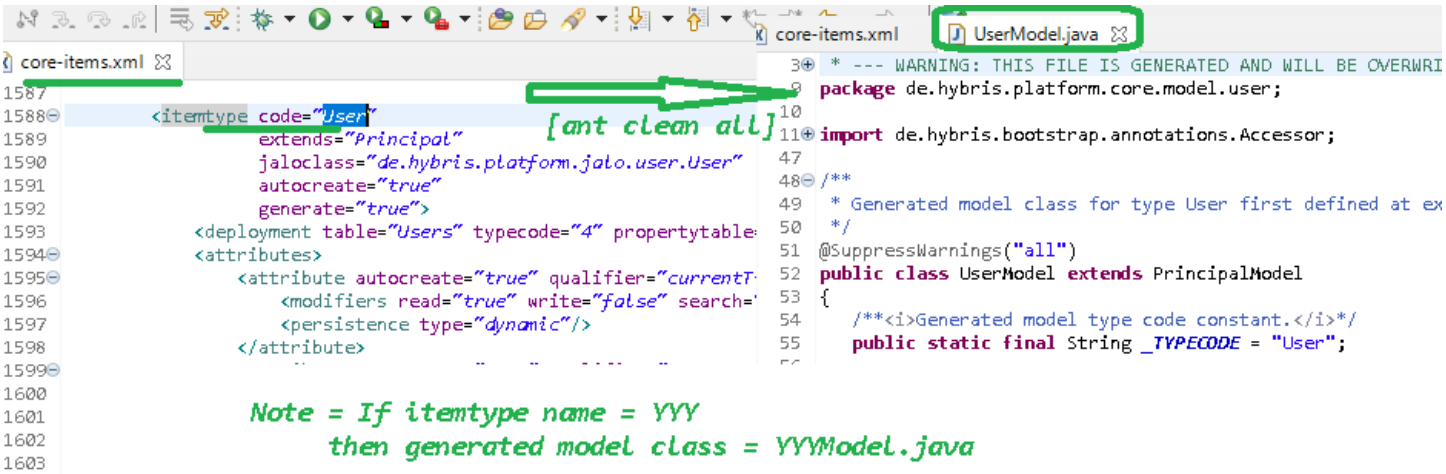
Scenario = Model Classes

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

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

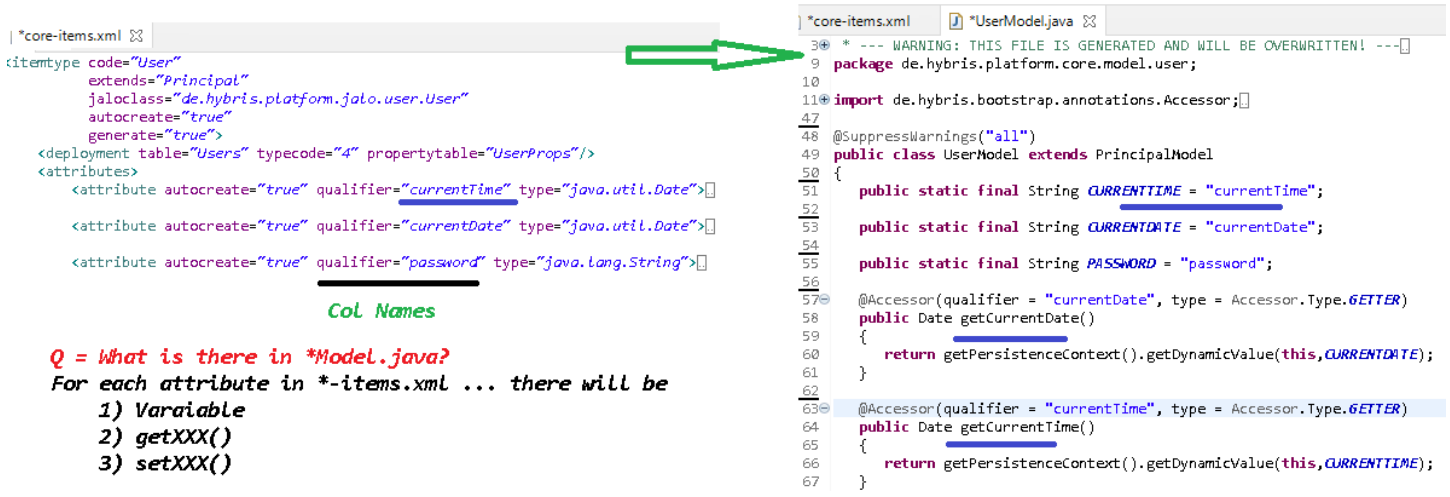
SAP Comm generates Model classes for each defined item type.

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



Q = What model classes contains?

Model class provides getter and setter method for attributes.



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

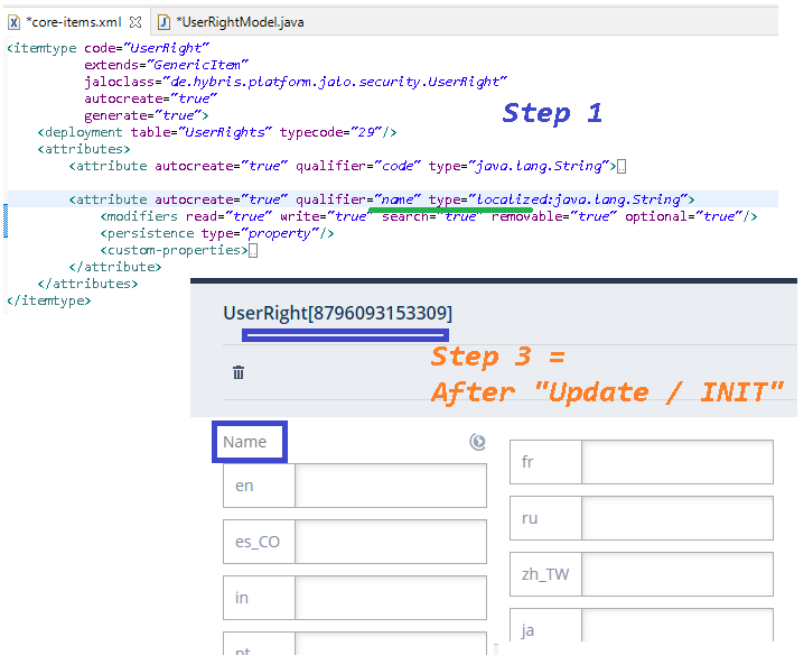
1) Let's say we have 8 attibutes in *-items.xml then
   In *Model.java – We will have
   === 8 Varaibles
   === 8 setXXX Methods
   === 8 getXXX Methods
   That means – For each attribute there will be 1 varaible … 1 setXXX method & 1 getXXX method.

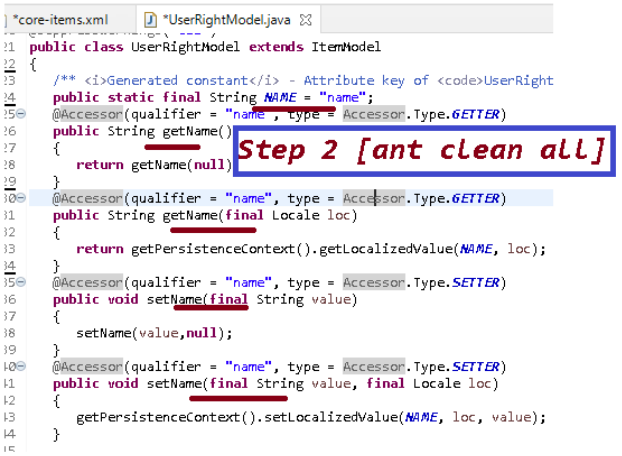
2) If the attribute <persistence type="**dynamic**" ==> then this attribute will **not be created** in DB as a **Col** and will **not be in *Model.java**.
3) If the attribute <persistence type=" **property**" ==> then this attribute will **be created** in DB as a **Col** and **will be in *Model.java**.
4) If the attribute type="**localized:java.lang.String**" ==> then this attribute will be created in DB.
   In *Model.java – We will have: -

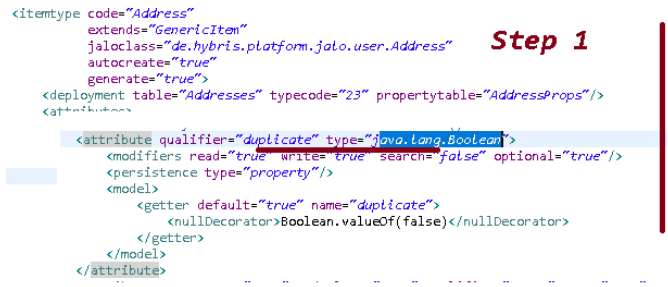
1) 1 variable

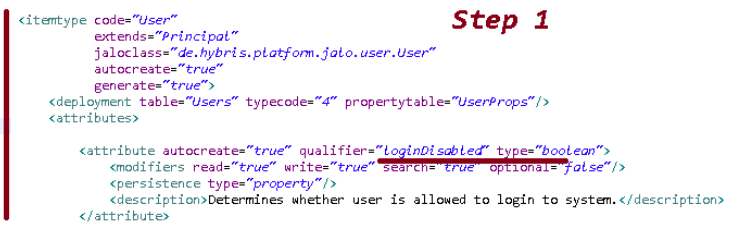
2) 2 setXXX() =

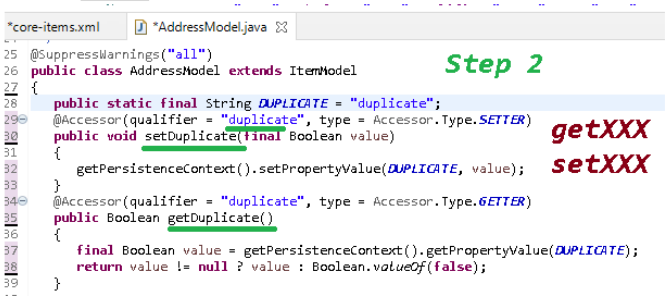
3) 2 getXXX() =

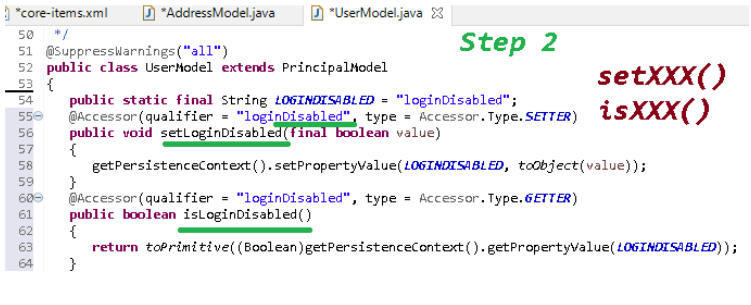


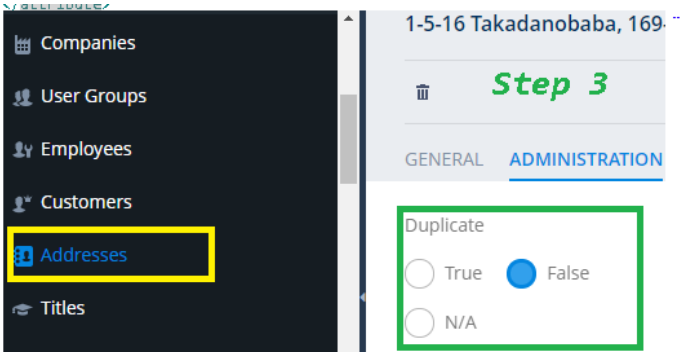


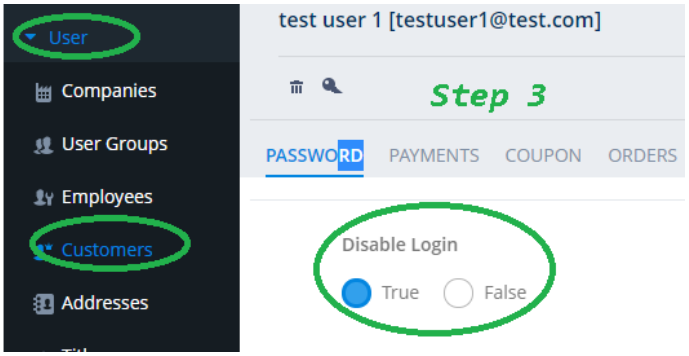










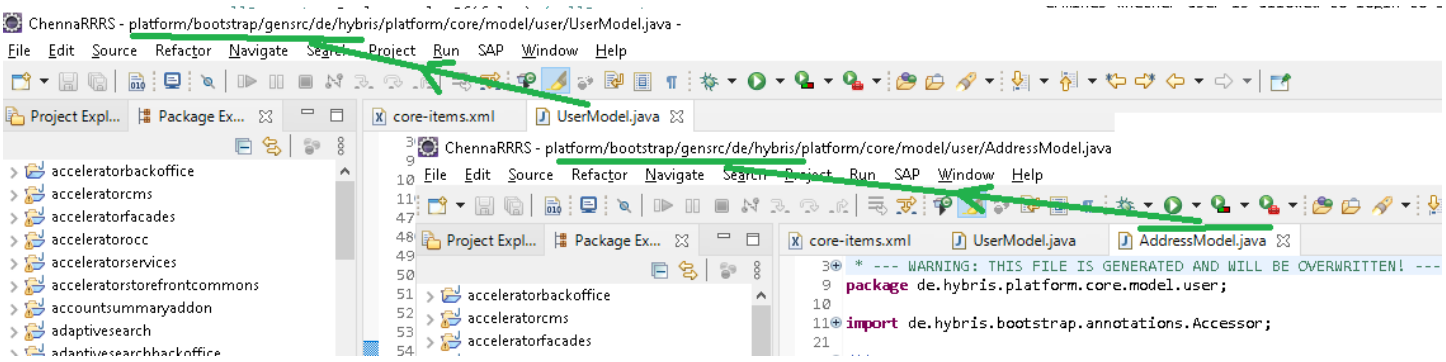


Contact Us = ChennaReddyTraining@RRRS.CO.IN

*core-items.xml | *core-advanced-deployment.xml

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

EmpId	EmpName	EmpSal
101	Chenna	10$
102	ABC	15$

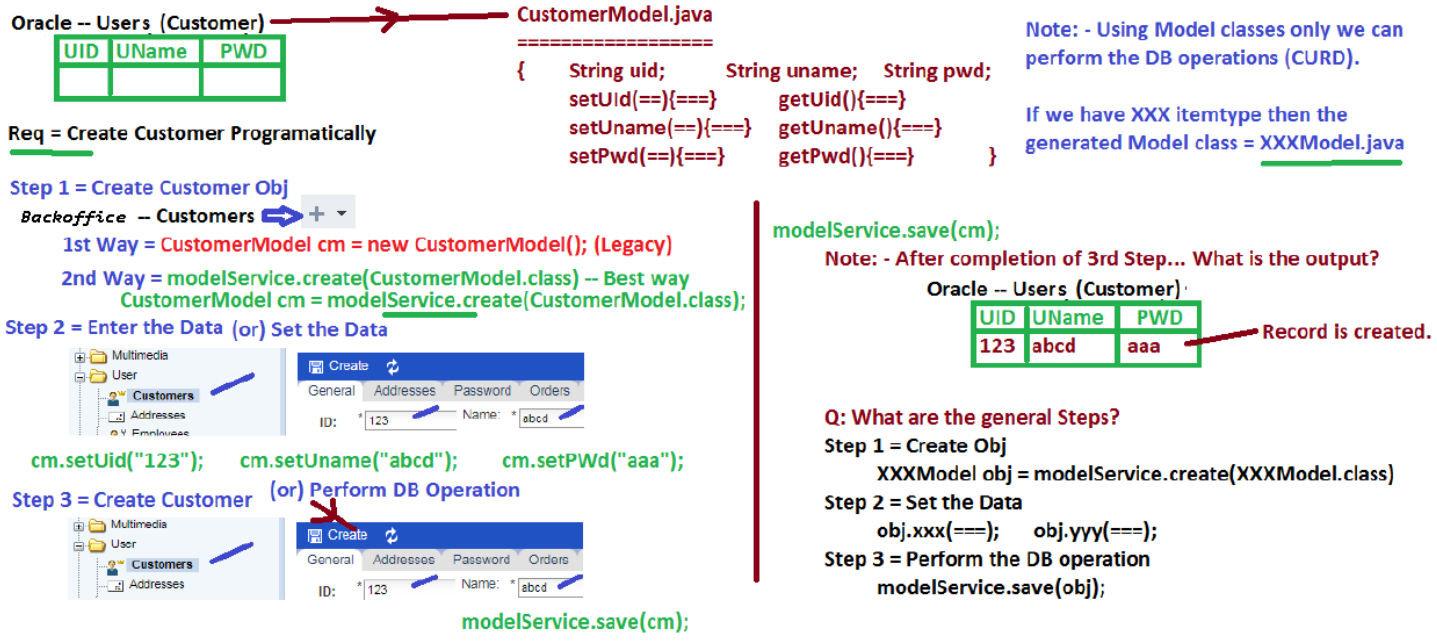
POJO Class (Java) == Model Class (Hybris)
Contains: -
1) For every DB Table Col there will be 1 variable
2) For every DB Table Col there will be 1 setXXX() & 1 getXXX()

Java Side

Emp.java (POJO Class): -

```
public class Emp
{
        String EmpId;
        String EmpName;
        String EmpSal;
        setEmpId(==){===}
        getEmpId(){===}
        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?



Conclusion = With this exmaple – 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.

Step 1 = DB Design

```
*-items.xml
<itemtype code="Emp1"
    <deployment table="Emp" typecode="29"/>
    <attributes>
        <attribute qualifier="EmpId" type="String">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="EmpName" type="String">
            <persistence type="property"/>
        </attribute>
    </attributes>
</itemtype>
```

Hybris

Hybris Deployment Name = DB Table Name

```
db.url=jdbc:hsqldb:file:${HYBRIS_
e;hsqldb.defrag_limit=2;hsqldb.ni
db.driver=org.hsqldb.jdbcDriver
db.username=sa
db.password=
```

This Hybris DB design ... will be converted into equivalent native table during the INIT / Update

Step 3

Emp (Table): -

==============

EmpId	EmpName

Database (HSQLDB) (Today)

```
db.url=jdbc:oracle:thin:@<host>:1521:<sid>
db.driver=oracle.jdbc.driver.OracleDriver
db.username=<username>
db.password=<password>
```

Perfrom INIT / Update

Emp (Table): -

==============

EmpId	EmpName
1010	Chenna

Database (Oracle) (Tomorrow)

Step 2 = Build ant clean all

After ant clean all successfull, you will get *Model.java file (Model class/ POJO)

If you have itemtype = XXX then after build, you will get XXXModel.java

```
Emp1Model.java{
    String EmpId;      StringName;
    setEmpId(){==}     getEmpId(){==}
    setEmpName(){==}
    getEmpName(){==}
}
```

This model obj used to perfrom DB operations

Q: How to perfrom the DB Operation (I want to insert 1 record into Emp Table)? =

```
Step 1 = Create Obj
        1st Way -->  Emp1Model Ob = new Emp1Model(); (Old)
        2nd Way -->  getModelService().create(Emp1Model.java) (Latest)
Step 2 = Set the Data
        ob.setEmpId("1010");       ob.setEmpName("Chenna");
Step 3 = Perfrom DB Operations --->   getModelService.save(ob);
```

In Hybris, We have Type = String, but in DB It's created with varchar(255) length.

Q: Where this conversion is happening (Hybris attribute type String is converted into DB Col varchar(255)?

```
<itemtype code="ExpressionAttributeDescriptor">
    <attributes>
        <attribute qualifier="defaultValueExpression"
            type="java.lang.String">
            <persistence type="property"/>
        </attribute>
    </attributes>
</itemtype>
```

- ATTR2VALUEREL609SN
- ATTRIBUTEDESCRI87SN
- ATTRIBUTEDESCRIPTORS
- ATTRIBUTEDESCRIPTORSLP
- B2BBOOKINGLINEE10005SN
- B2BBOOKINGLINEENTRY

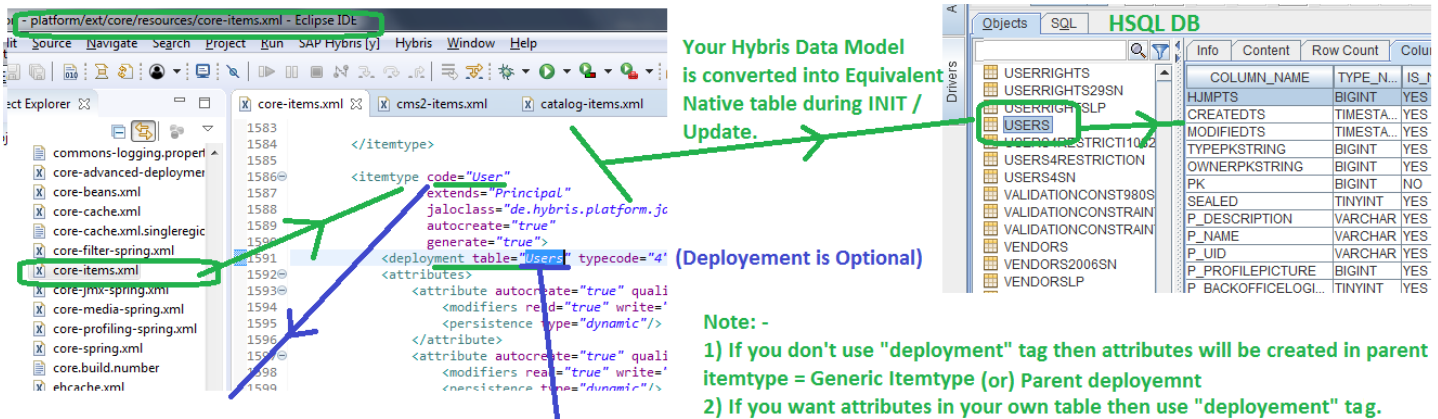
COLUMN_NAME	COLUMN_...
P_ATTRIBUTEHANDLER	255
SELECTIONDESCRIPTORPK	64
P_DEFAULTVALUEEXPRESSION	255
P_ISSOURCE	8

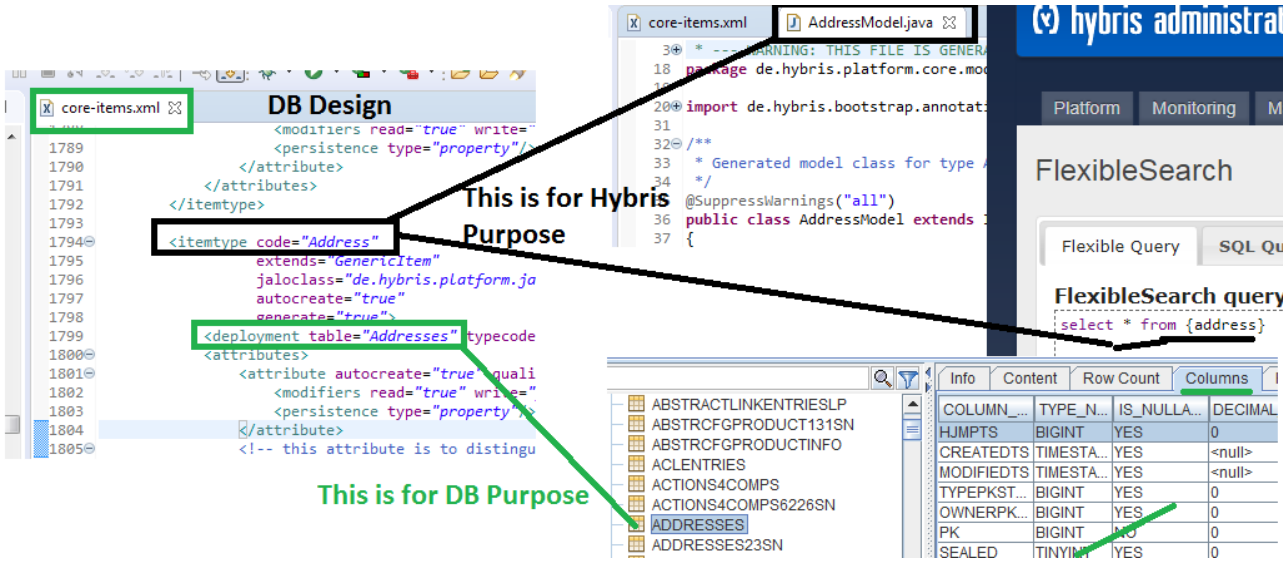
s.xml | core-advanced-deployment.xml

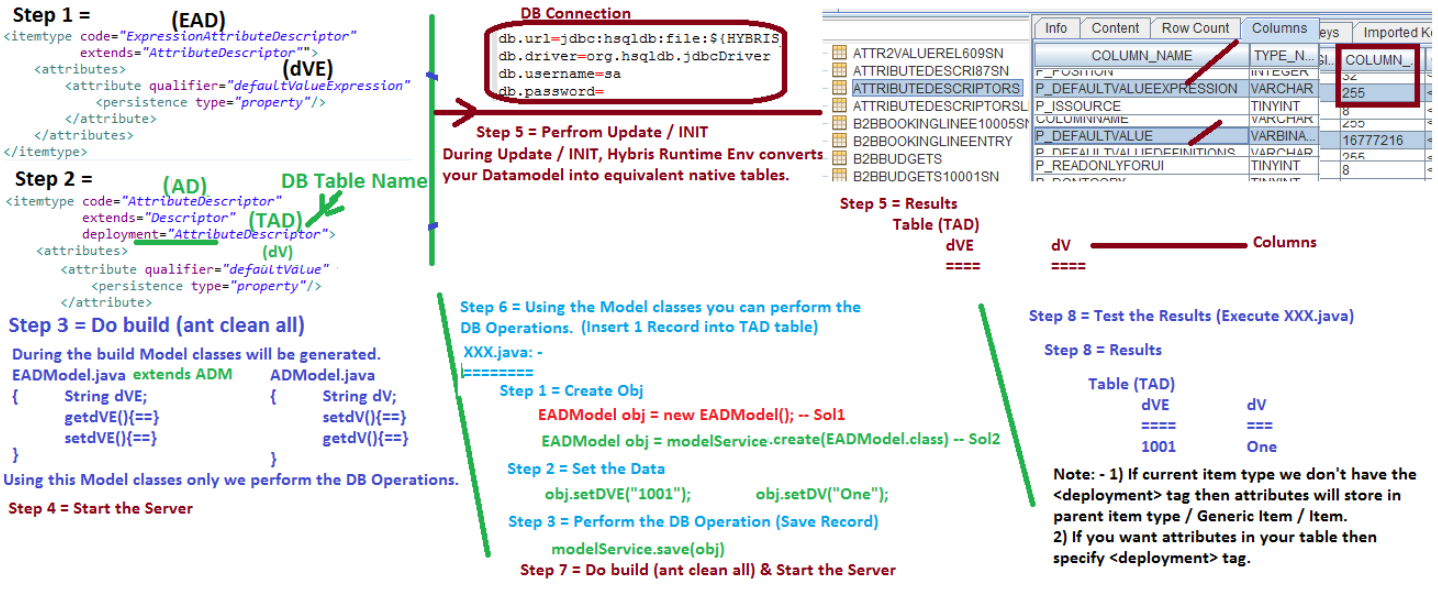
```
<database-schema database="hsqldb" primary-key="primary key" null="" not-null
    <type-mapping type="java.lang.String" persistence-type="VARCHAR(255)" />
    <type-mapping type="String" persistence-type="VARCHAR(255)" />

    <type-mapping type="java.lang.Float" persistence-type="float" />
    <type-mapping type="java.lang.Double" persistence-type="double" />
    <type-mapping type="java.lang.Byte" persistence-type="smallint" />
```

Scenario = Itemtype vs Deployment

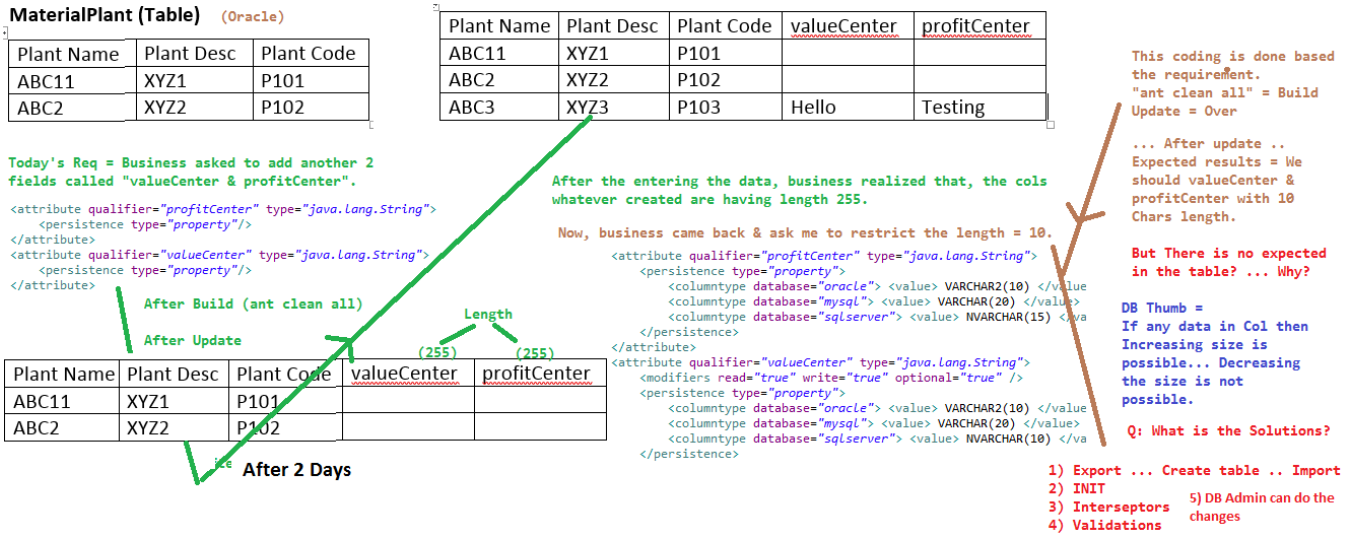


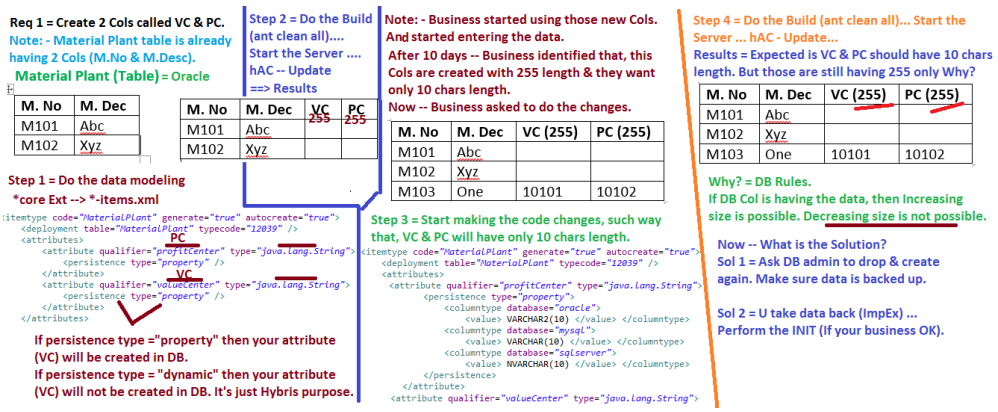




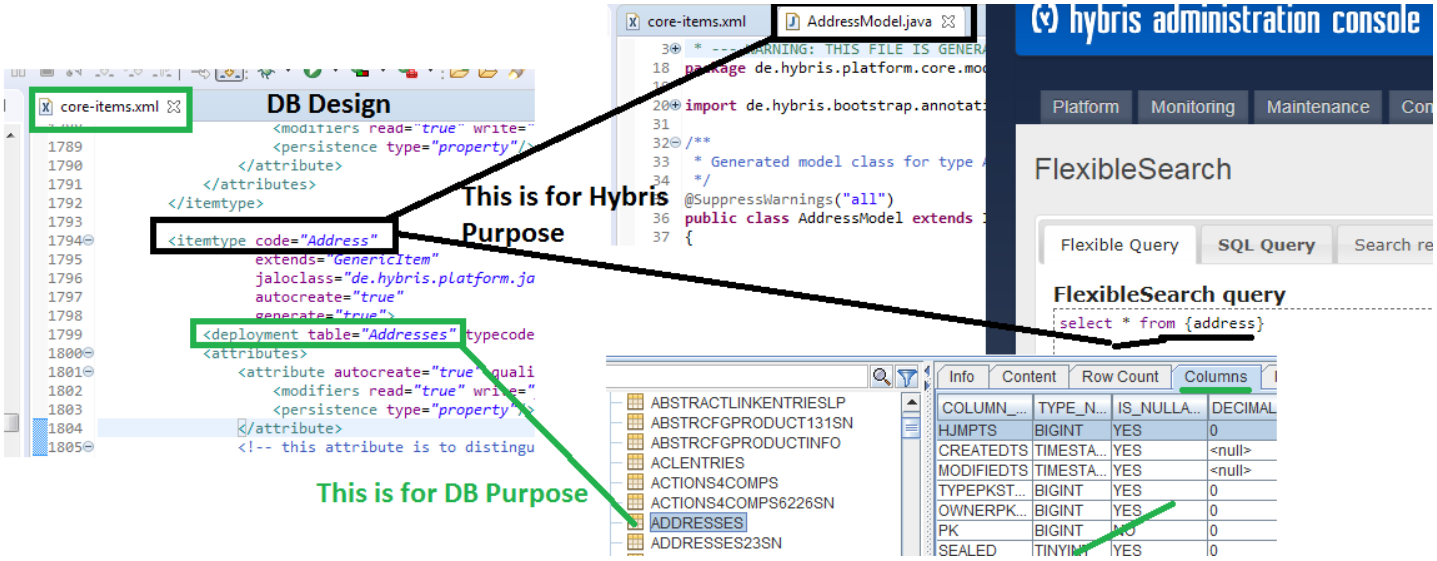
Contact Us = ChennaReddyTraining@RRRS.CO.IN

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?





Scenario = core-advanced-deployment.xml?



Scenario = Localized Attributes

Itemtype Name

```
<itemtype code="UserRight"          This is for Hybris Purpose
          extends="GenericItem"
          autocreate="true"
          generate="true">          This is for DB Purpose
    <deployment table="UserRights" typecode="29"/>
    <attributes>                    Col Name
        <attribute autocreate="true" qualifier="code" type="java.lang.String">
            <persistence type="property"/>
        </attribute>
Col     <attribute autocreate="true" qualifier="name" type="localized:java.lang.String">
        </attribute>
    </attributes>
</itemtype>
```

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.

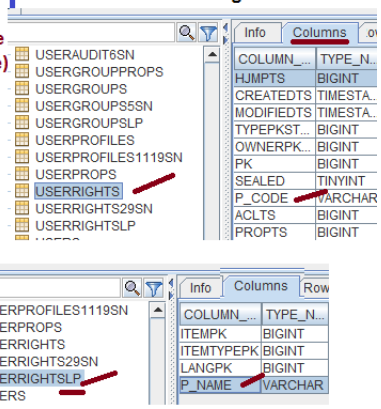
If XXX deployment is having localized attributes then ... these will be stored in separate table.
Example: - In above case, code will be stored in "UserRights" table & name will be stored in "UserRightsLP" table.

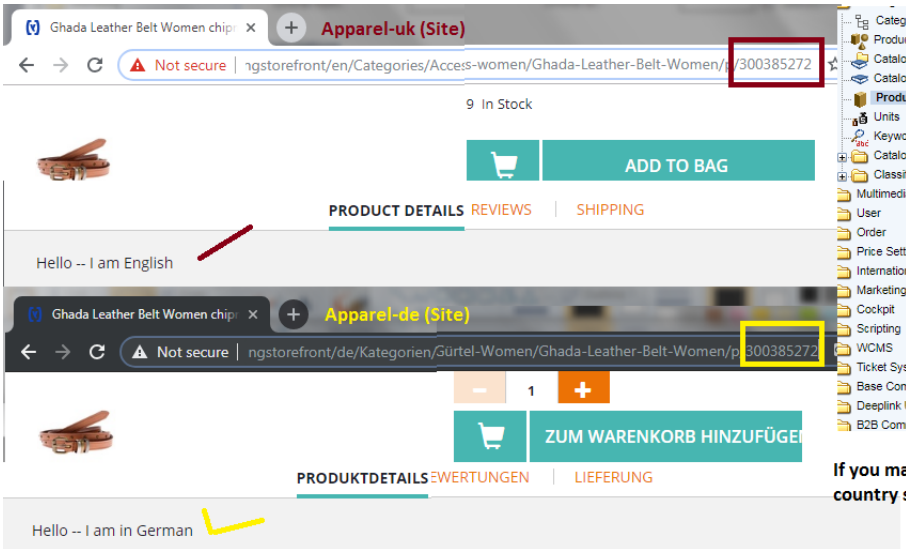
When we do the Build (ant clean all) --
For every itemtype, there will be Model class generated.
If Itemtype Name = XXX then Model class name = XXXModel.java.
Using this Model class only we perform the DB Operation (Create ... Update ... Remove)

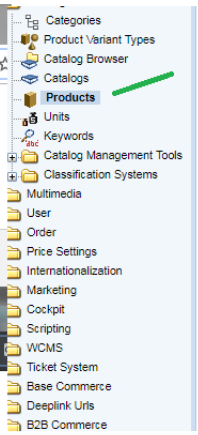
UserRightModel.java
==========

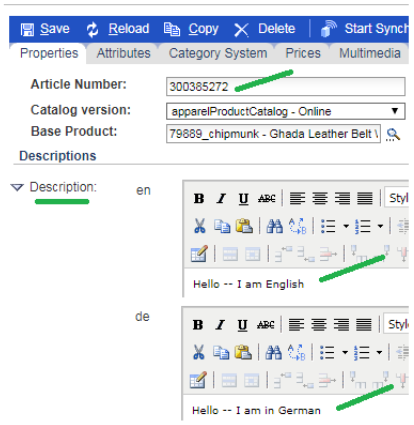
```
public class UserRightModel
{       String code;        String name;
        setCode(==);        getCode();
        setName(==);        setName(==,loc);
        getName();          getName(loc);
}
```

Let's say -- You started the Server.
& Performed hAC - Update.
Results: -
In the DB -- UserRights



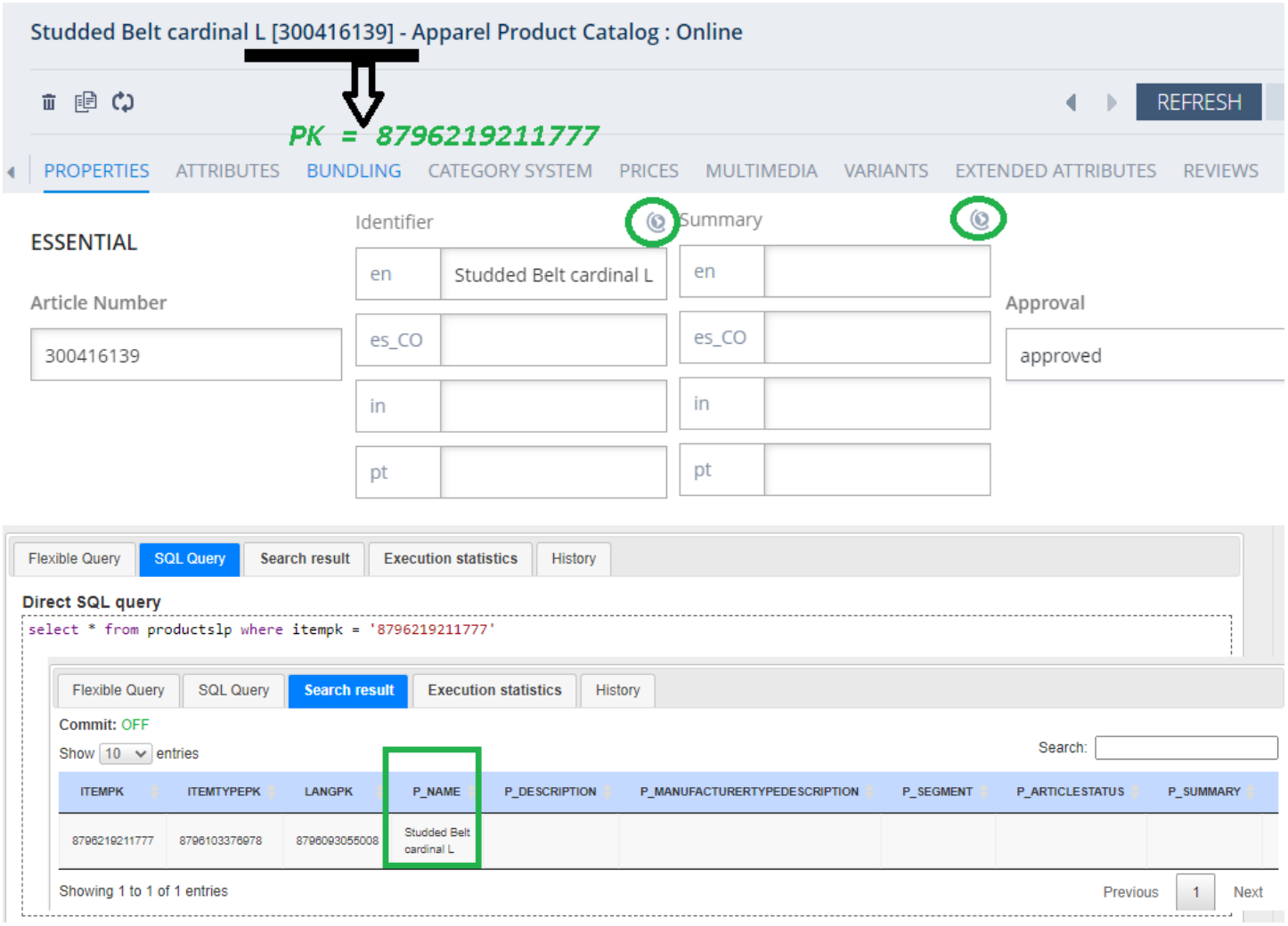


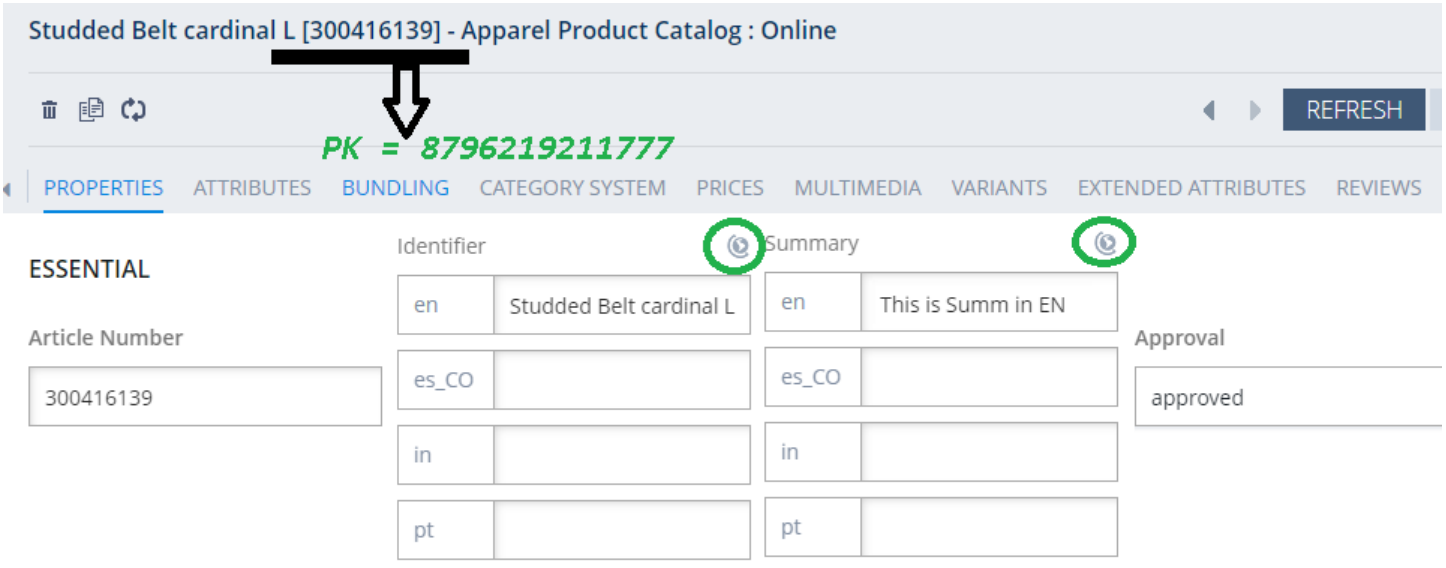


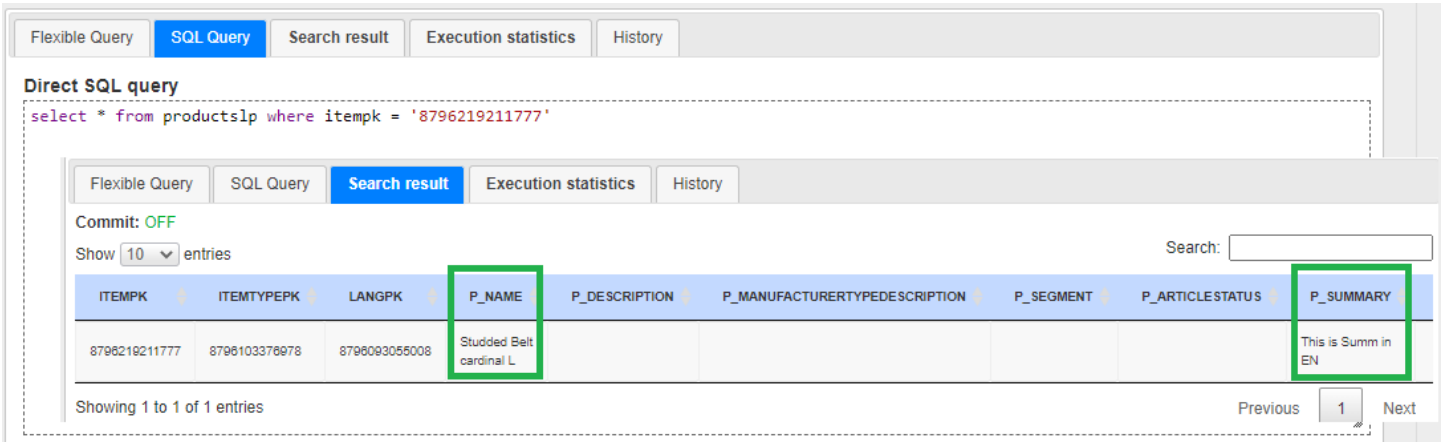


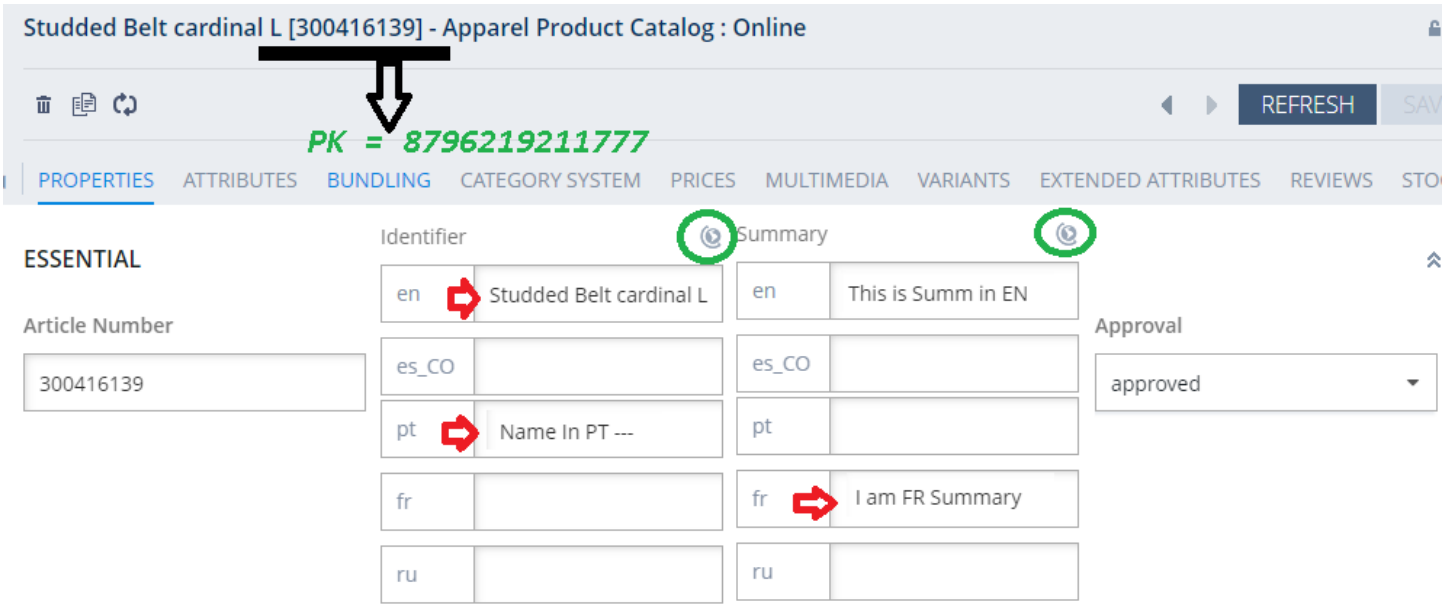
If you make a Col of type localized, then it allows you manage the country specific values (these values used to display in Storefront).

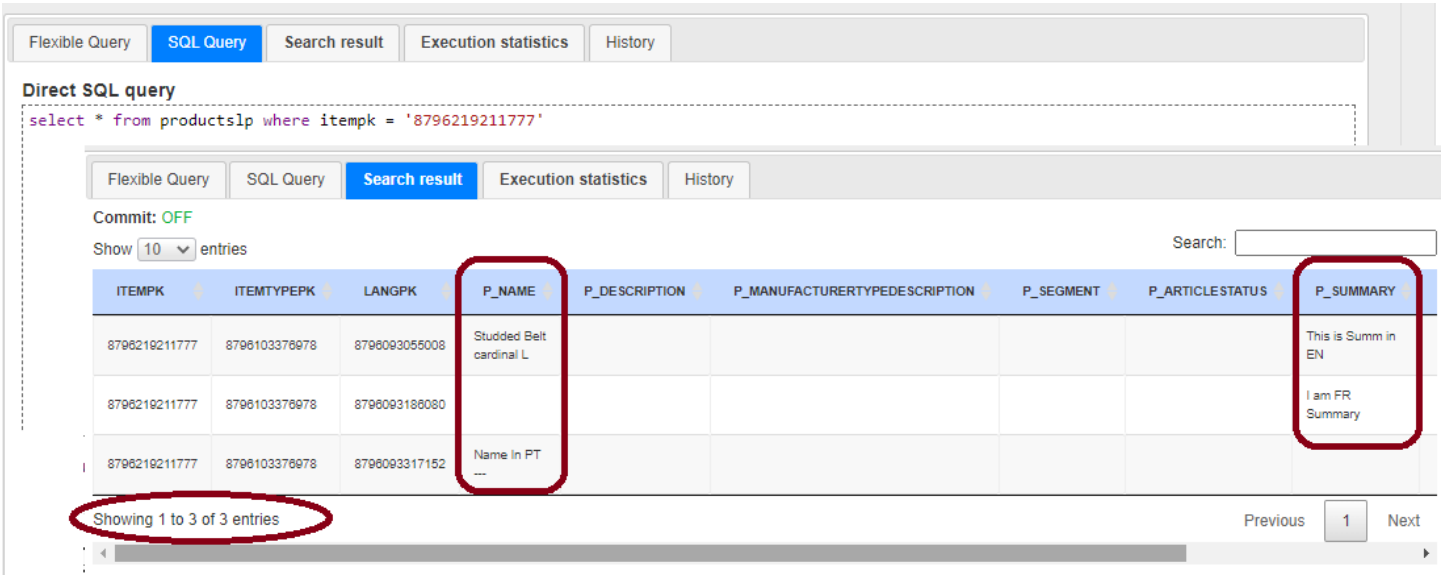
Scenario = Understand Localized ***LP** tables data?











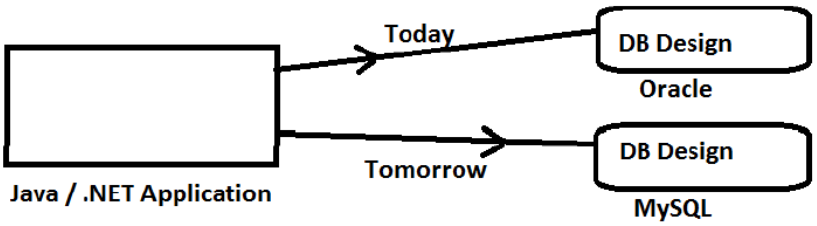
Conclusion = For 1 Language – 1 Record is maintained.

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

Scenario = Data

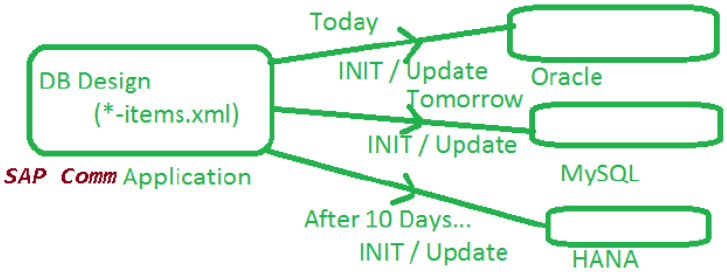
In case of “Java / .NET” application, DB design happens in native database (like – Oracle, MS SQL and …).

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**”.



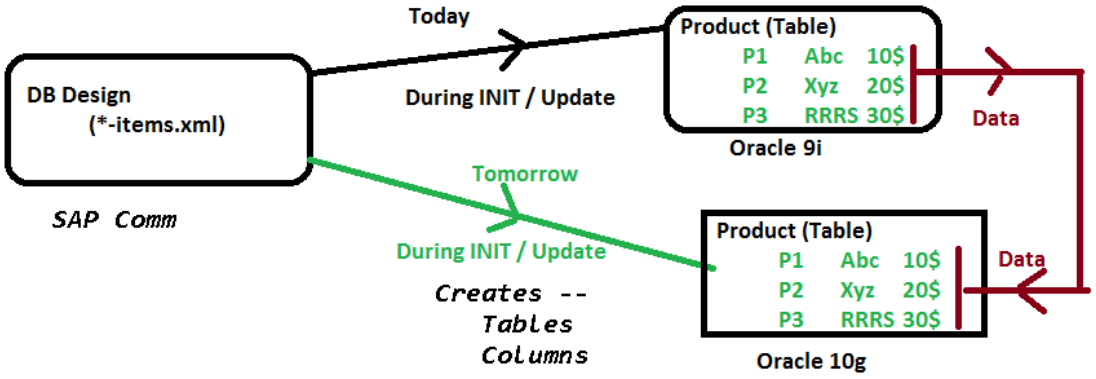
MySQL Data Type	Oracle Data Type
BIGINT	NUMBER(19, 0)
BIT	RAW
BLOB	BLOB, RAW

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) ....

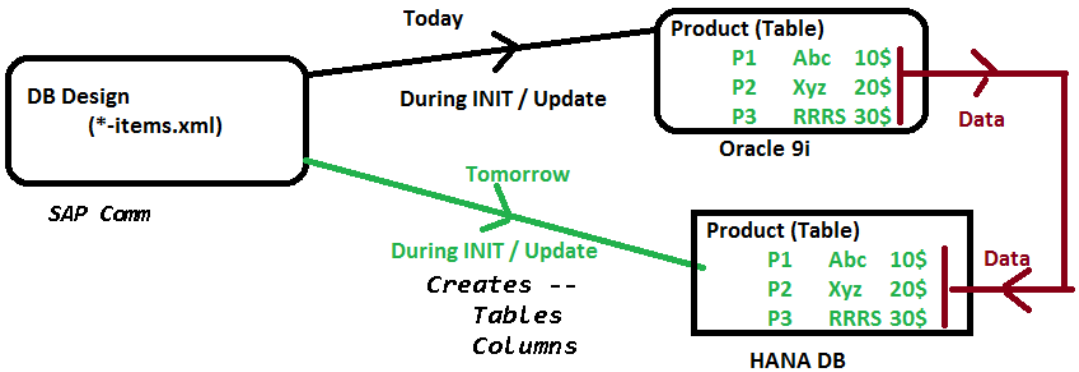


In case hybris -- We do the DB Design in hybri self.
This DB design will be conerted into Equivalent table during INIT / Update.
So -- Today you have Oracle... Tomorrow u want to move to HAHA... Then You simply do below: -

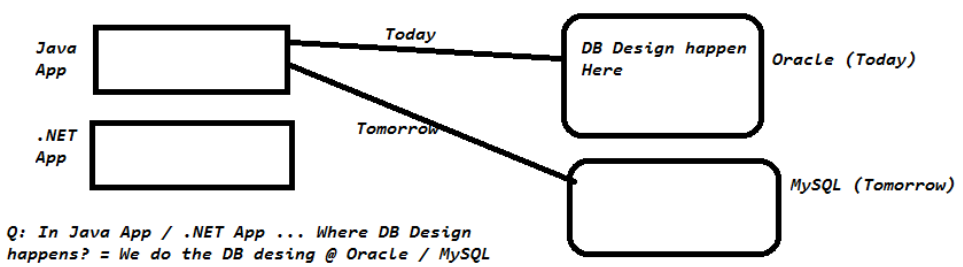
Step 1 = Config\local.properties -- Specify HANA Conn Details
Step 2 = If HANA requires Specific driver then copy it.
Step 3 = Start the Server & Perform INIT / Update.



Source = Orace Target = Oracle
Q: How to move the Data?
Sol1 = Export ...
Import in Oracle
Sol2 = ImpEx in SAP Comm

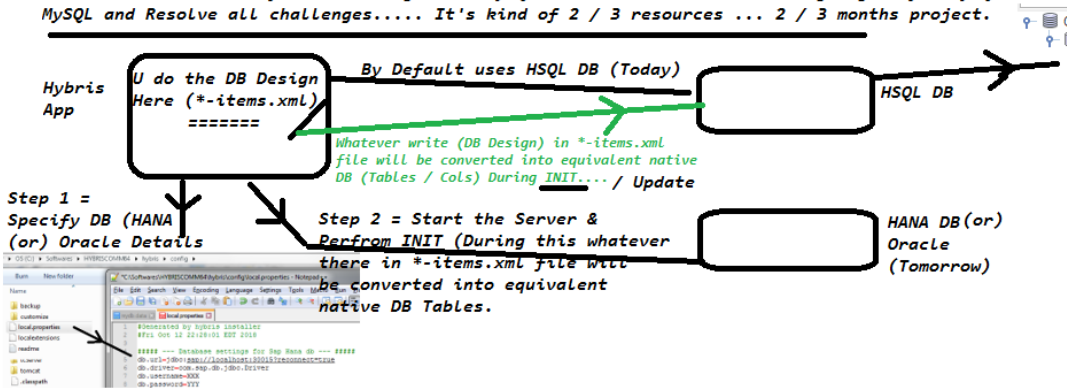


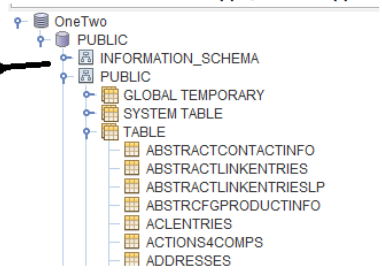
Source = Orace Target = HANA
Q: How to move the Data?
Sol = ImpEx in SAP Comm
Note: - Using ImpEx, We can do: -
1) Insert Record
2) Update Record
3) Delete Record

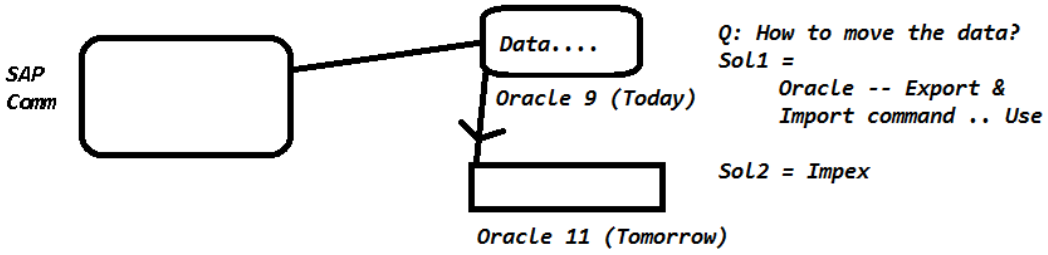


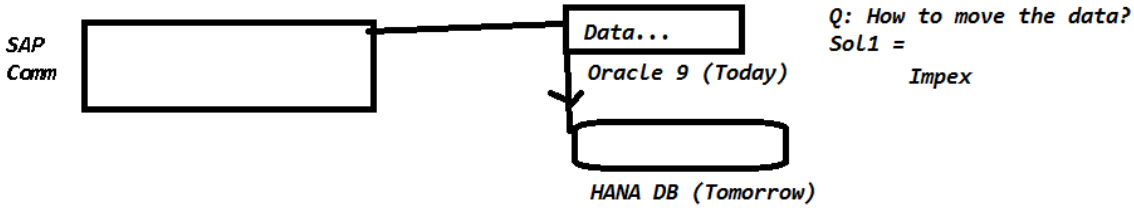
MySQL Data Type	Oracle Data Type
BIGINT	NUMBER(19, 0)
BIT	RAW
BLOB	BLOB, RAW
DATETIME	DATE
DECIMAL	FLOAT (24)

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.

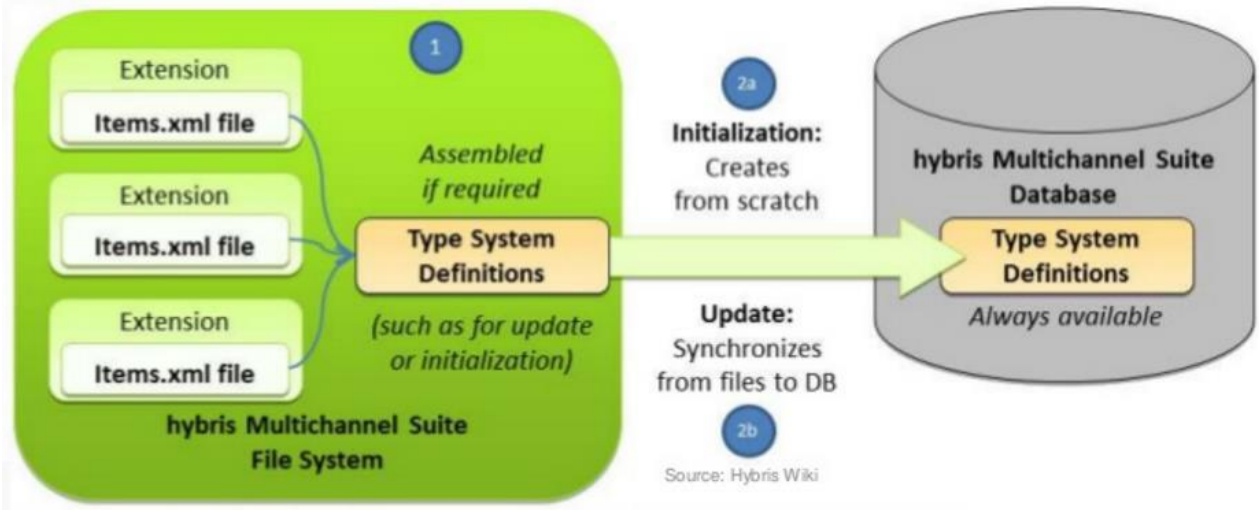








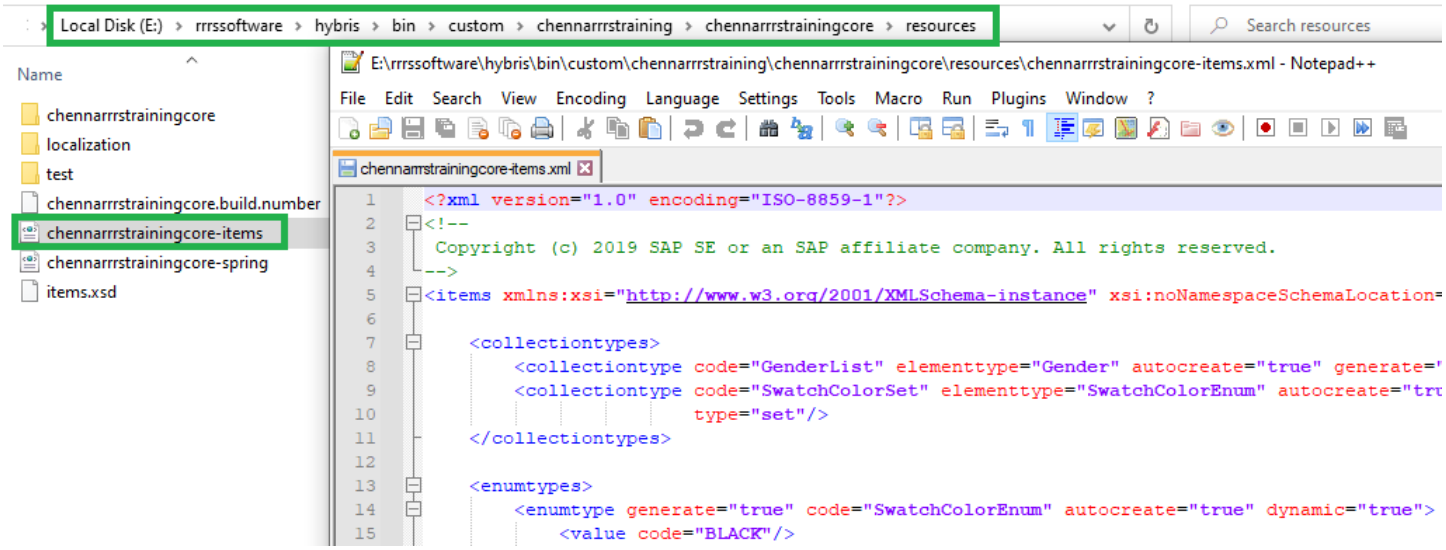
Q = Who is giving / defining this XSD? = SAP Comm / FWK.



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

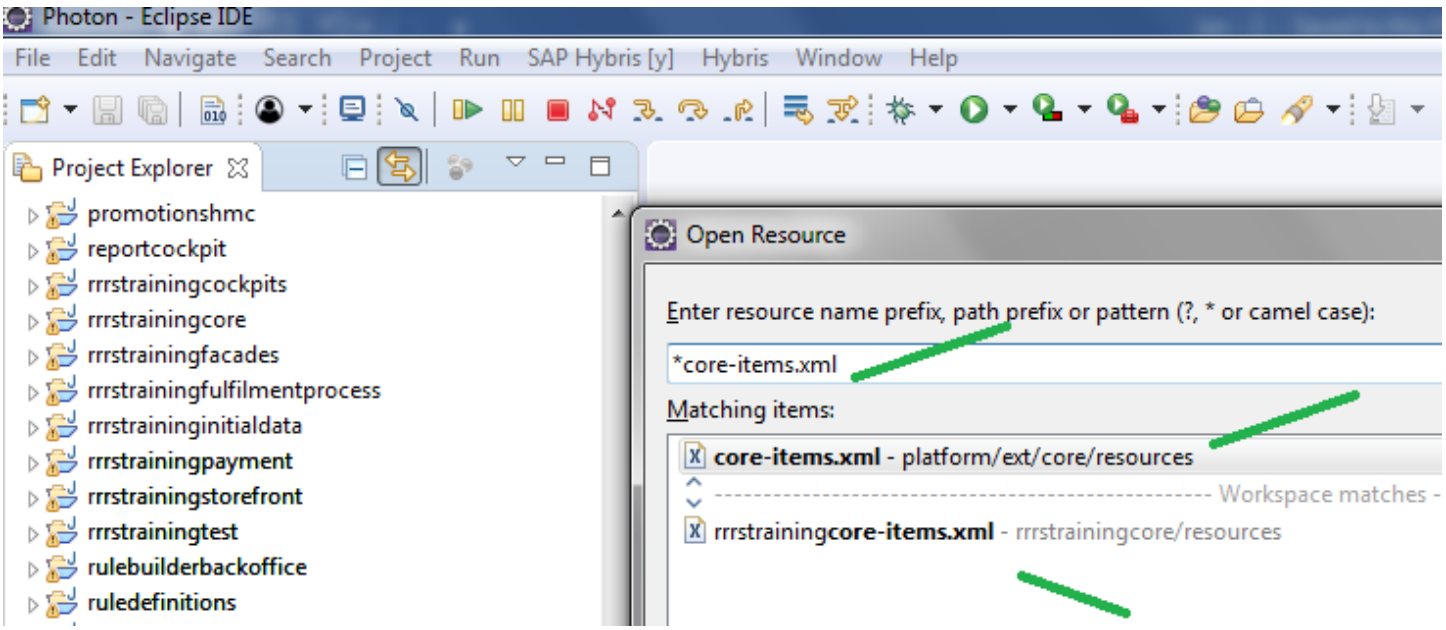
Note = Generally we do the data modeling in "*core = chennarrrstrainingcore" Ext.

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



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) chennarrrstrainingcore-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.

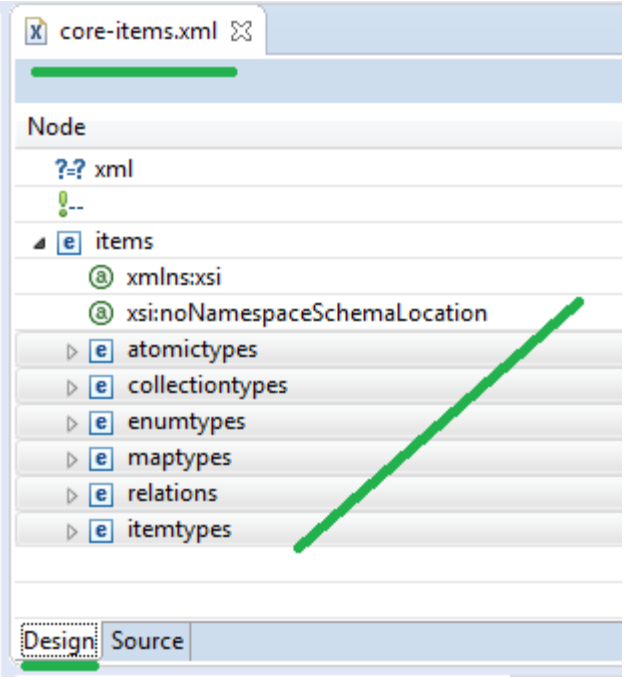
Means for data modeling we will use **these 6 tags**.

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) Collectiontype

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

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

Select User.UserID, Count(Order.OderID) 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?

User Table

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

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 possible 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**.

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

This is used to store key – value Paris.

Each key is unique & values can be duplicate.

Emp (Table)

EmpId	EmpName(Maptypes)
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) Relationtypes

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.

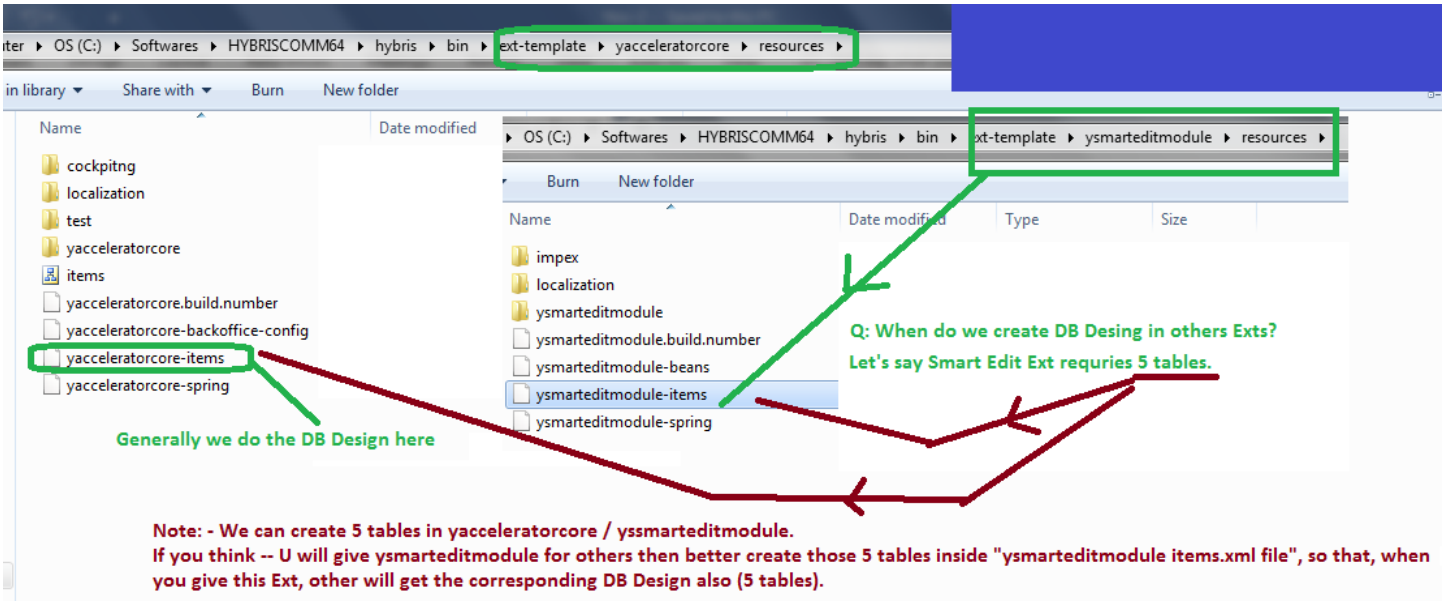
Let's say – SAP Comm is giving 50 columns for Product table.

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**



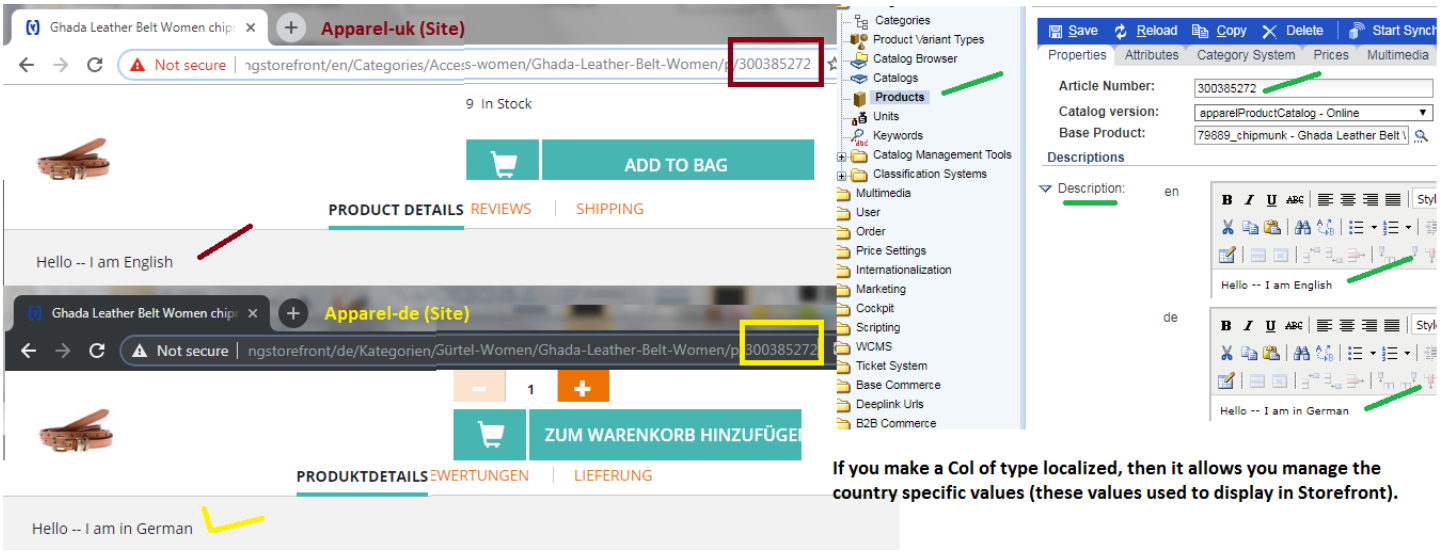
Note =Localization we do in 3 places: -

1) **PIM** Localization = Example … Product Description

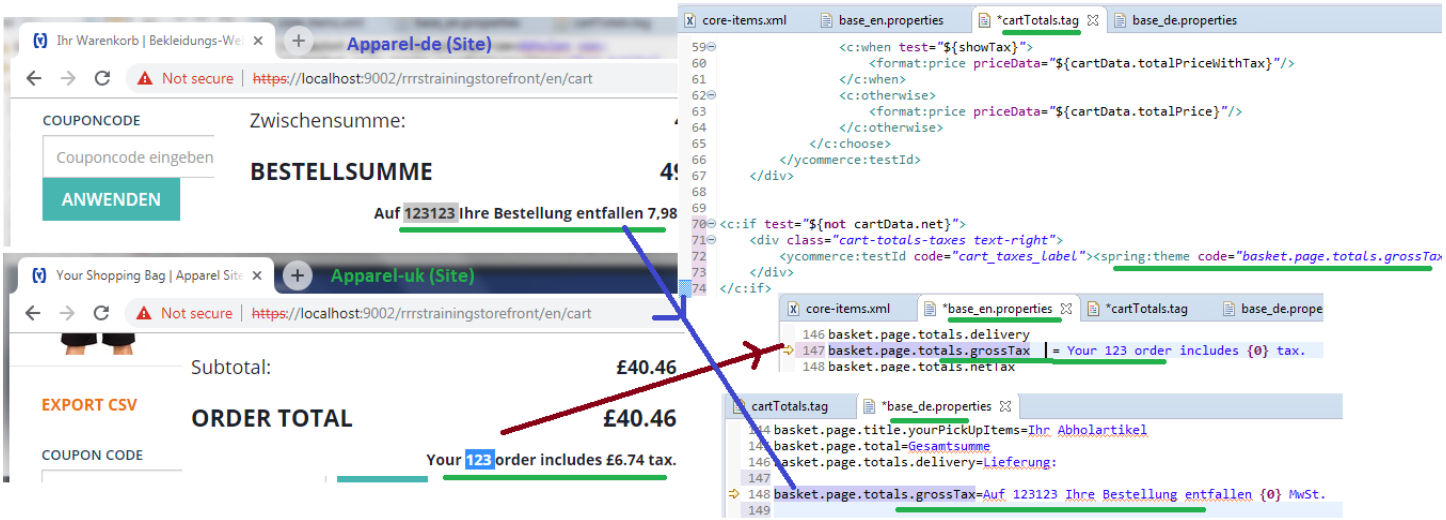
2) **Content** Localization = Example … Banner Description

3) **Programmatic** – Label localization ….

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



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



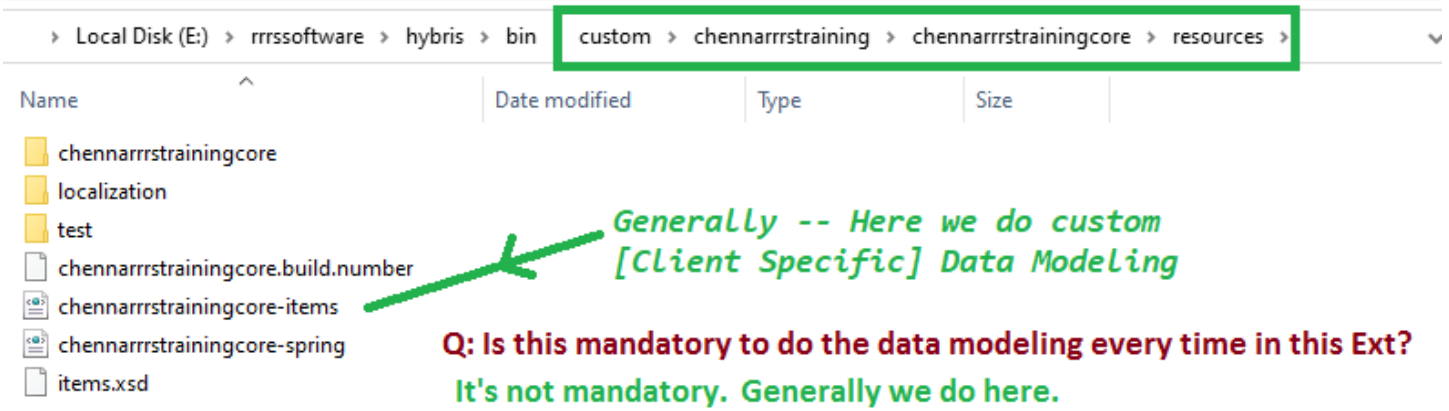
Q = Explain Important Features of "SAP Comm" Data Modeling?

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) …..



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)

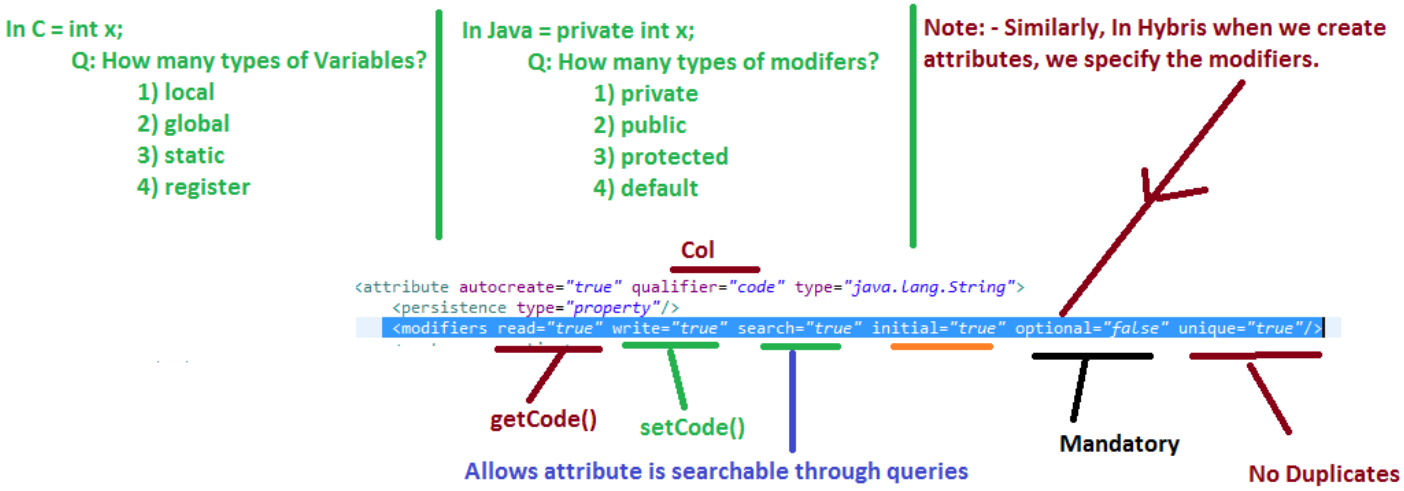
```
<itemtype code="UserRight"
          extends="GenericItem"
          jaloclass="de.hybris.platform.jalo.security.UserRight"
          autocreate="true"
          generate="true">
```

autocreate="true" -- It creates DB entries during INIT.
autocreate="false" -- It understands that this itemt 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.

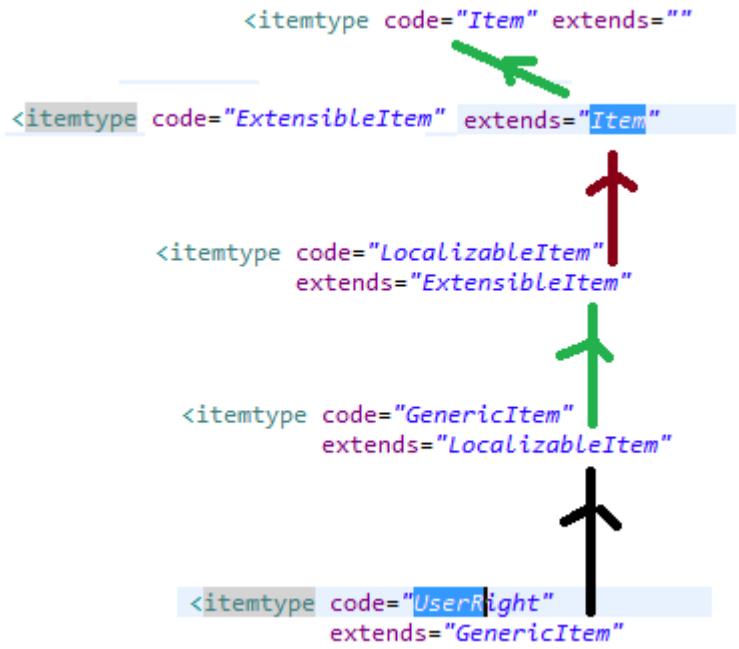
```
<itemtype code="UserRight"
          extends="GenericItem"
          jaloclass="de.hybris.platform.jalo.security.UserRight"
          autocreate="true"
          generate="true">
    <deployment table="UserRights" typecode="29"/>
    <attributes>
        <attribute autocreate="true" qualifier="code"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

Col Name



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

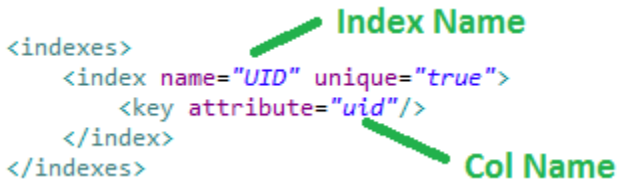
Note = Creating too many PK's in same table decreases the performance.

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

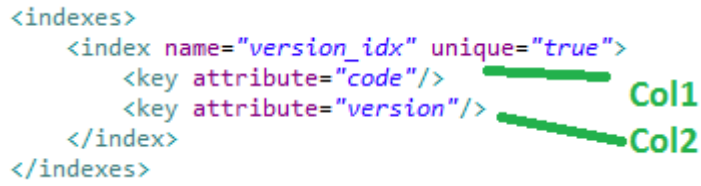
Q4 = A Composite key can have max how many Cols? = 16

====> PK = Unique + Index

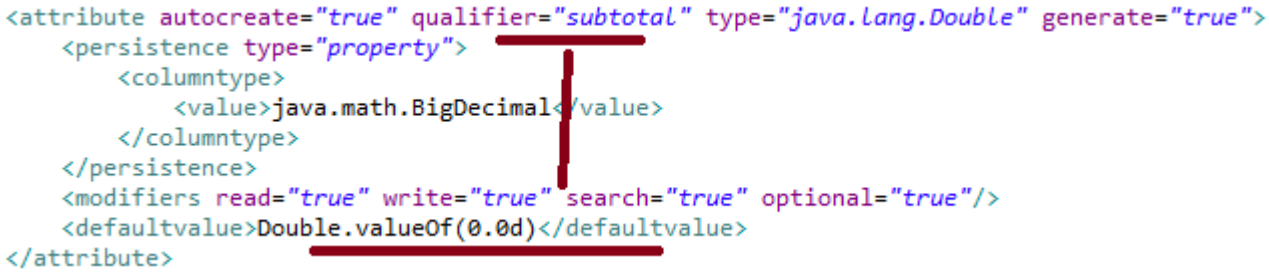
Q = How to create 1 Col Indexed?



Q = How to create >1 col indexed?



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

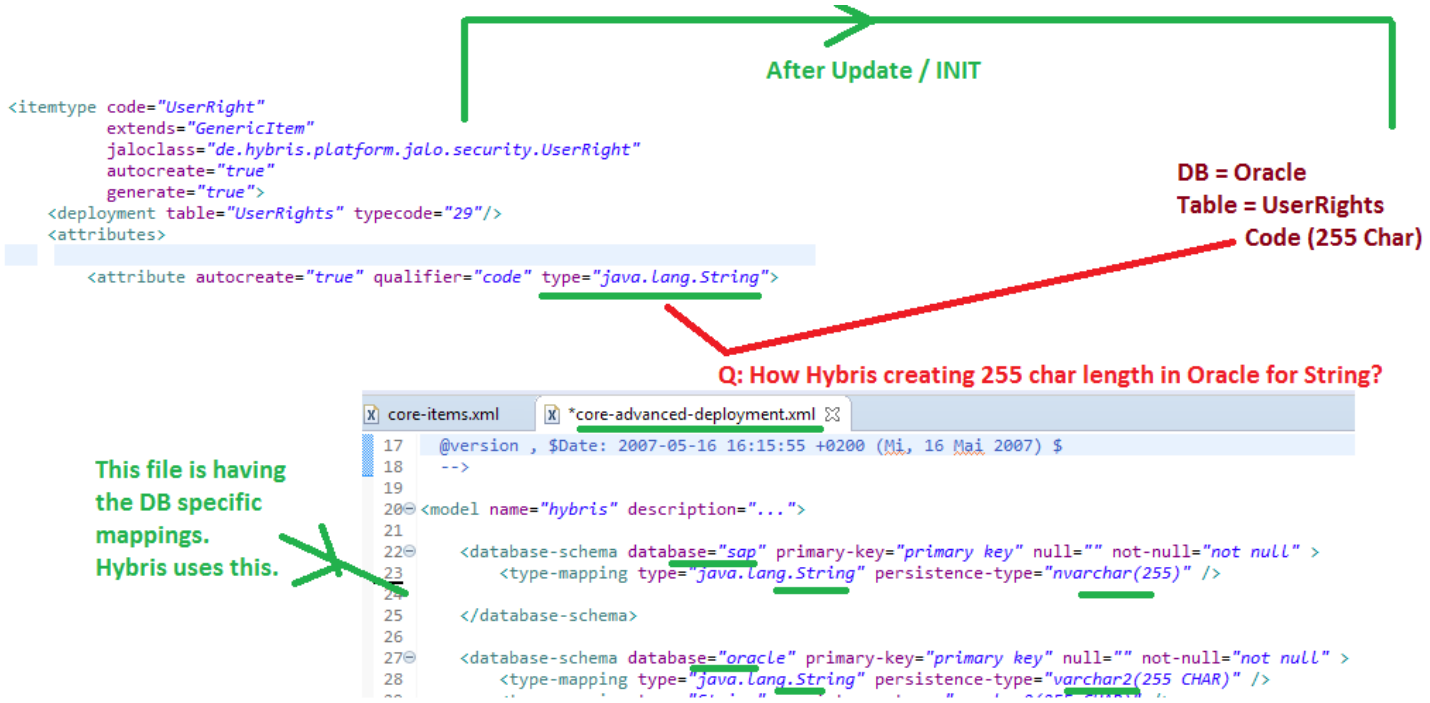


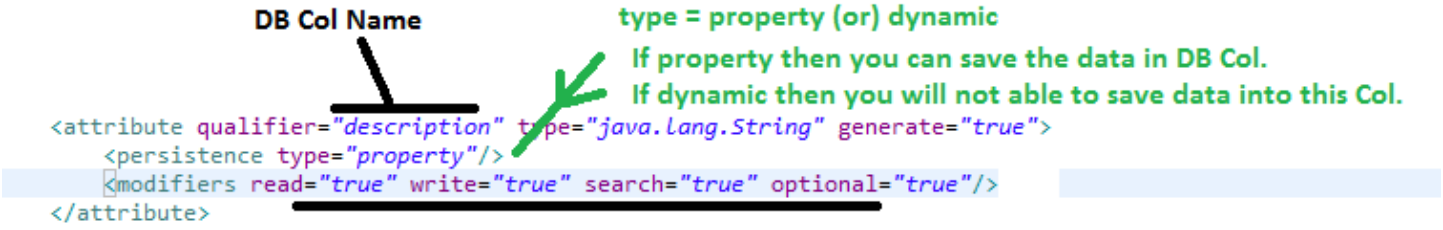
<custom-properties> = If you want to search on particular columns in hMC then put that column in custom properties. For PK columns, this will be created by default.

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**





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">

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

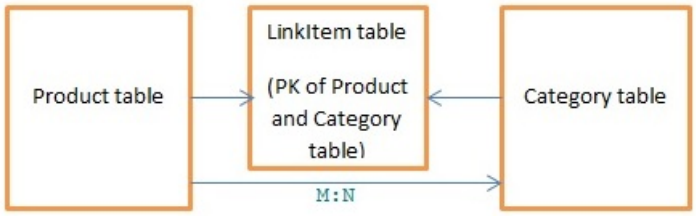
CategoryModel.java

```
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 =

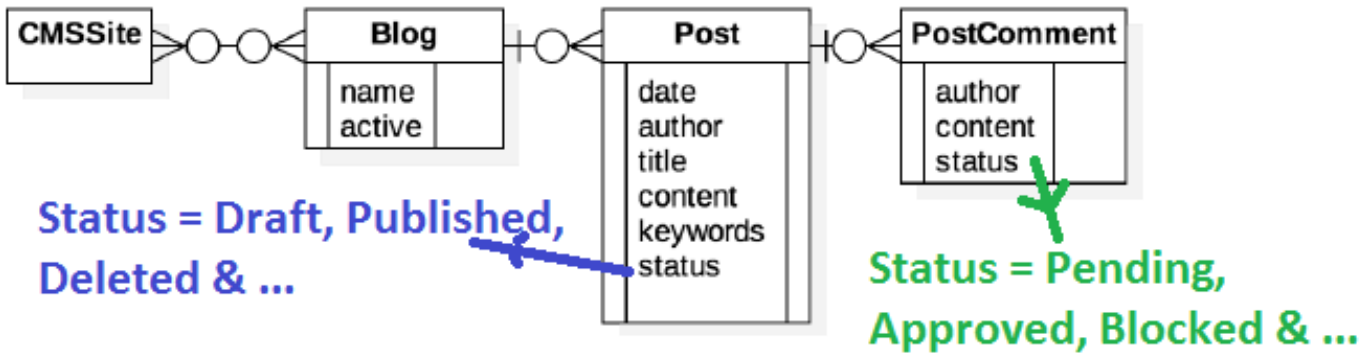
We can change the modifier from **read=true** to **read=false**

We can make attribute as **unique**.

We can add **write=false**.

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

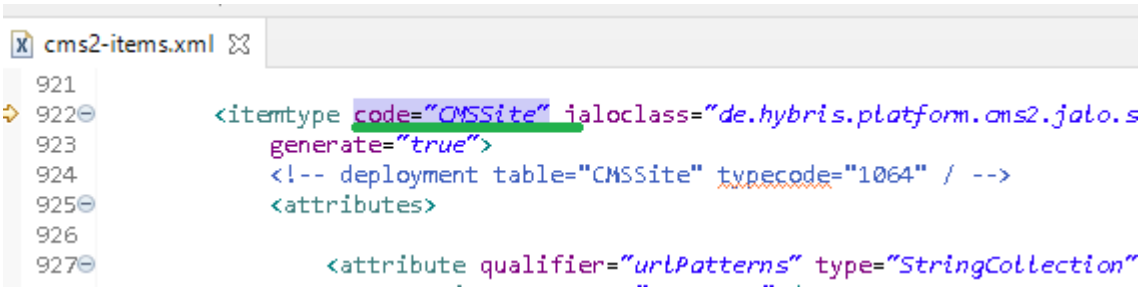
Business Scenario



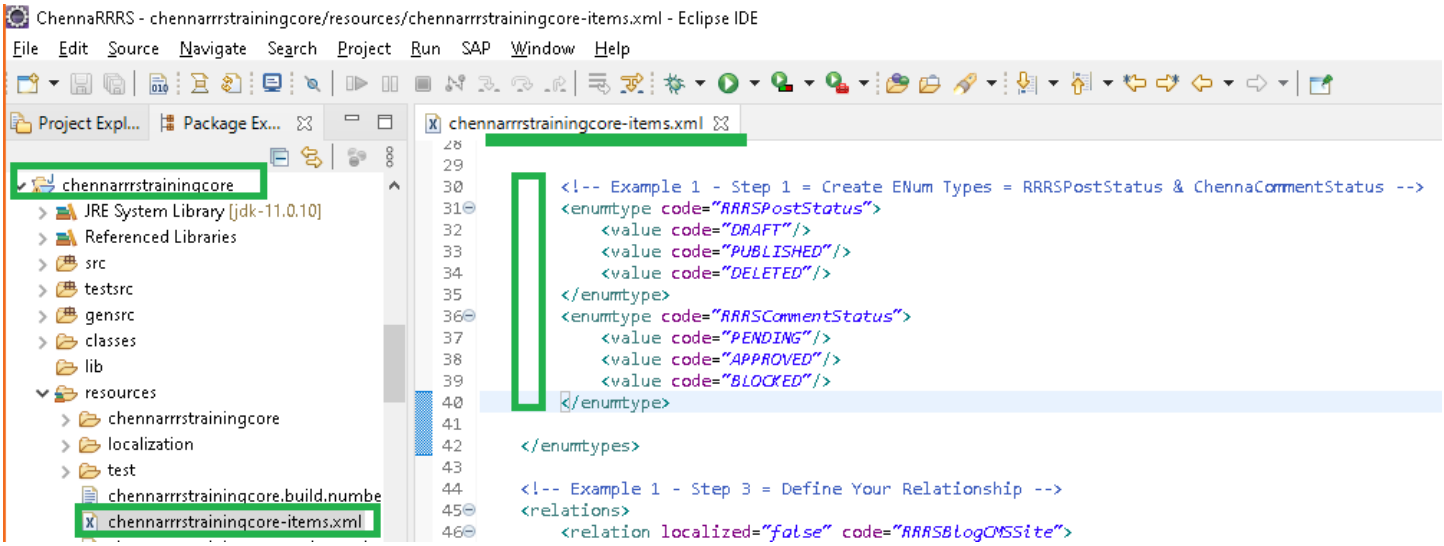
Solution =

Step 1 = Create CMSSite

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



Step 2 = Create 2 Enum Types



Step 3 = Create “Blog … Post … PostComment” tables

*chennarrrstrainingcore-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>
```

chennarrrstrainingcore-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>
```

Don't type FYI...

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

chennarrrstrainingcore-items.xml

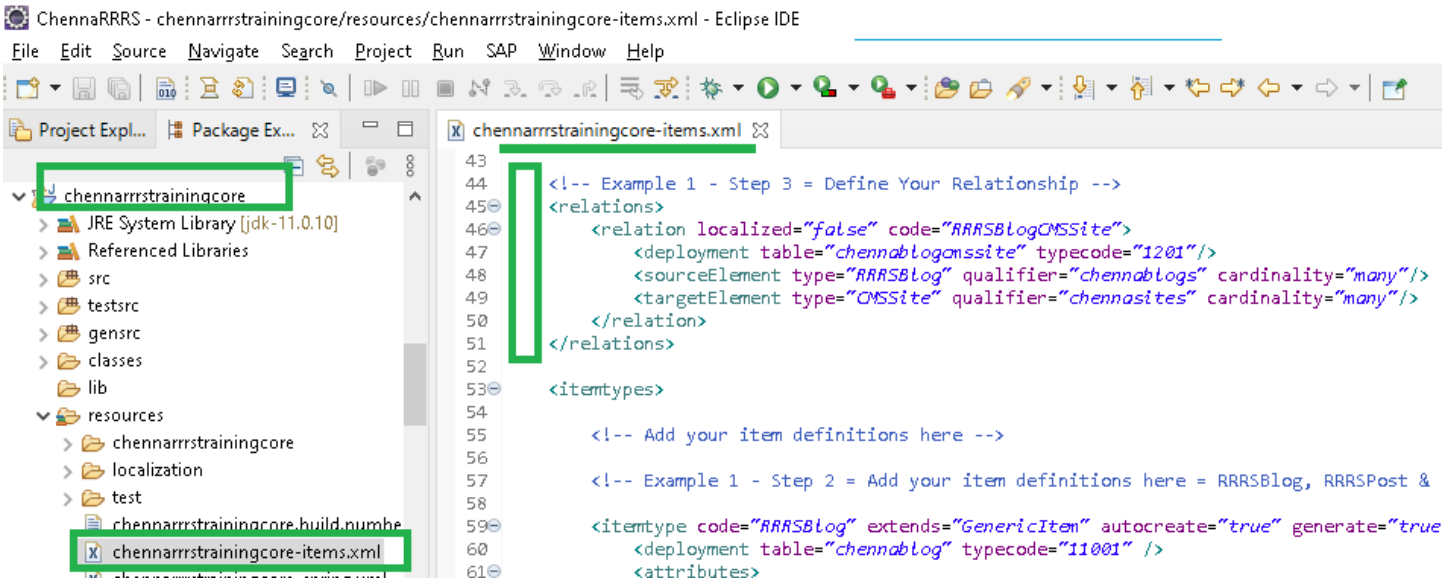
```
<itemtype code="RRRSPostComment" extends="GenericItem" autocreate="true" generate="true">
    <deployment table="rrrspostcomment" typecode="11003" />
    <attributes>
        <attribute qualifier="author" type="User">
            <persistence type="property" />
        </attribute>
        <attribute 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>
```

core-items.xml

FYI... Don't Type.

```
1588    <itemtype code="User"
1589              extends="Principal"
1590              jaloclass="de.hybris.platform.jalo.user.User"
1591              autocreate="true"
1592              generate="true">
1593        <deployment table="Users" typecode="4" propertytable="UserProps"/>
1594        <attributes>
```

Step 4 = Create Relation



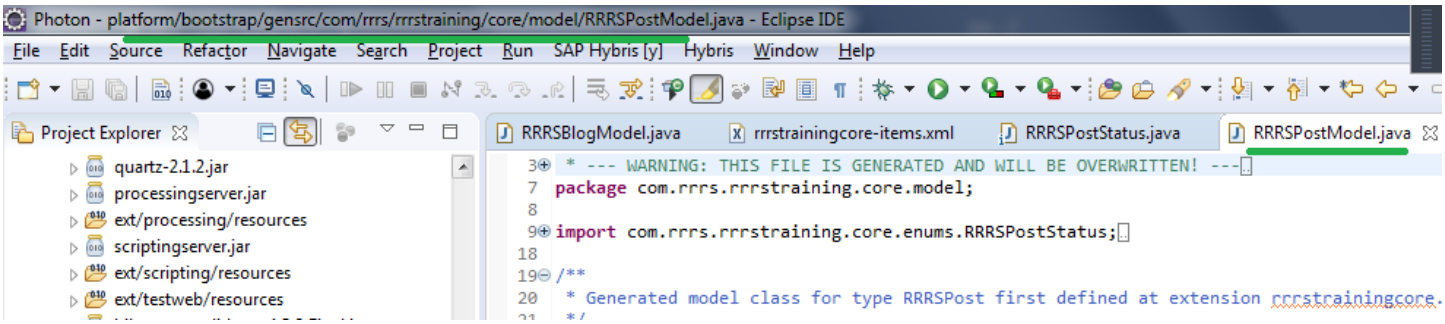
Step 5 = Do the build [ant clean all]

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.**



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.

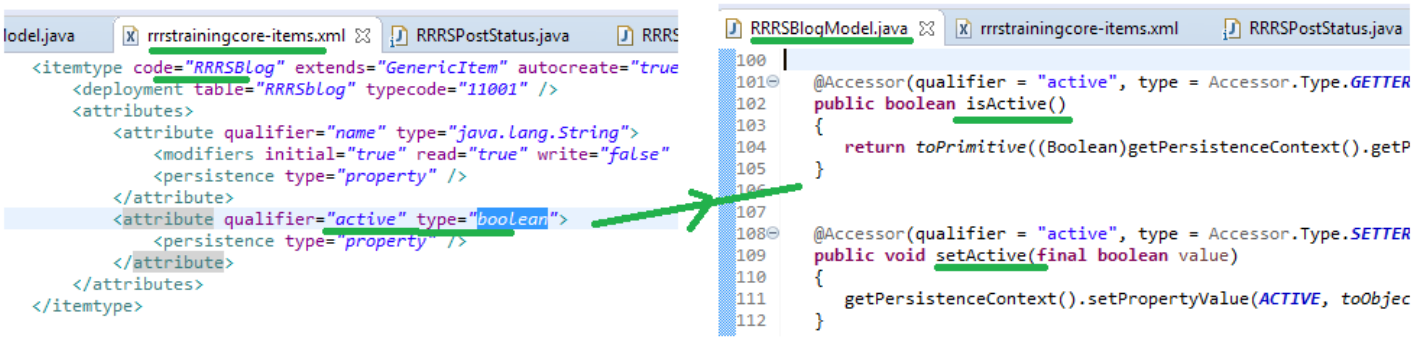
3 = If itemtype is having n attributes, then what model class will have: -

a. For each attribute there will be **1 variable**

b. For each attribute there will be **1 setter & 1 getter**

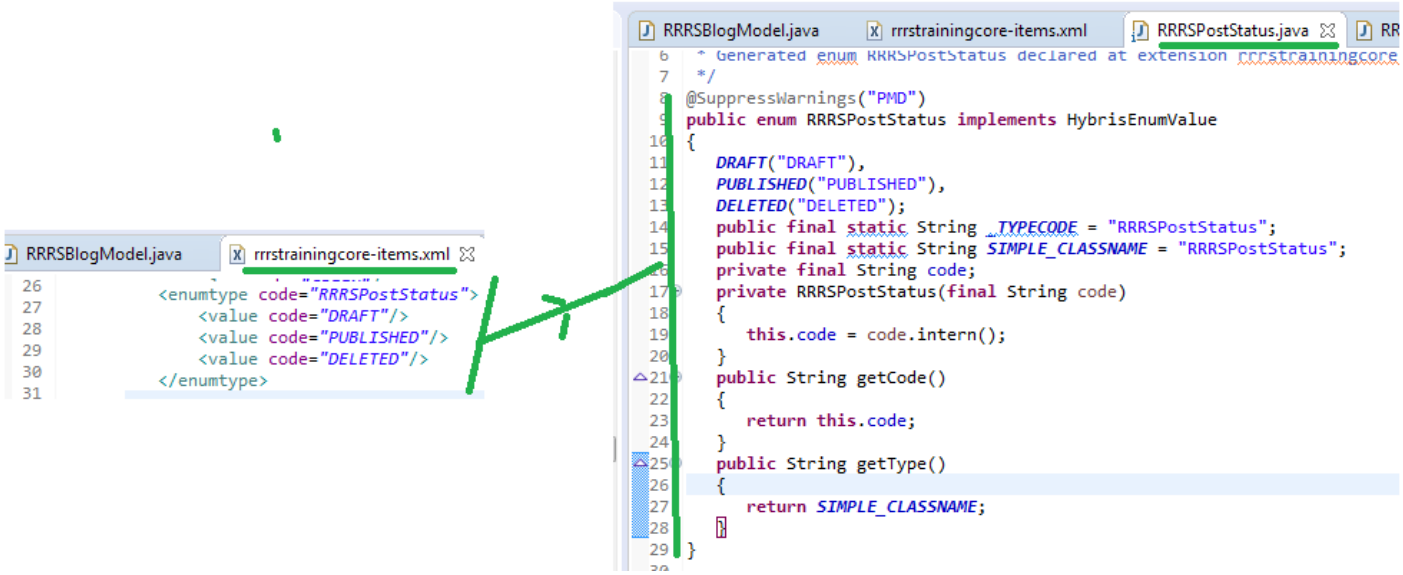
c. If attribute is **localized** – there will be **2 setters & 2 getter** method

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



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**



Step 6 = Start the Hybris Server

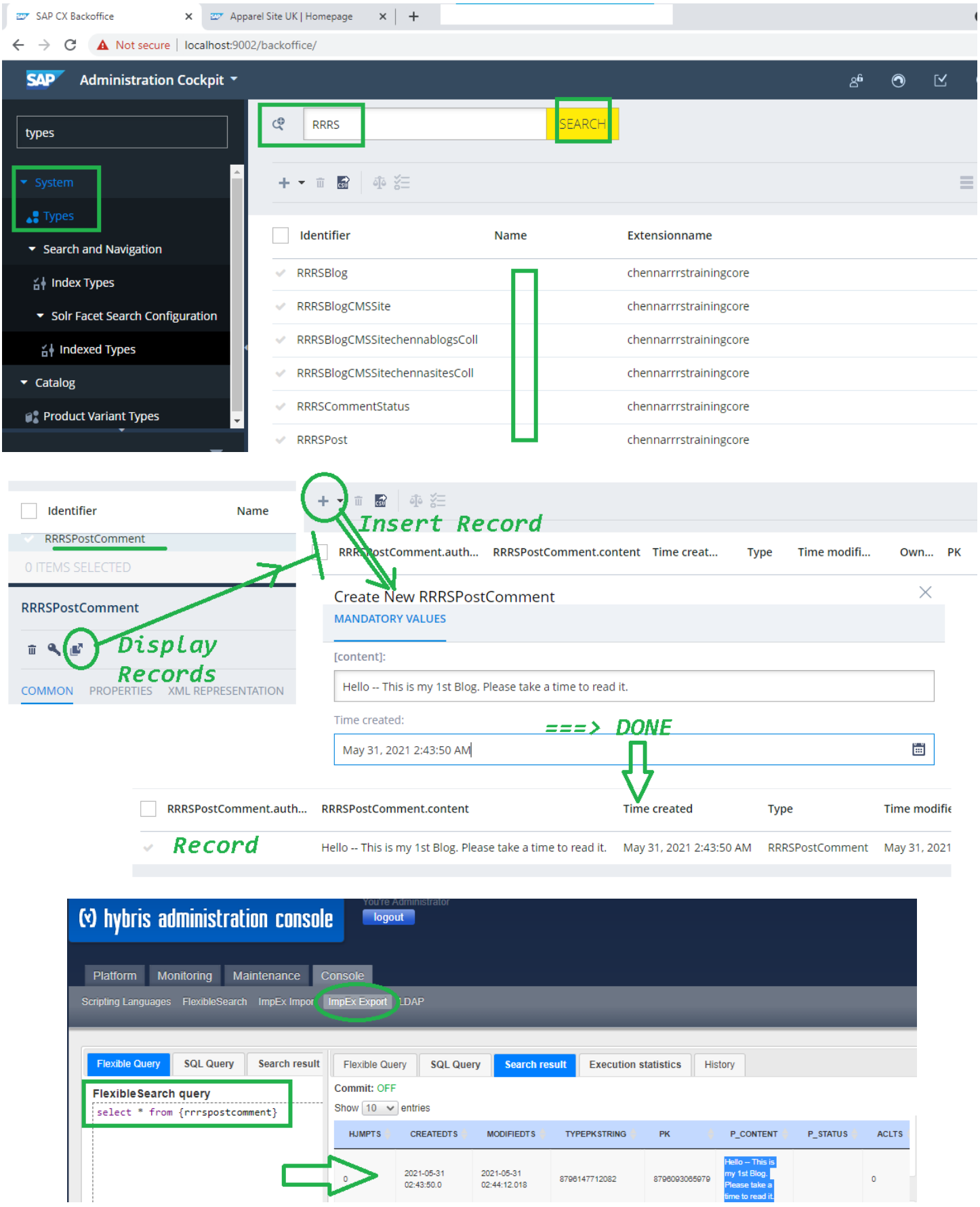
E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

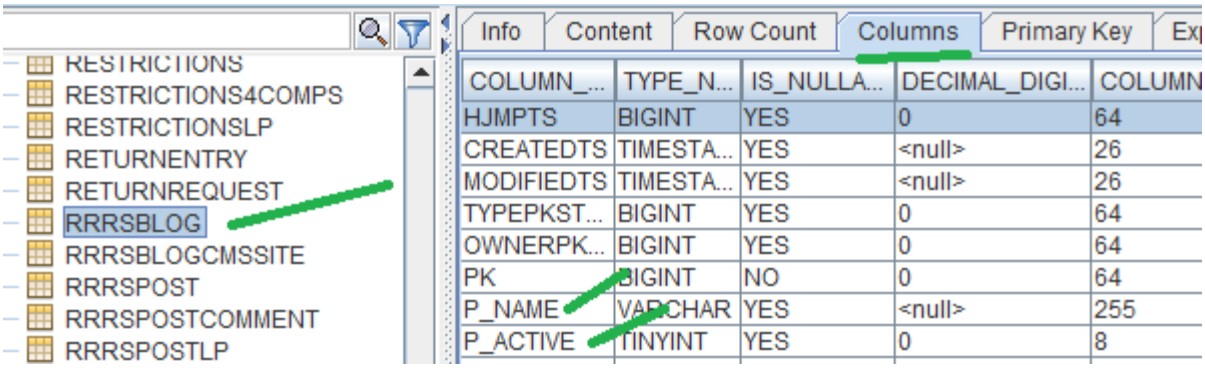
Step 7 = Perform Platform Update

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

Platform ➔ Update

Step 8 = Results

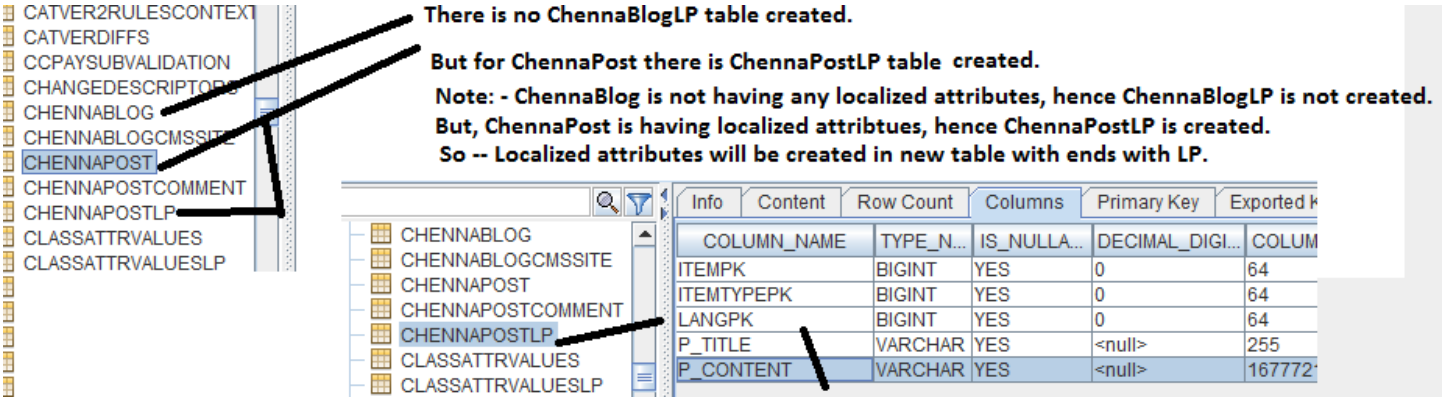


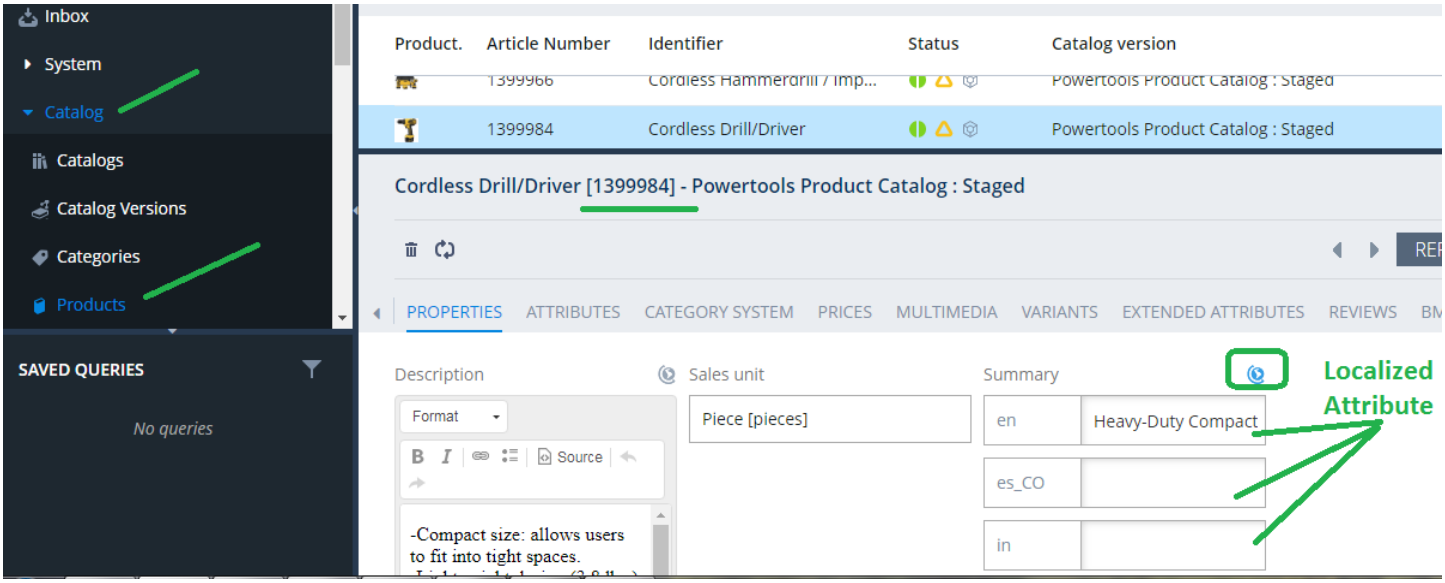


Note = If you observe above, there is no "**RRRSBlogLP**", but there is "**RRRSPOSTLP**".

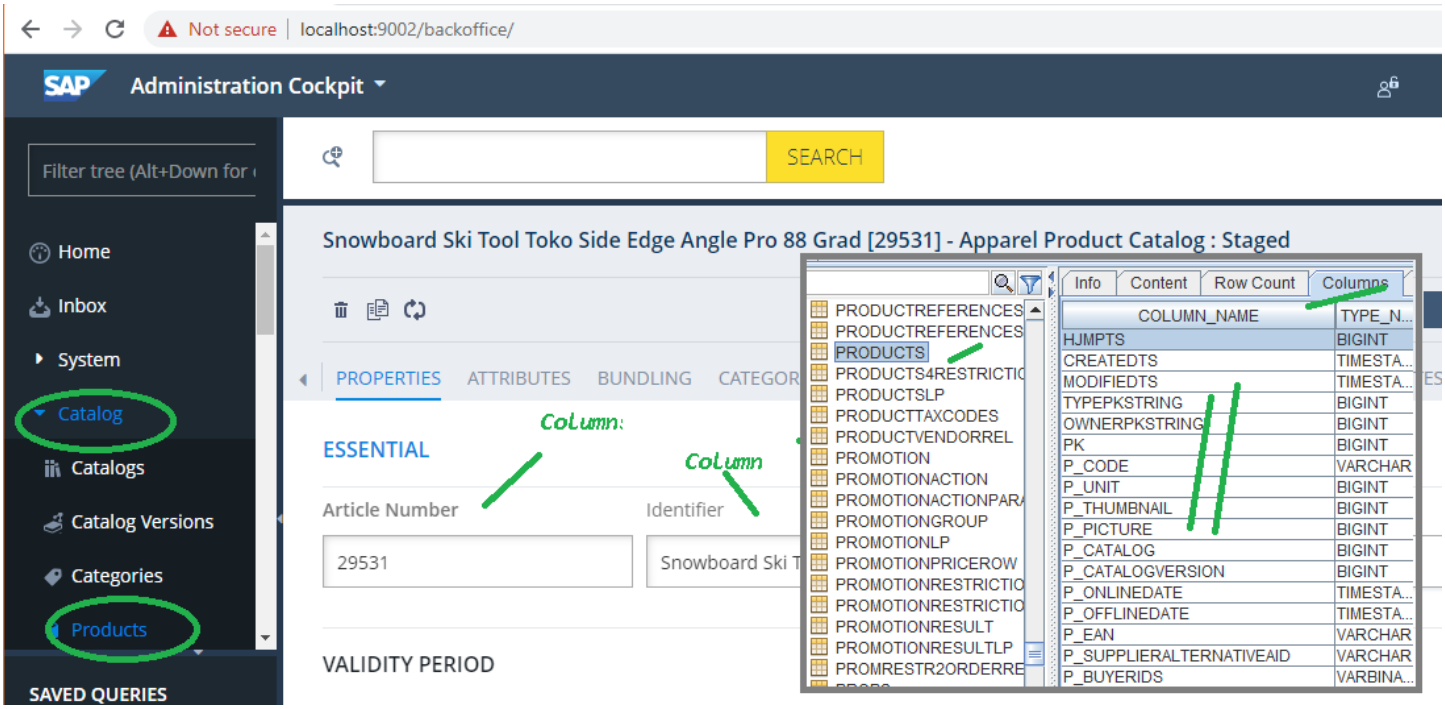
RRRSPOSTLP is created because, in "**RRRSPOST**" is having localized attributes.

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





Scenario = Extend existing itemtype



"SAP Comm" already providing the **Product** table.

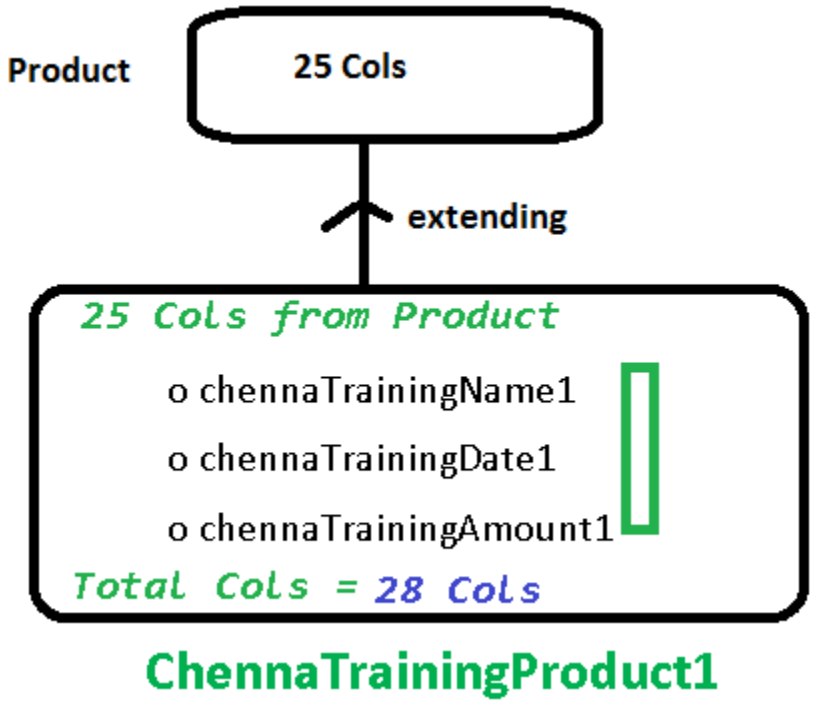
Assume that "SAP Comm" providing table is having **25 Cols**. Do you think that, this 25 Cols are sufficient for your company = May (or) May not.

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

Requirment = We want to create **new itemtype** called "**ChennaTrainingProduct1**" by extending "**Product**" and

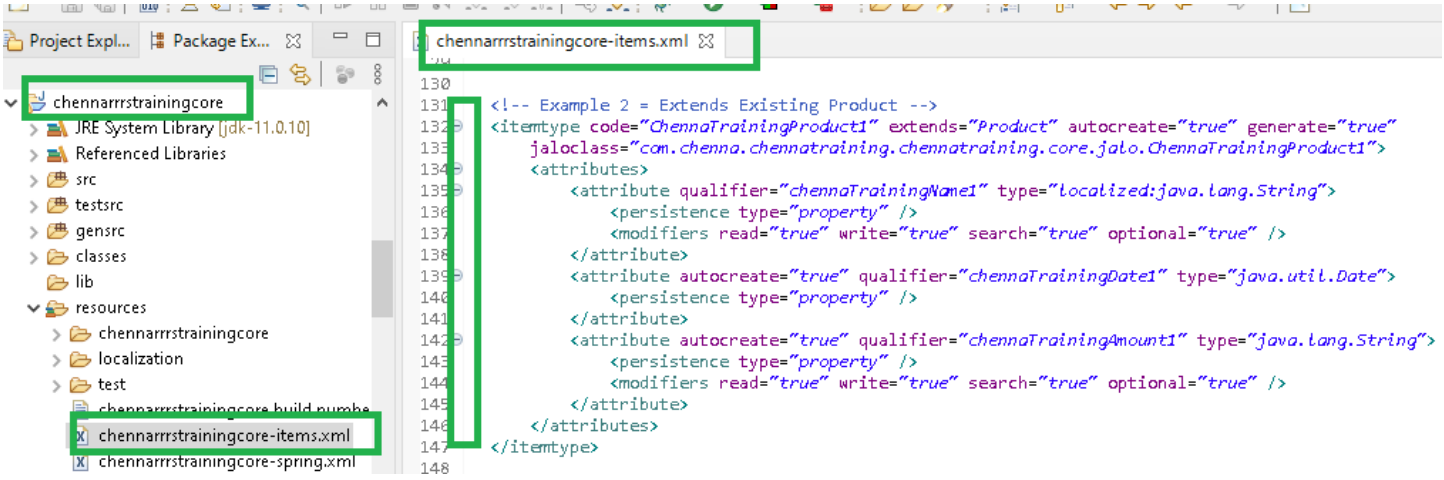
in new itemtype add **another 3 cols**: -

- o chennaTrainingName1
- o chennaTrainingDate1
- o chennaTrainingAmount1



Solution =

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

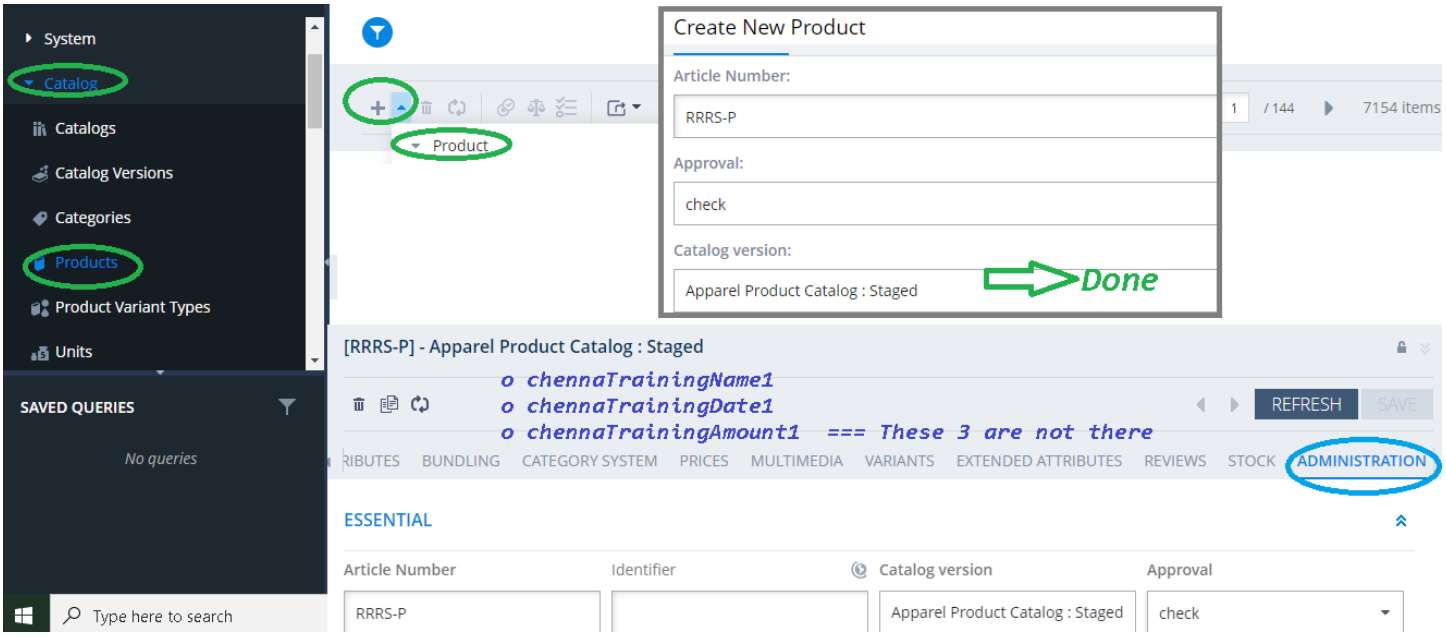


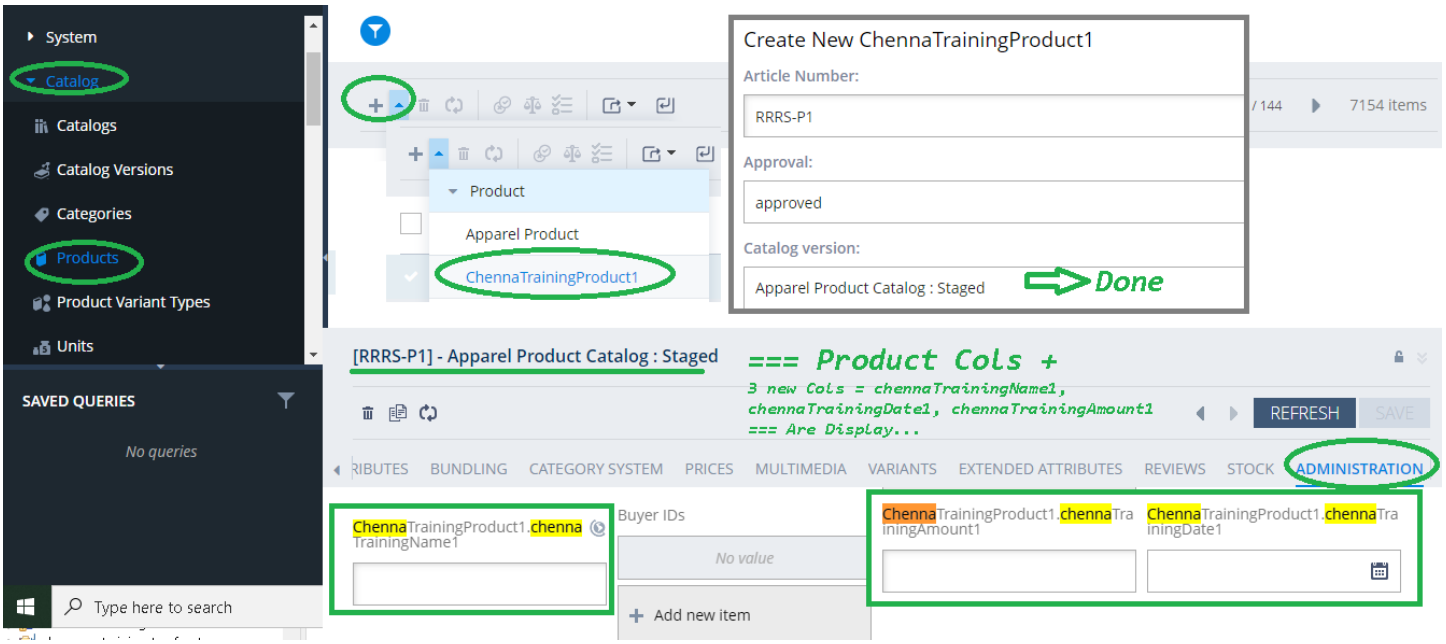
Step 2 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

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

Step 4 = Perfrom "Platform – Update" [→ hAC → Platform → Update]

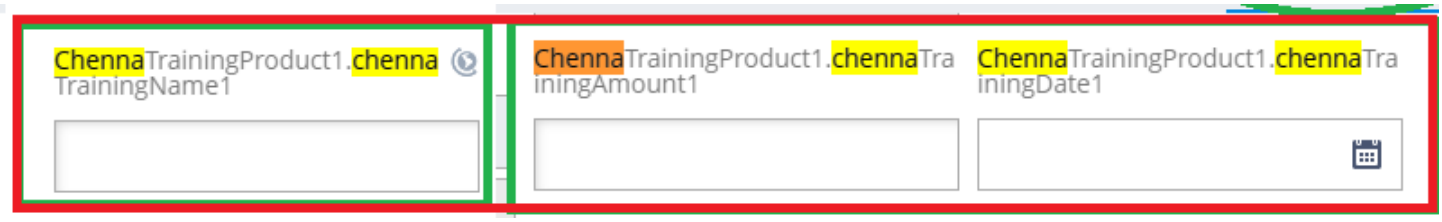
Step 5 = Results



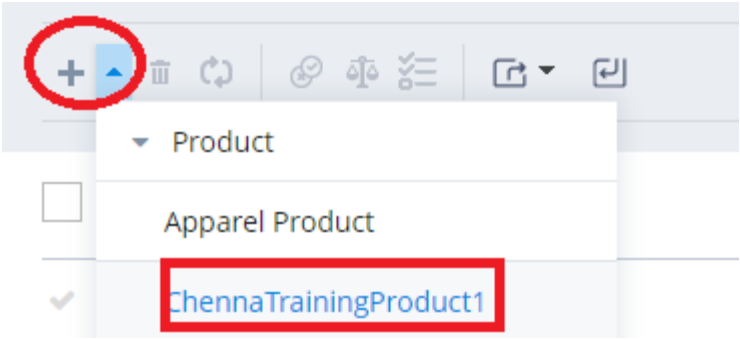


Challenges in above Scenario =

1 = Coulumn Label Names are not proper



2 = Itemtype Lable names are not proper



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

Requirement =

Columns =

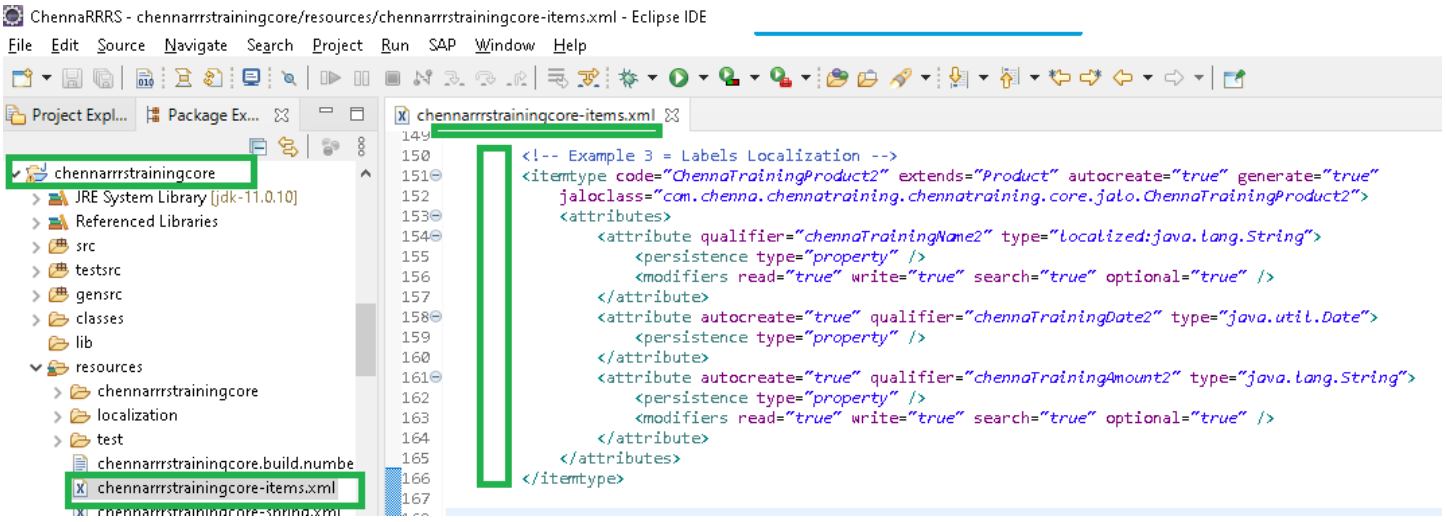
- chennaTrainingName2 = Chenna Training Name 2
- chennaTrainingDate2 = Chenna Training Date 2
- chennaTrainingAmount2 = Chenna Training Amount 2

Itemtype

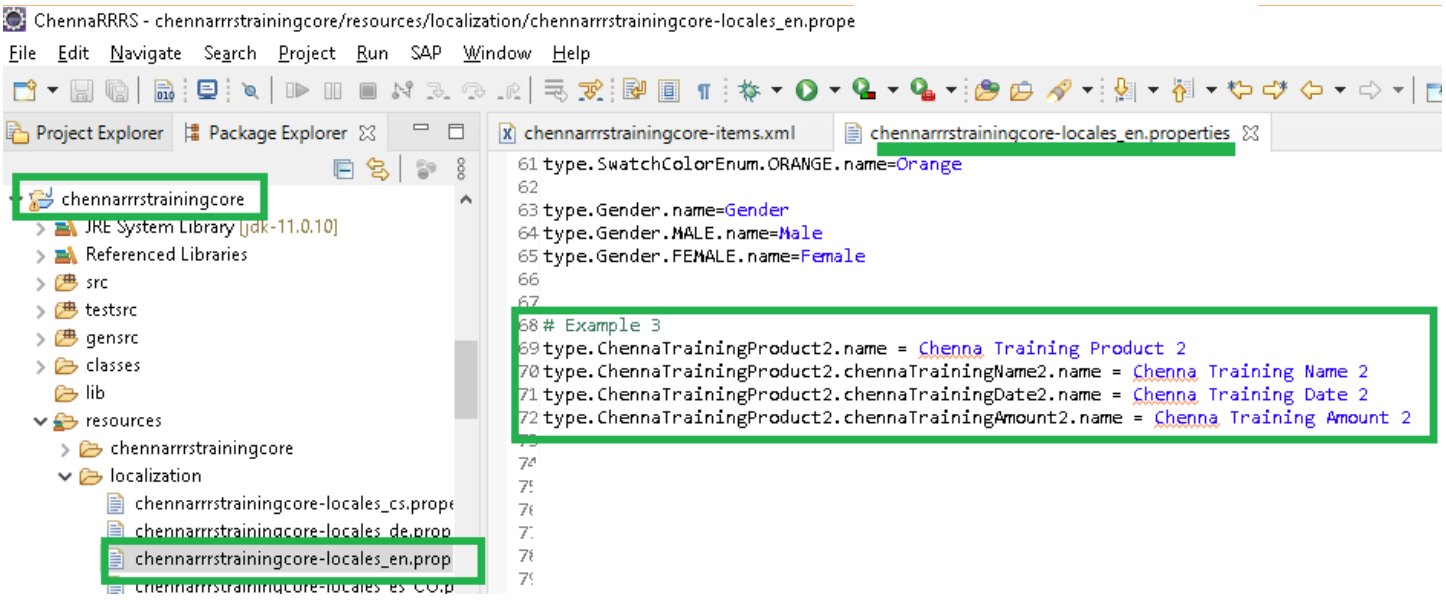
ChennaTrainingProduct2 = Chenna Training Product 2

Solution =

Step 1 = Create new itemtype called "**ChennaTrainingProduct2**" by exteding existing type called **Product** – and 3 new Cols.



Step 2 = Give proper display lable names

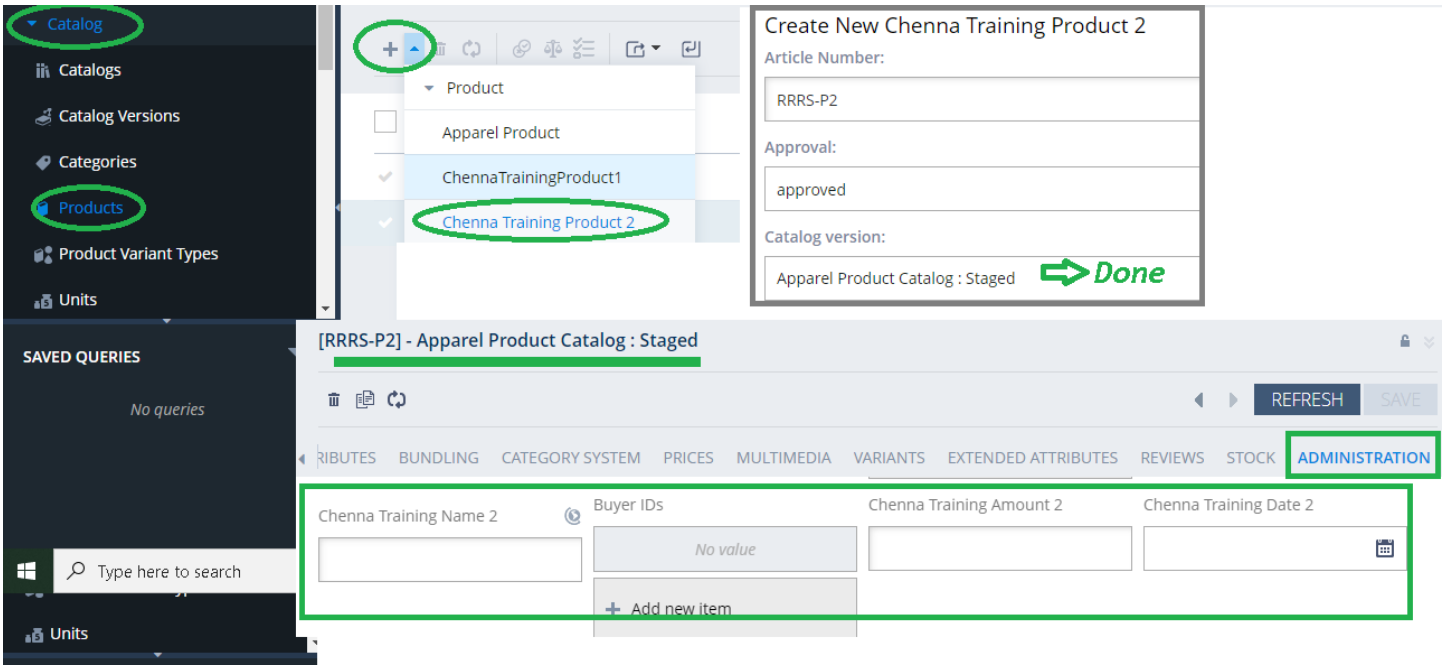


Step 3 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

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

Step 5 = Perfrom "Platform – Update" [→ hAC → Platform → Update]

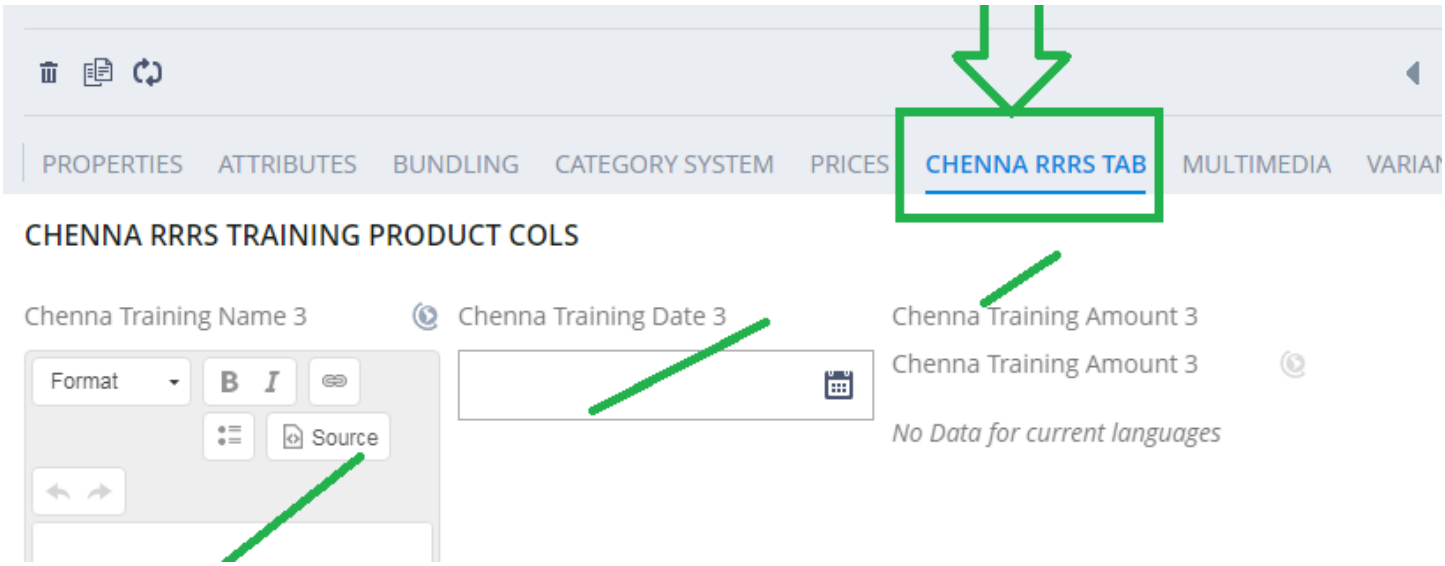
Step 6 = Results



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 =



Step 1 = Create new itemtype called "**ChennaTrainingProduct3**" by exteding existing type called **Product** – and 3 new Cols.

chennarrrstrainingcore-items.xml

```
<!-- Example 4 = New Tab & Label Localization -->
<itemtype code="ChennaTrainingProduct3" extends="Product" autocreate="true" generate="true"
    jaloclass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct3">
    <attributes>
        <attribute qualifier="chennaTrainingName3" type="localized:java.lang.String">
            <persistence type="property" />
            <modifiers read="true" write="true" search="true" optional="true" />
        </attribute>
        <attribute autocreate="true" qualifier="chennaTrainingDate3" type="java.util.Date">
            <persistence type="property" />
        </attribute>
        <attribute autocreate="true" qualifier="chennaTrainingAmount3" type="java.lang.String">
            <persistence type="property" />
            <modifiers read="true" write="true" search="true" optional="true" />
        </attribute>
    </attributes>
</itemtype>
```

Step 2 = Give proper display lable names

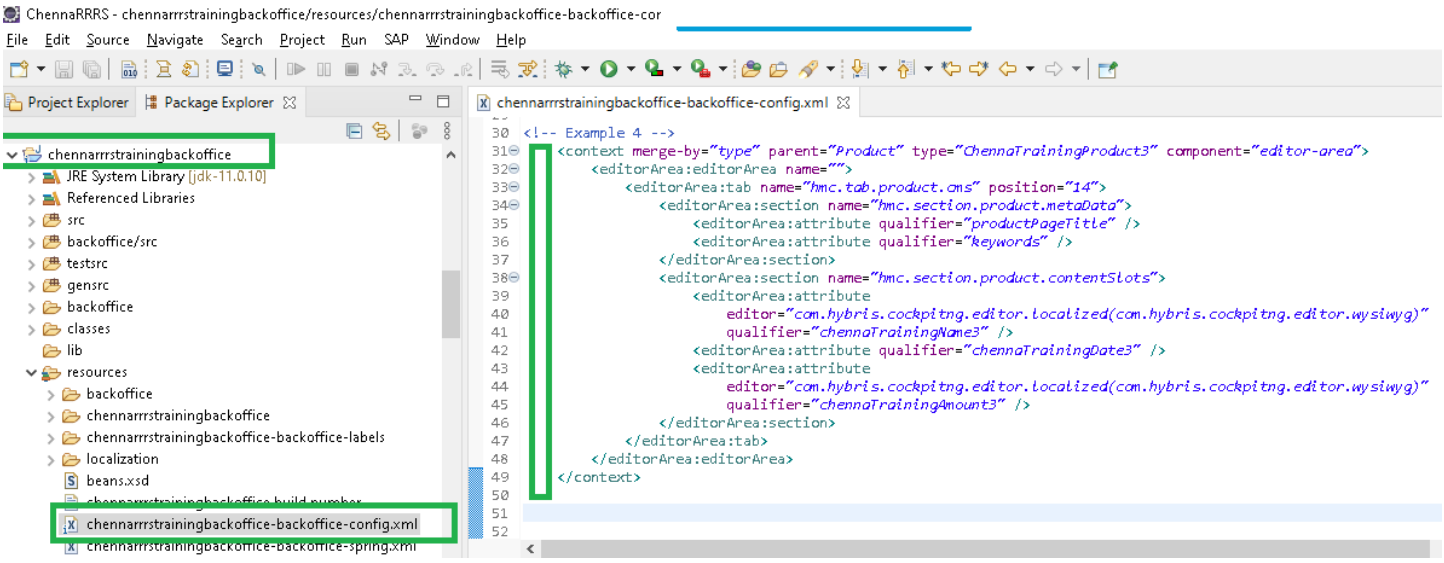
chennarrrstrainingcore-items.xml | chennarrrstrainingcore-locales_en.properties

```
# Example 4
type.ChennaTrainingProduct3.name = Chenna Training Product 3
type.ChennaTrainingProduct3.chennaTrainingName3.name = Chenna Training Name 3
type.ChennaTrainingProduct3.chennaTrainingDate3.name = Chenna Training Date 3
type.ChennaTrainingProduct3.chennaTrainingAmount3.name = Chenna Training Amount 3
```

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

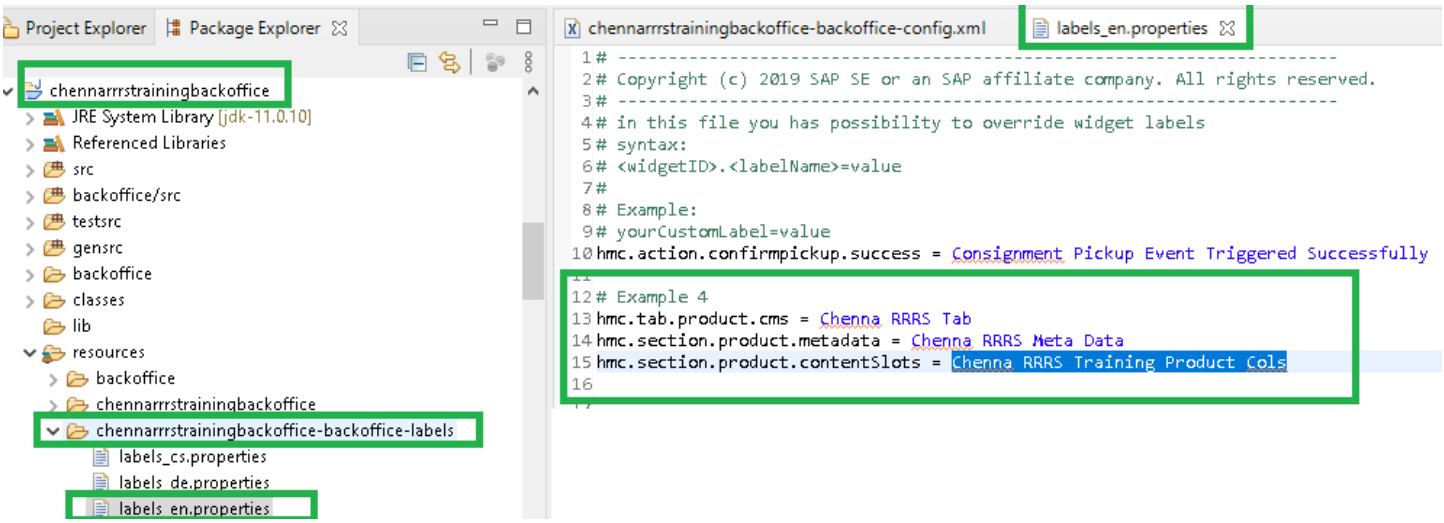
chennarrrstrainingbackoffice-backoffice-config.xml

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



Step 4 = Give Proper Display Names for Backoffice UI Elements

Example = Tab name, Section name and …

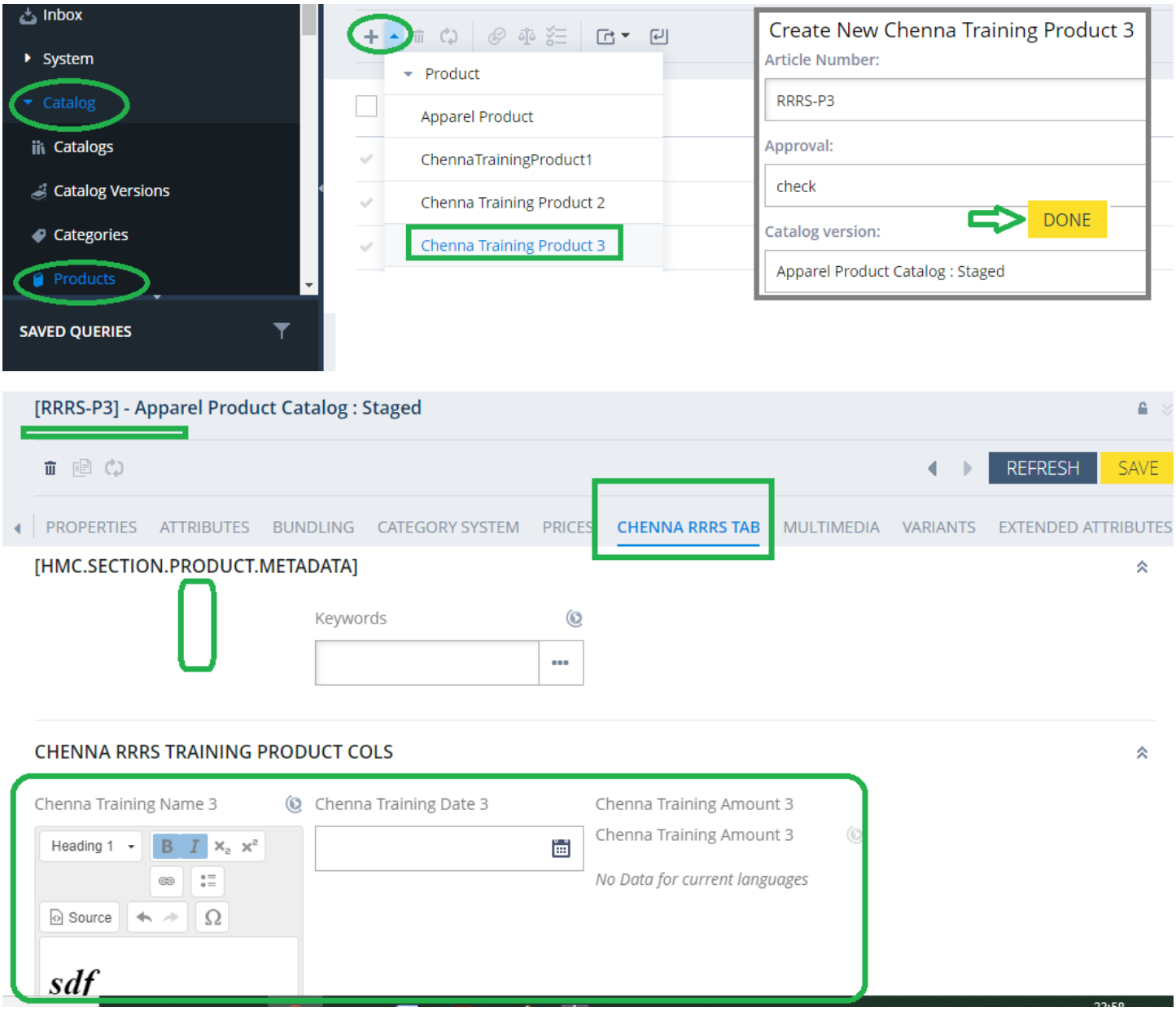


Step 5 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

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

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

Step 8 = Results



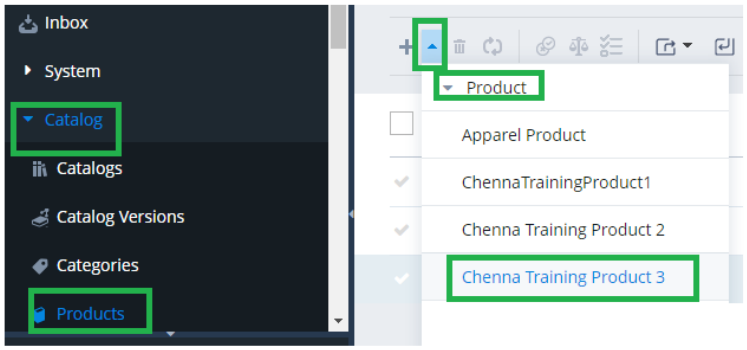
Challenges in above Scenario =

To create product – How many clicks required?

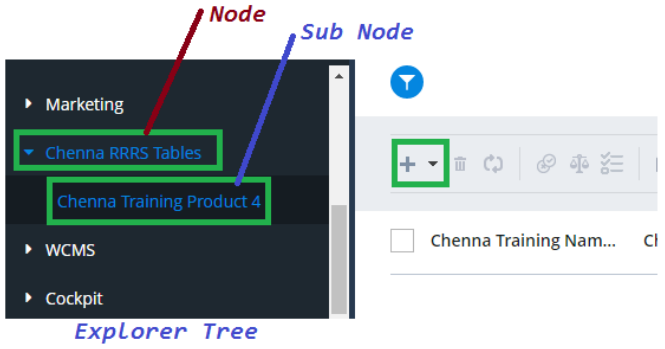
-- Catalog – Products – [+] -- Product --- Chenna RRRS Product 3

=== Total Clicks = 5

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

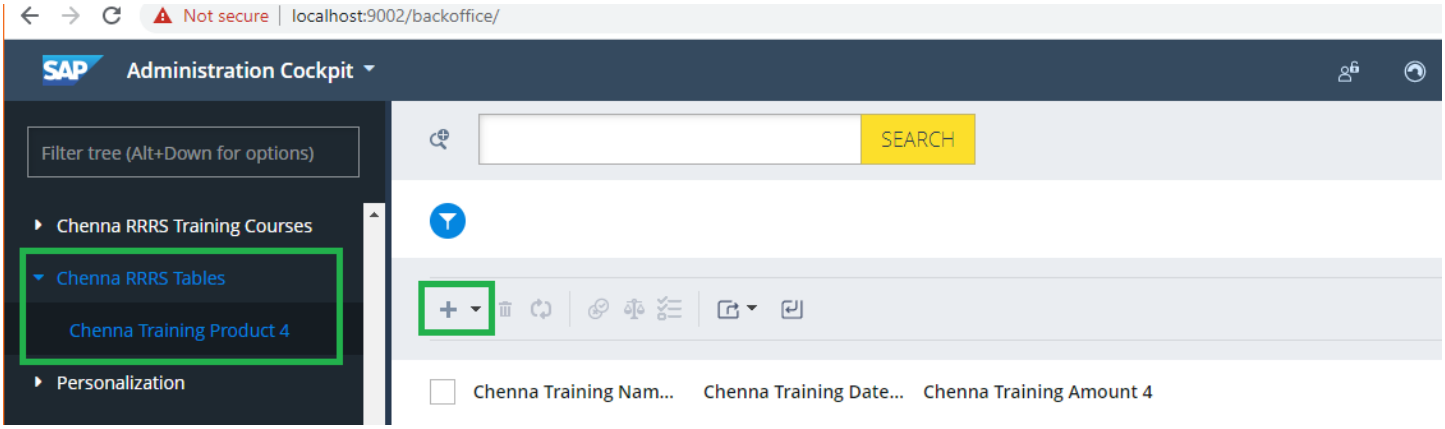


This is Results from **Requirement 3**



This is Results Customer want.

Requirement = You want something like below =



Step 1 = Create new itemtype called "**ChennaTrainingProduct4**" by exteding existing type called **Product** – and 3 new Cols.

chennarrrstrainingcore-items.xml

```
<!-- Example 4 = New Tab & Label Localization -->
<itemtype code="ChennaTrainingProduct3" extends="Product" autocreate="true" generate="true"
    jaloclass="com.chenna.chennatraining.chennatraining.core.jalo.ChennaTrainingProduct3">
    <attributes>
        <attribute qualifier="chennaTrainingName3" type="localized:java.lang.String">
            <persistence type="property" />
            <modifiers read="true" write="true" search="true" optional="true" />
        </attribute>
        <attribute autocreate="true" qualifier="chennaTrainingDate3" type="java.util.Date">
            <persistence type="property" />
        </attribute>
        <attribute autocreate="true" qualifier="chennaTrainingAmount3" type="java.lang.String">
            <persistence type="property" />
            <modifiers read="true" write="true" search="true" optional="true" />
        </attribute>
    </attributes>
</itemtype>
```

Step 2 = Give proper display lable names

chennarrrstrainingcore-items.xml | chennarrrstrainingcore-locales_en.properties

```
# Example 5
type.ChennaTrainingProduct4.name = Chenna Training Product 4
type.ChennaTrainingProduct4.chennaTrainingName4.name = Chenna Training Name 4
type.ChennaTrainingProduct4.chennaTrainingDate4.name = Chenna Training Date 4
type.ChennaTrainingProduct4.chennaTrainingAmount4.name = Chenna Training Amount 4
```

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

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

chennarrrstrainingbackoffice-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 …

chennarrrstrainingbackoffice-backoffice-config.xml | labels_en.properties

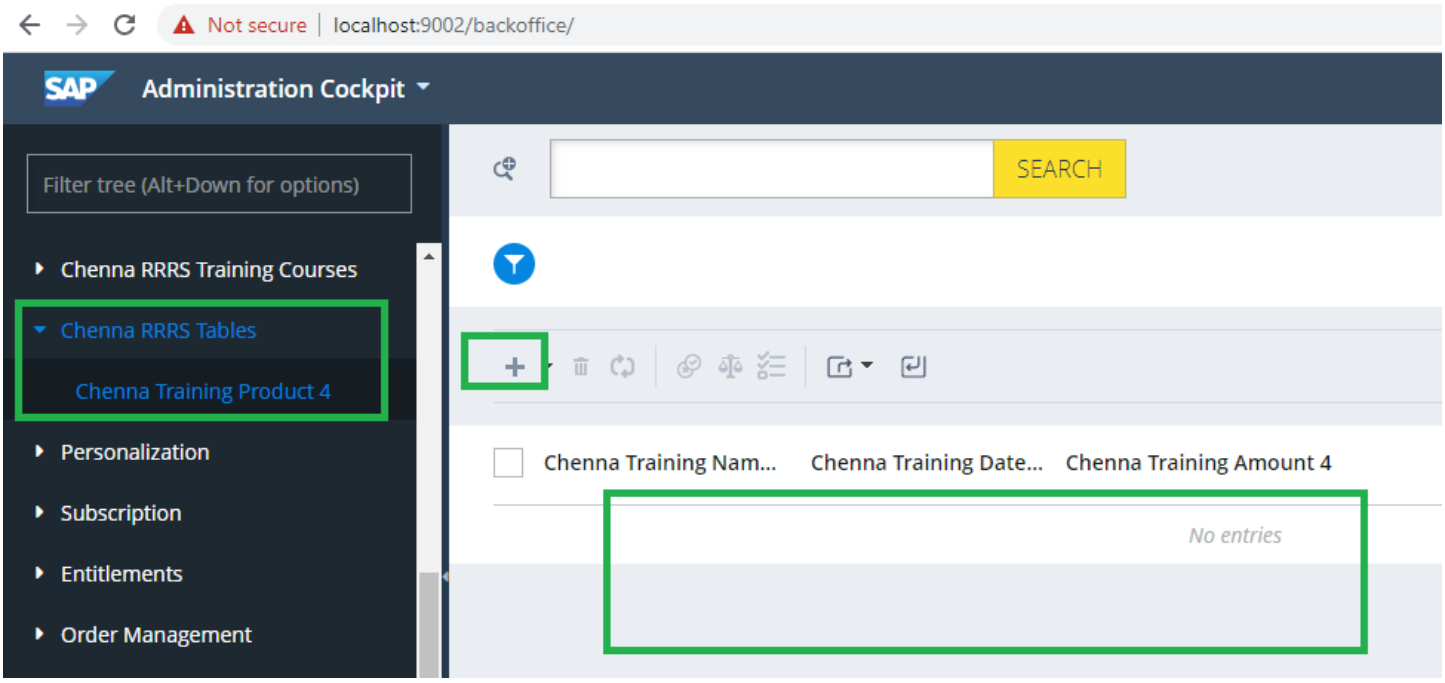
```
# Example 5
hmc_treenode_chennarrrs = Chenna RRRS Tables
```

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

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

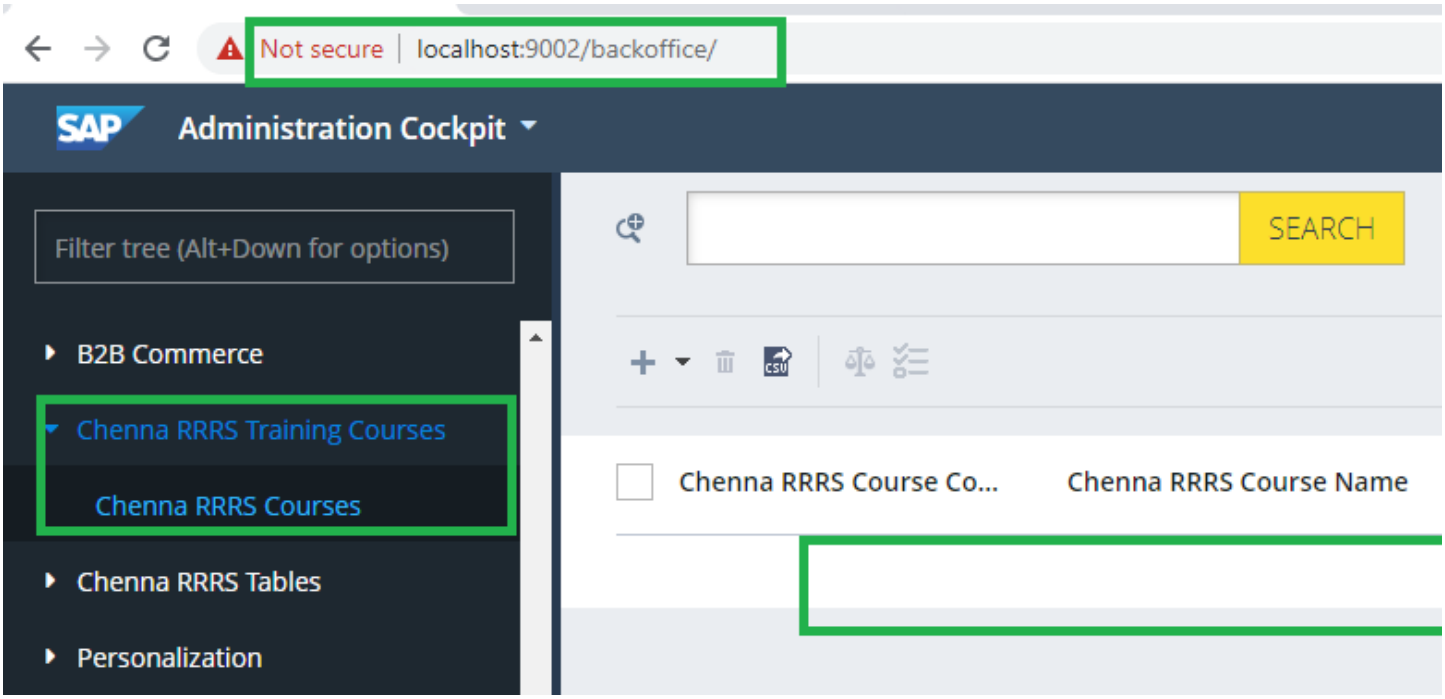
Step 7 = Perfrom “Platform – Update” [→ hAC → Platform → Update]

Step 8 = Results

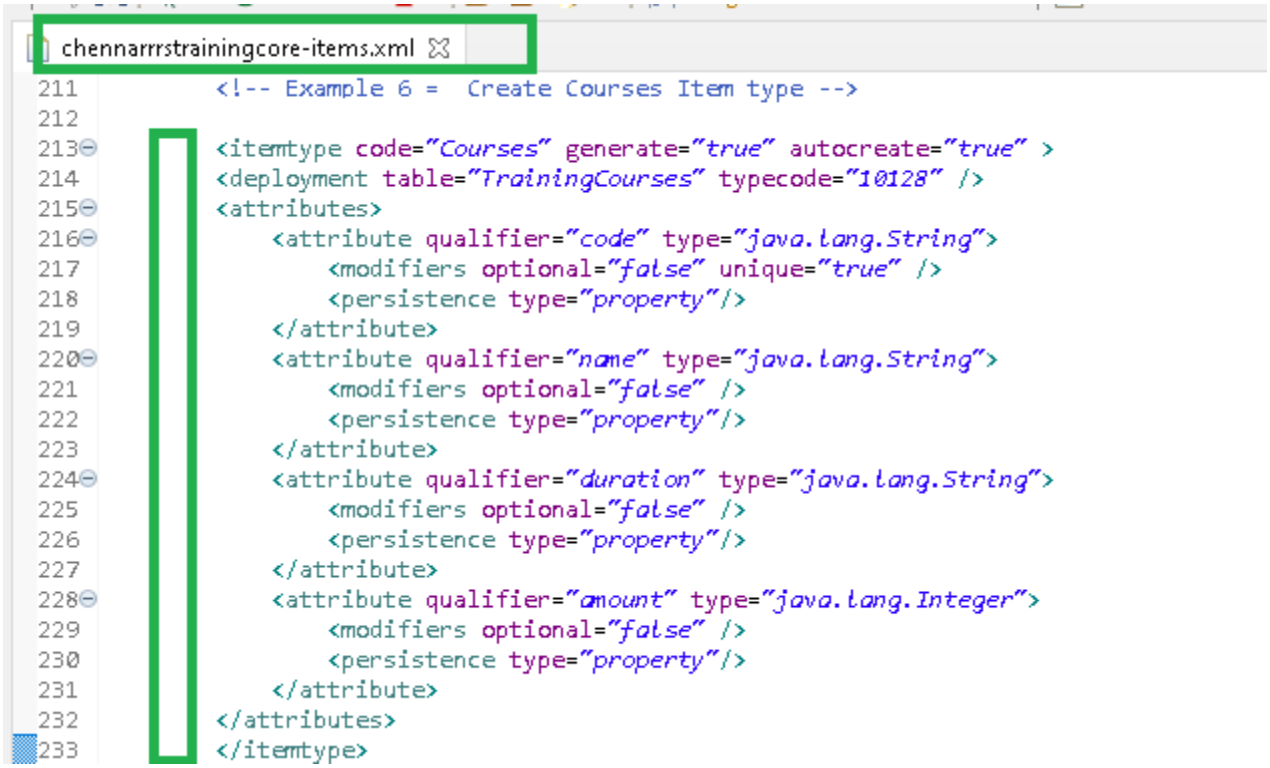


Scenario = Create itemtype called Courses with 4 Cols [Code, Name, Duration & Amount].

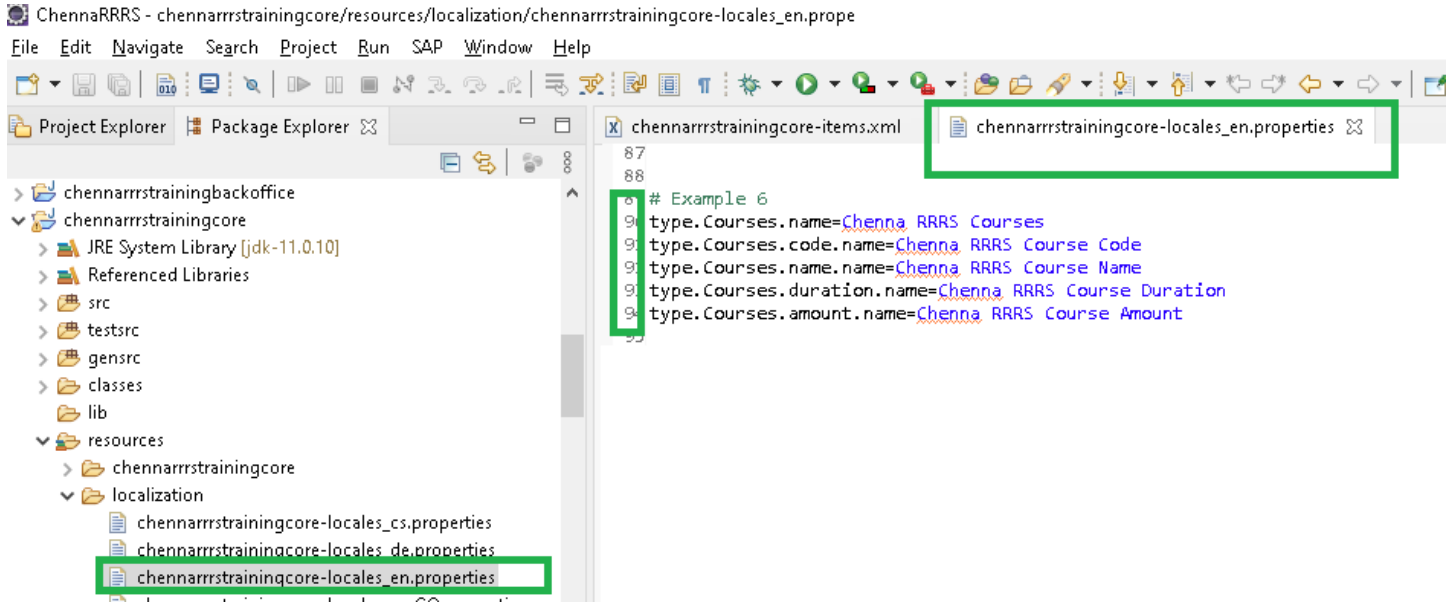
Diplay Courses in Backoffice Left Navigation [Explorer Tree].



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

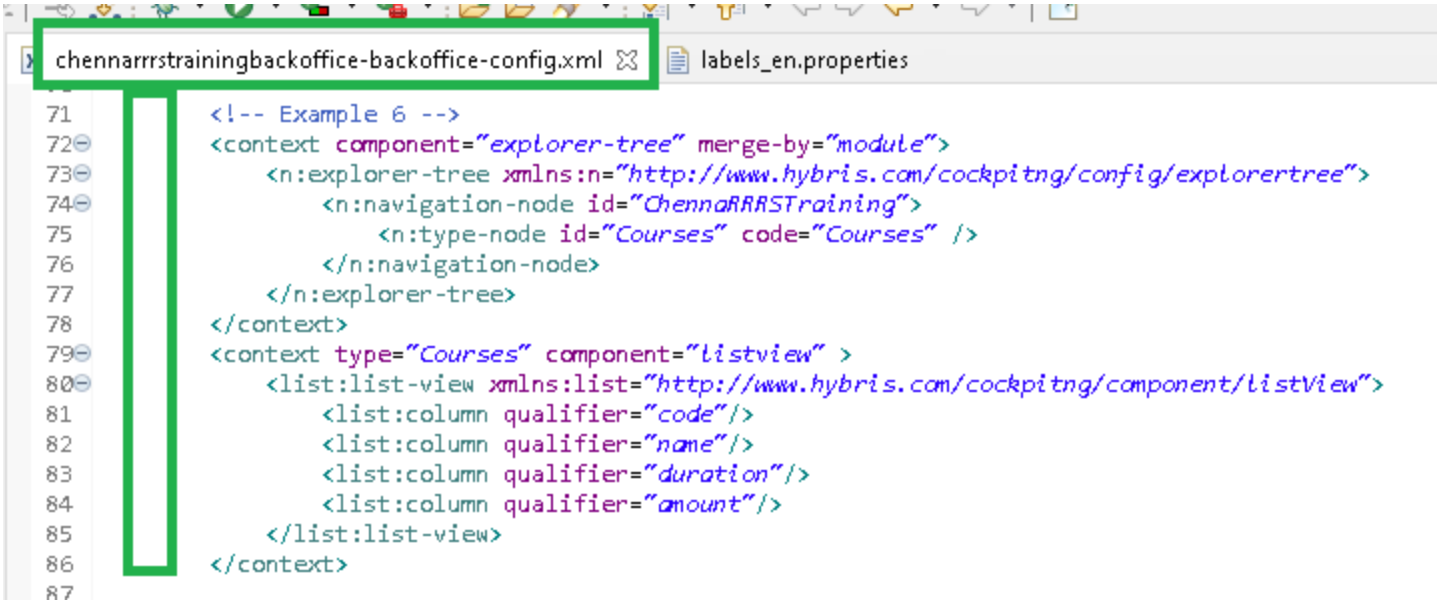


Step 2 = Give proper display lable names



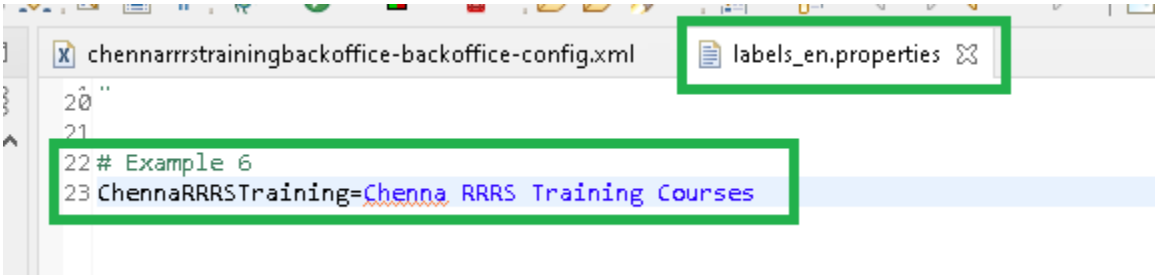
Step 3 = Display **Courses** itemtype in **explorer tree**.

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



Step 4 = Give Proper Display Names for Backoffice UI Elements

Example = Node name and …

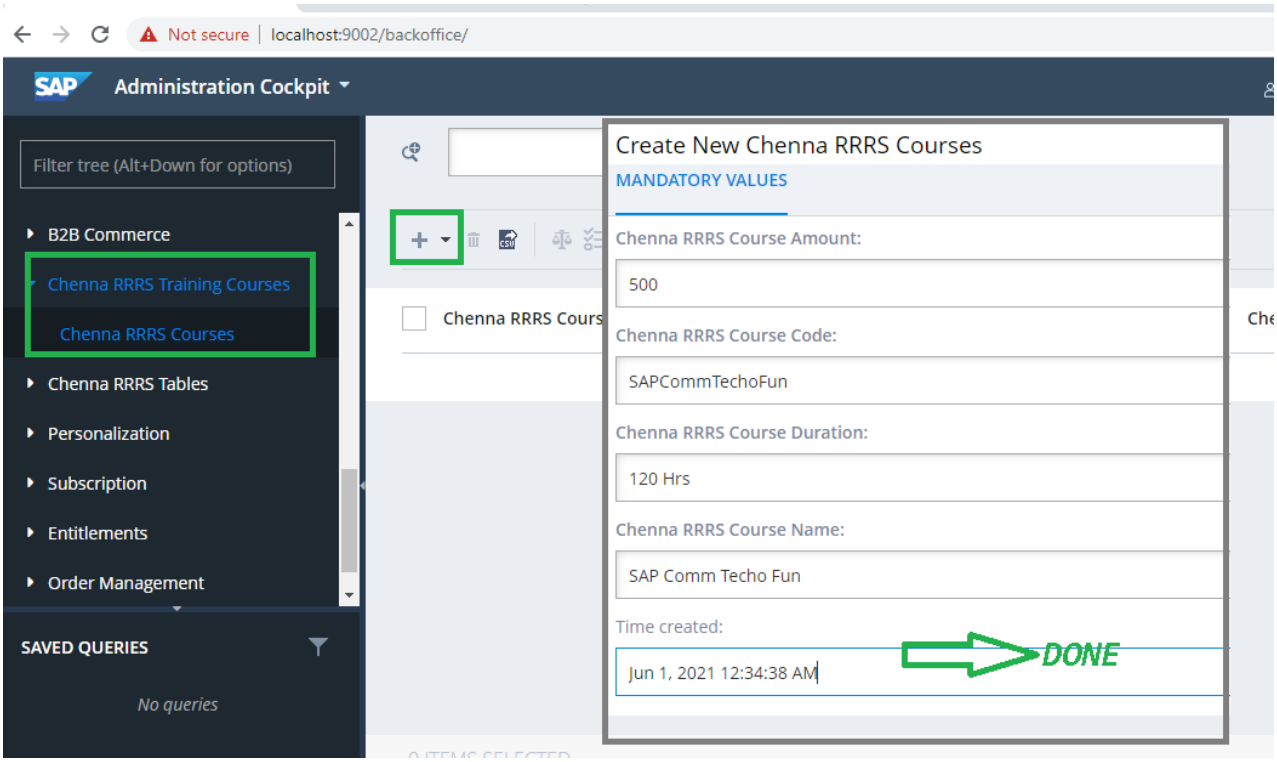


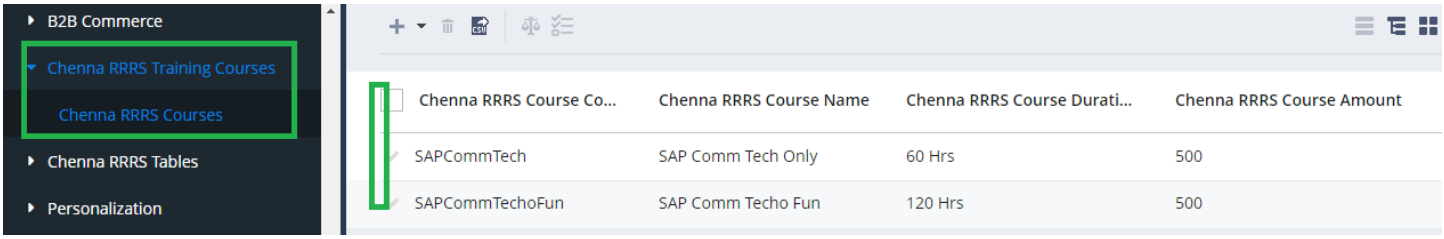
Step 5 = Do the build = E:\rrrssoftware\hybris\bin\platform>**ant clean all**

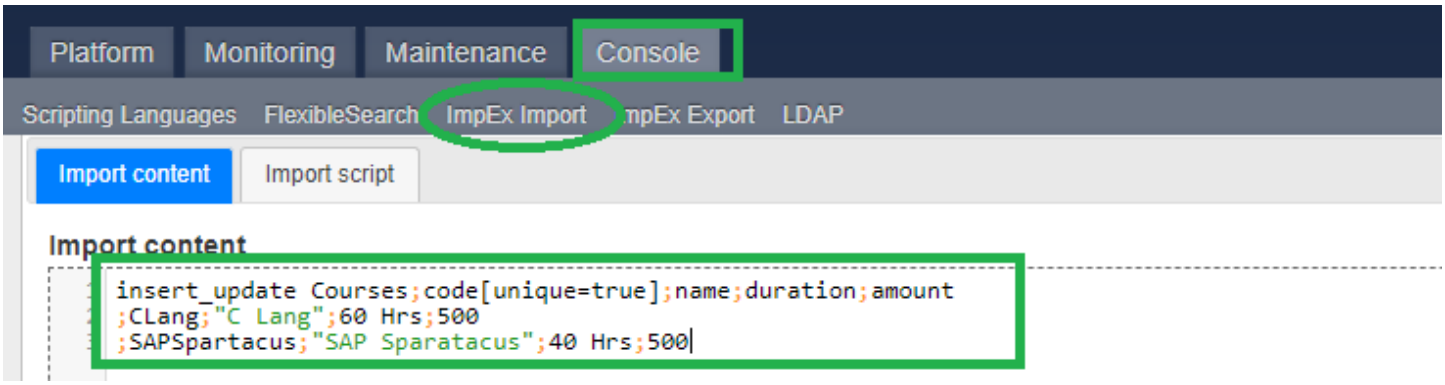
Step 6 = E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

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

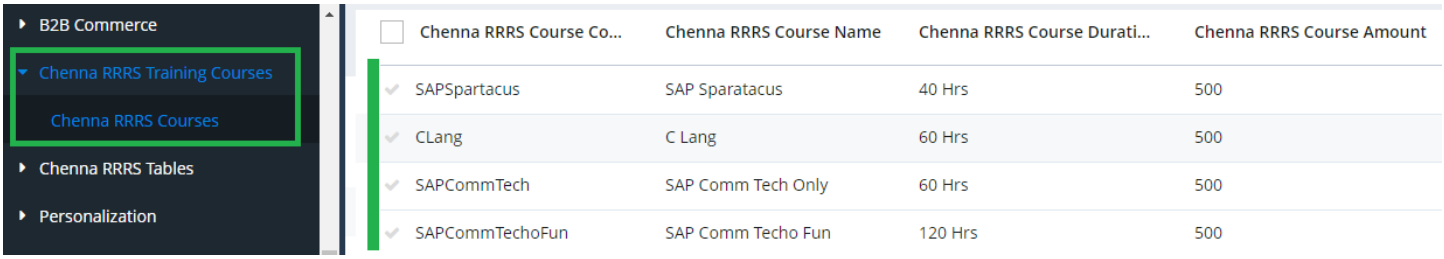
Step 8 = Results







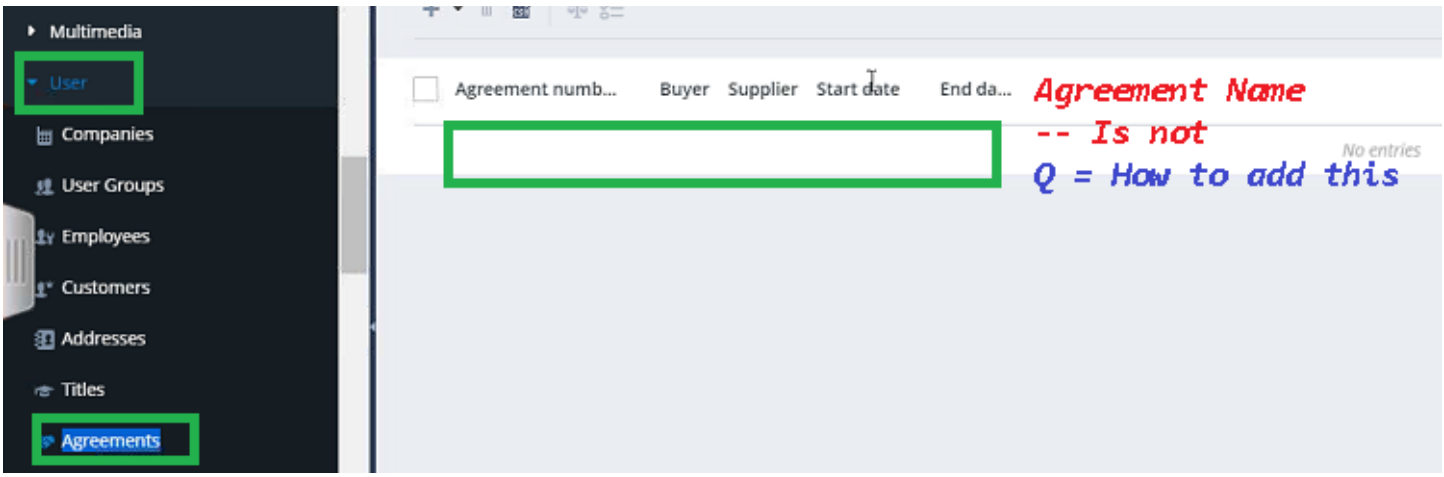
Output =



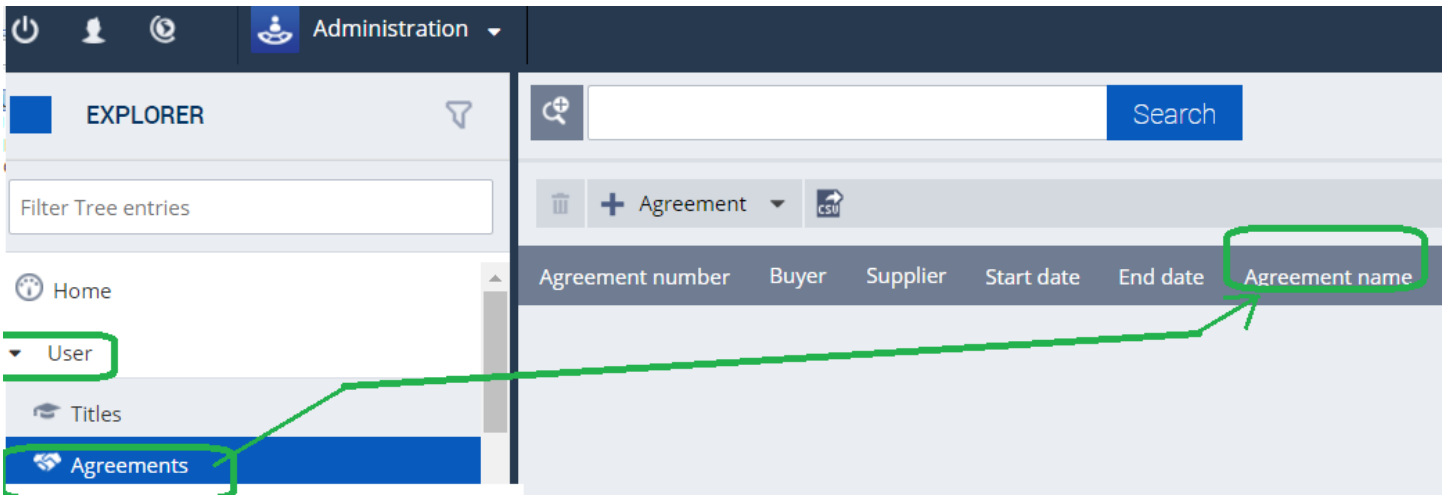
Scenario = How to add a **new field / new attribute** in the backoffice to the **existing node**? (or) How to customize the back office with new attribute in the existing node?

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 exteding existing type called **Agreement** – and 1 new Cols **[agreementName].**

chennarrrstrainingcore-items.xml

```
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>
```

Step 2 = Give proper display lable names

chennarrrstrainingcore-items.xml | chennarrrstrainingcore-locales_en.properties

```
# Example 7
type.MyAgreement.agreementName.name = Agreement Name
```

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

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

chennarrrstrainingbackoffice-backoffice-config.xml

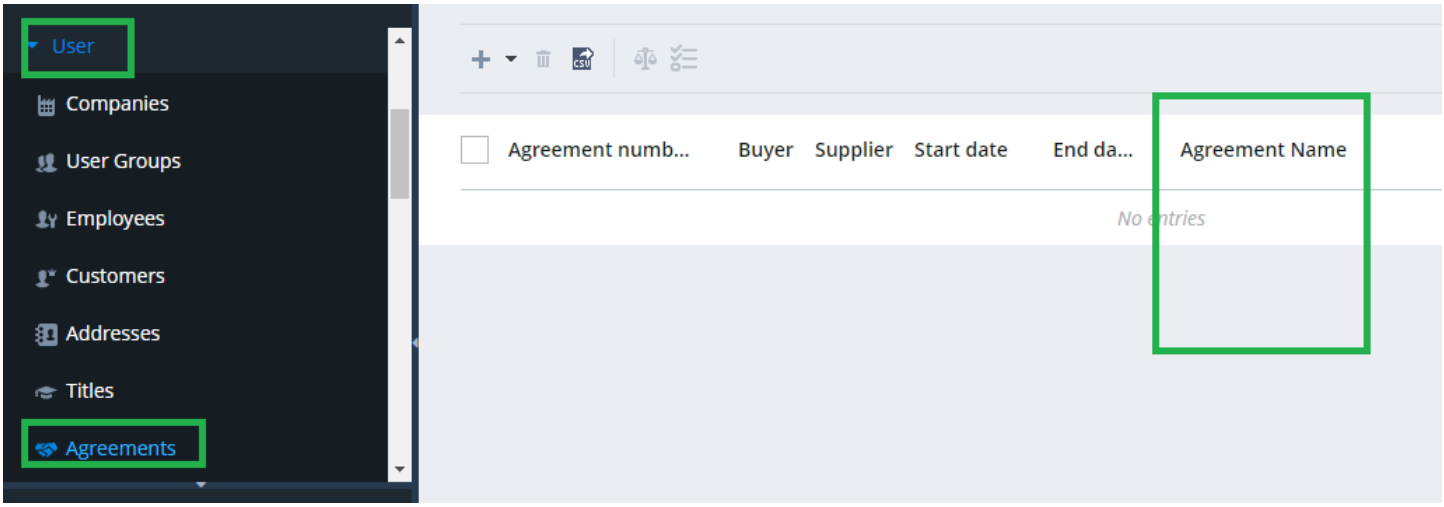
```
83  <!--  Example 7 -->
84  <context component="explorer-tree" merge-by="module">
85  <n:explorer-tree xmlns:n="http://www.hybris.com/cockpitng/config/explorertree">
86      <n:navigation-node id="hmc_treenode_user">
87          <n:type-node id="hmc_typenode_agreement" code="MyAgreement" />
88      </n:navigation-node>
89  </n:explorer-tree>
90  </context>
91
92  <context type="MyAgreement"  parent="Agreement" merge-by="type" component="listview">
93  <list:list-view xmlns:list="http://www.hybris.com/cockpitng/component/listView">
94      <list:column qualifier="agreementName"/>
95  </list:list-view>
96  </context>
```

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

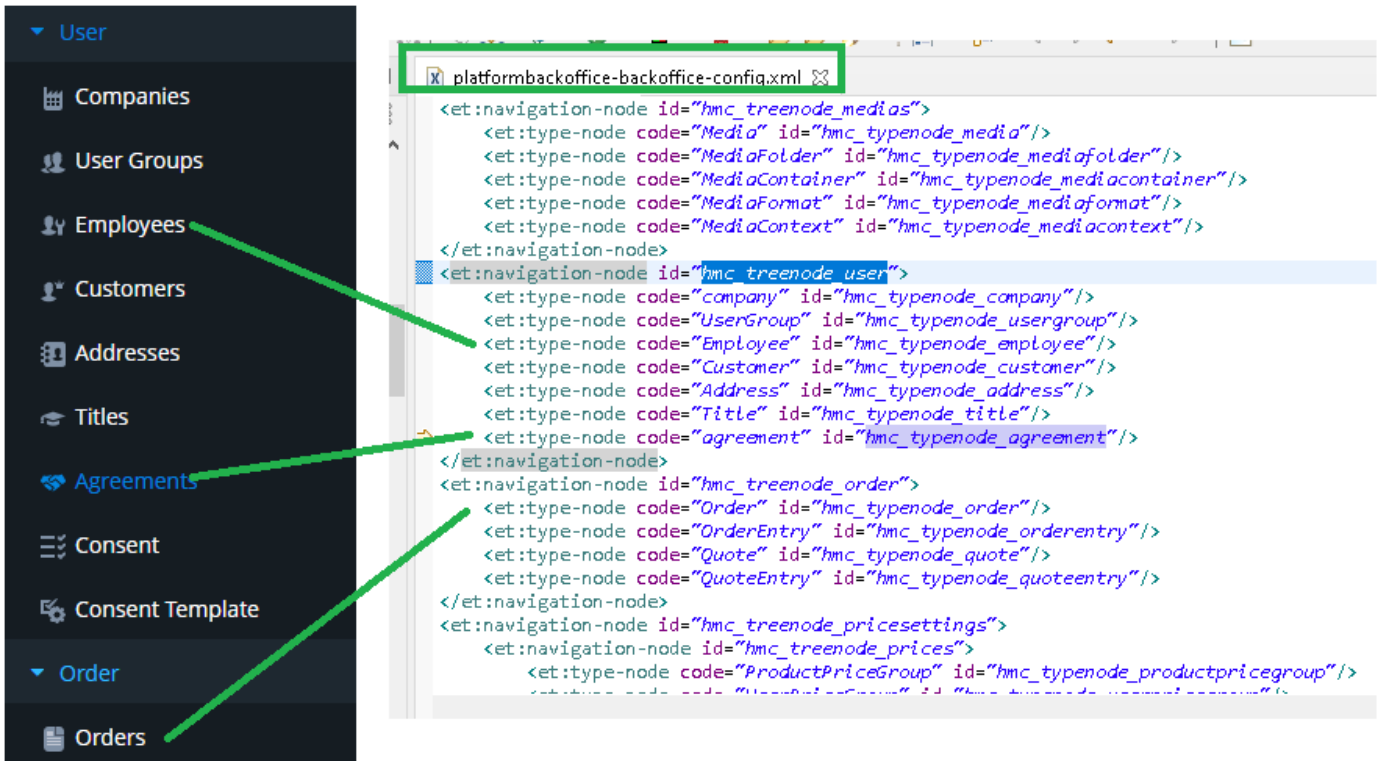
Step 5 = E:\rrrssoftware\hybris\bin\platform>**hybrisserver.bat**

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

Results =

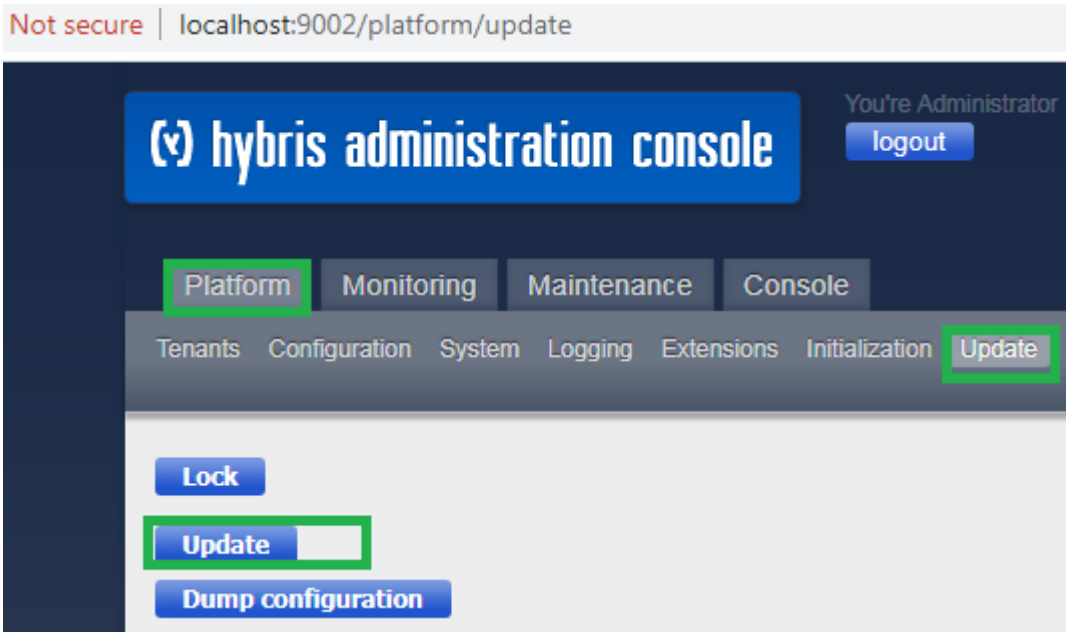


Q = How to find the node Id’s?

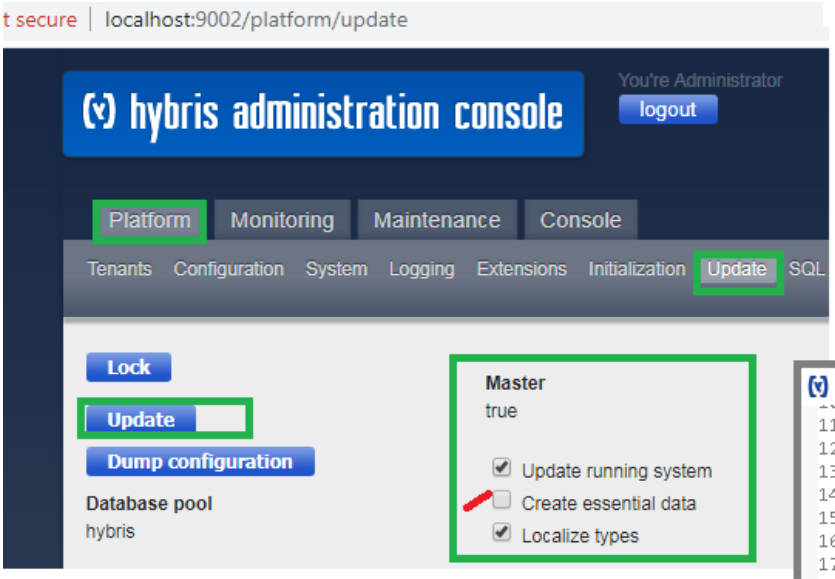


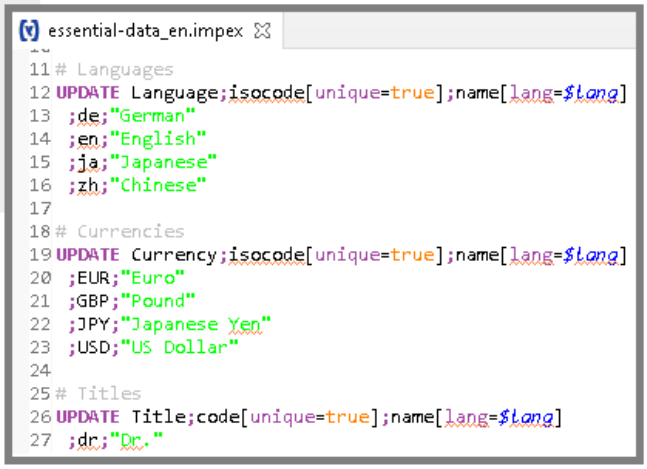
Scenario = hAC – Platform – Update Options

Solution 1 =

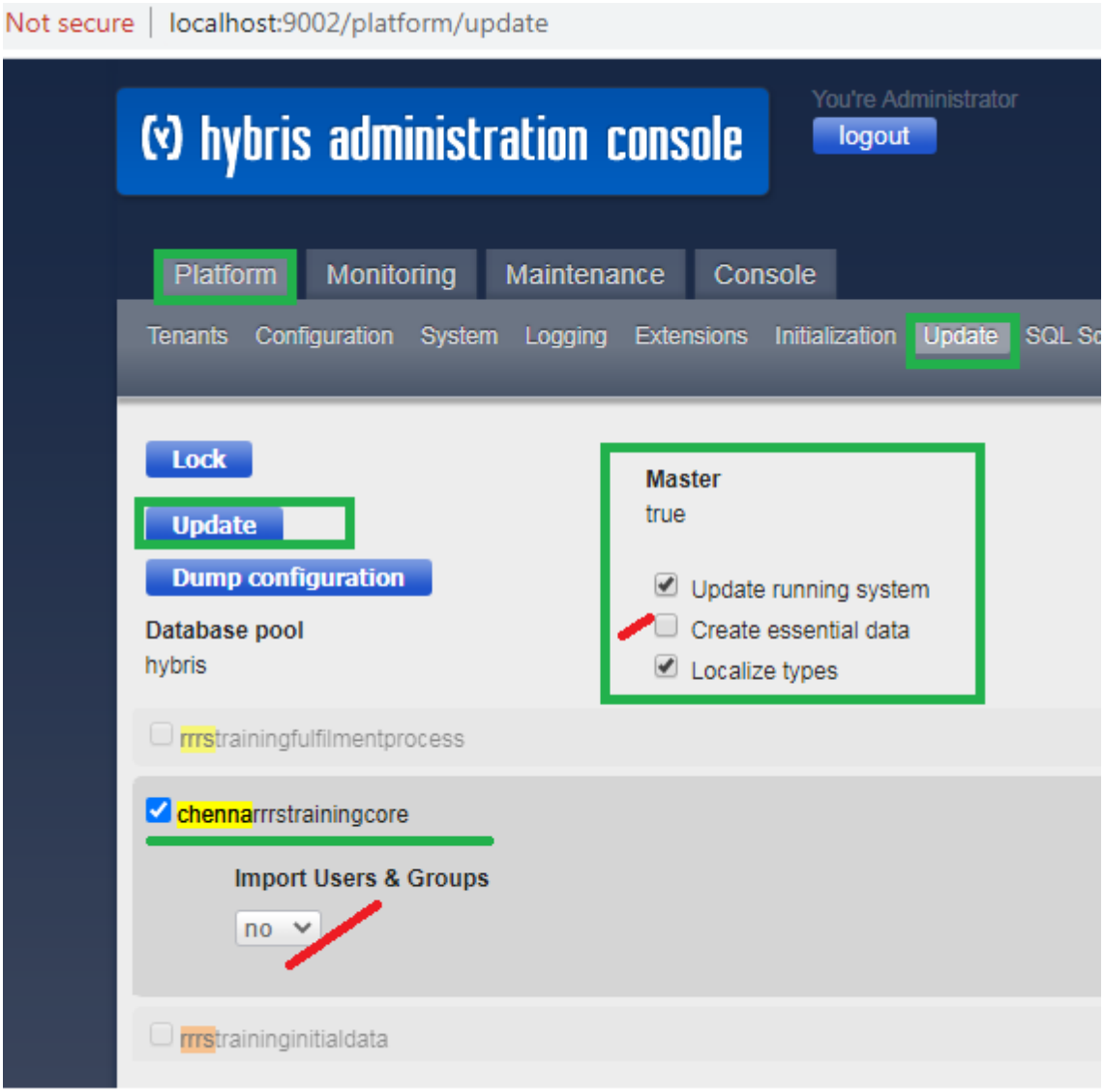


Solution 2 =





Solution 3 =

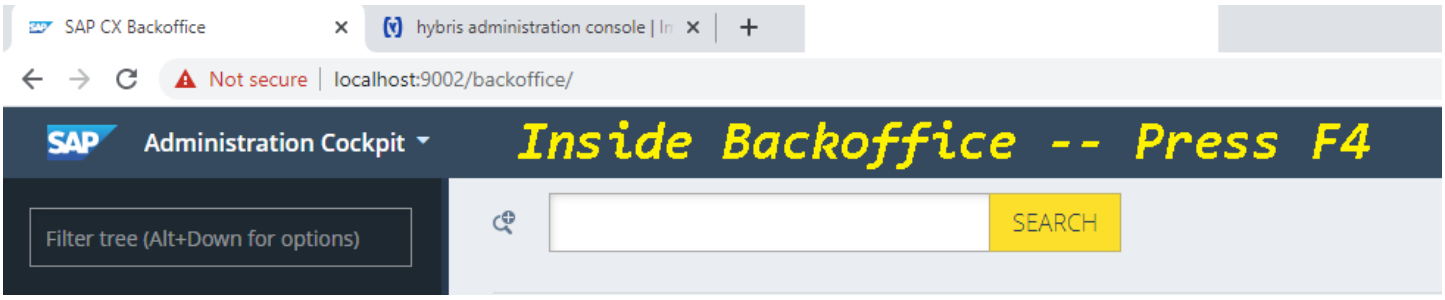

Not secure | localhost:9002/platform/update
hybris administration console
You're Administrator
logout
Platform
Monitoring
Maintenance
Console
Tenants
Configuration
System
Logging
Extensions
Initialization
Update
SQL Sc
Lock
Update
Dump configuration
Database pool
hybris
Master
true
Update running system
Create essential data
Localize types
rrrstrainingfulfilmentprocess
chennarrrstrainingcore
Import Users & Groups
no
rrrstraininginitialdata

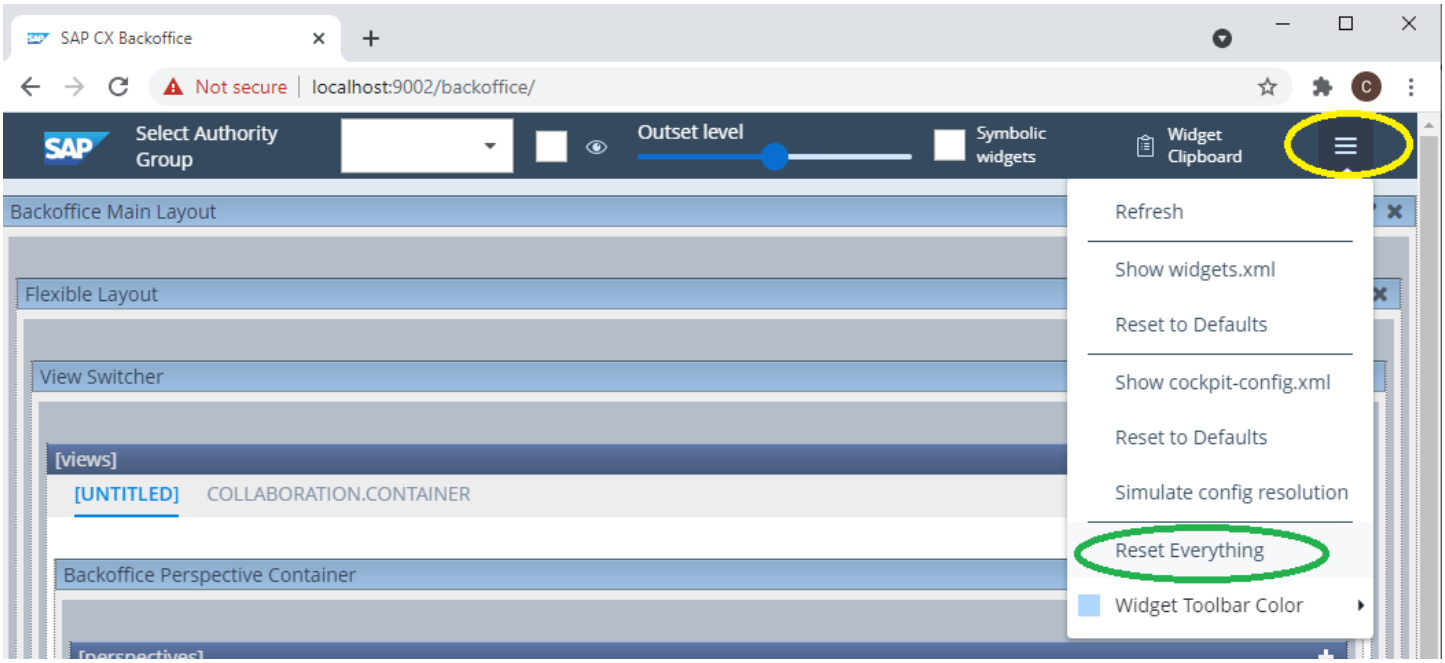
Note = When we do “backoffice customization changes” – some times we don’t see expected results even though the everything correct.

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 ***wedget*.xml** file.

You can find widgets information in below files: -

platformbackoffice-backoffice-config.xml

platformbackoffice-backoffice-widgets.xml

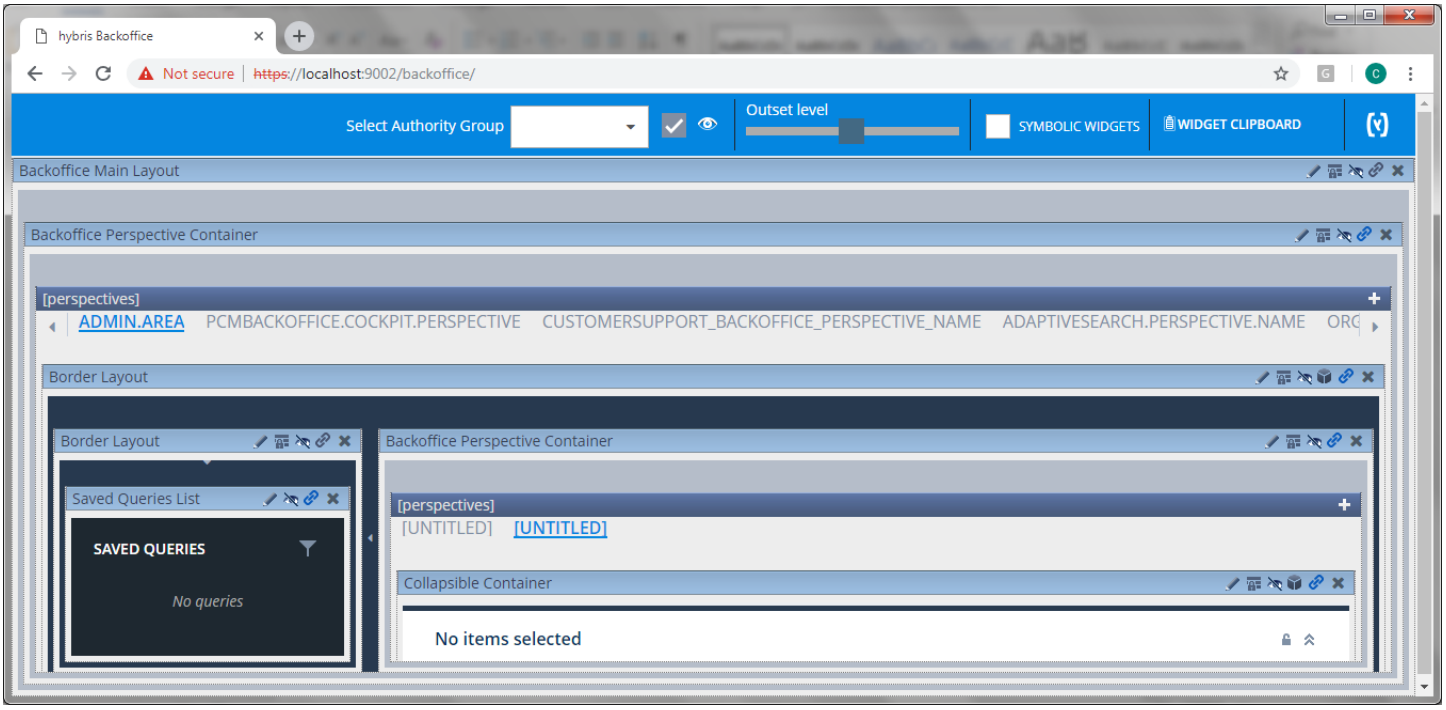
backoffice-widgets.xml

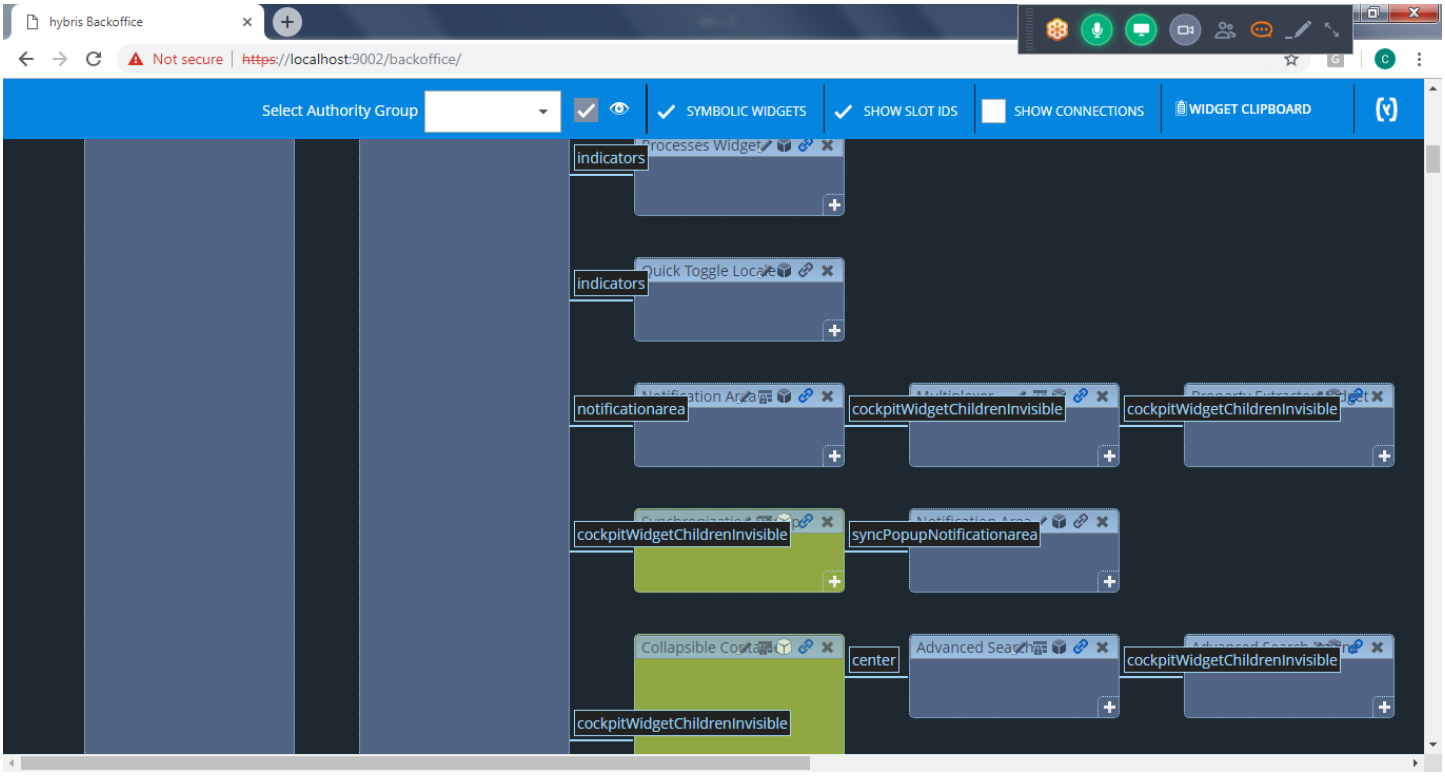
All these widgets results are combined & you have the backoffice.

1 widget can be connected with another widget.

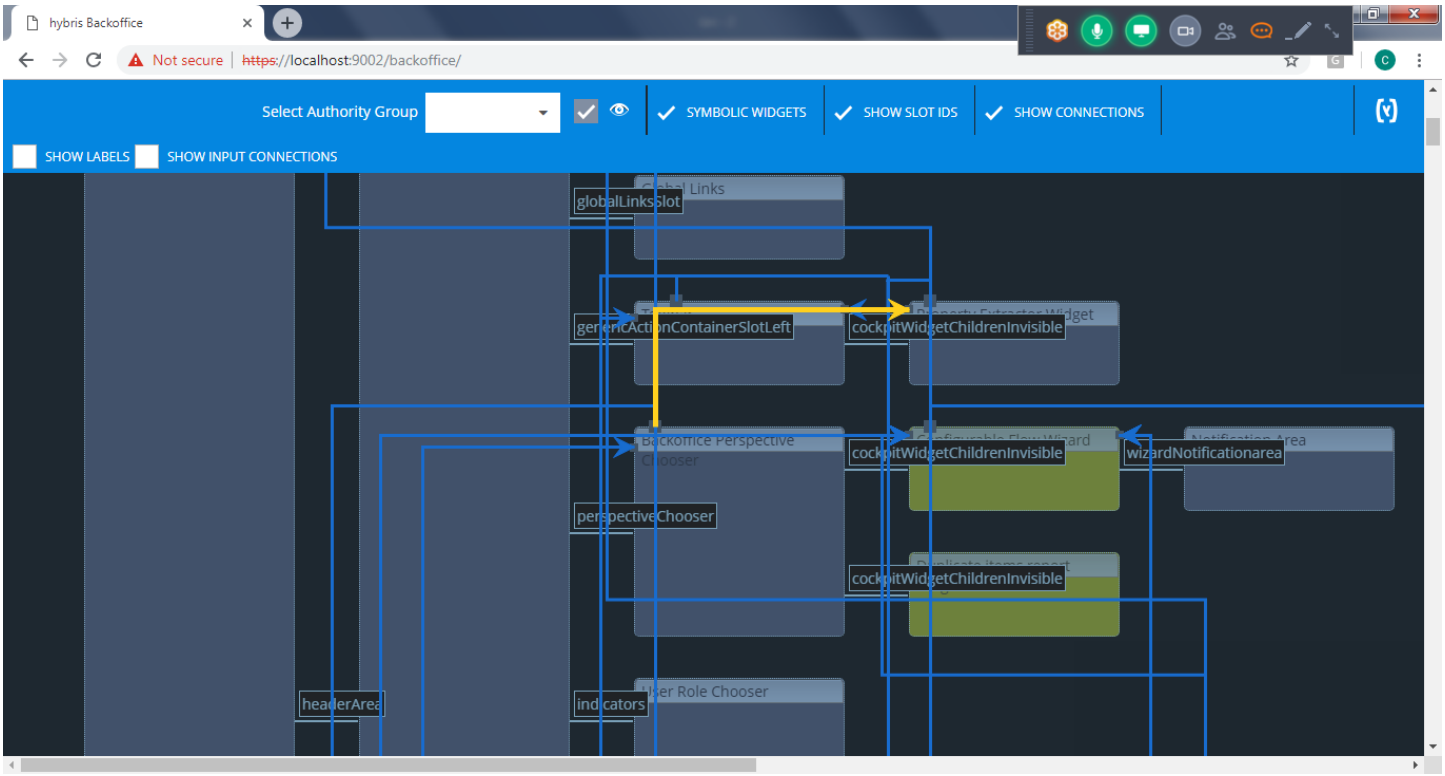
Orchestration mode – Will give full picture.

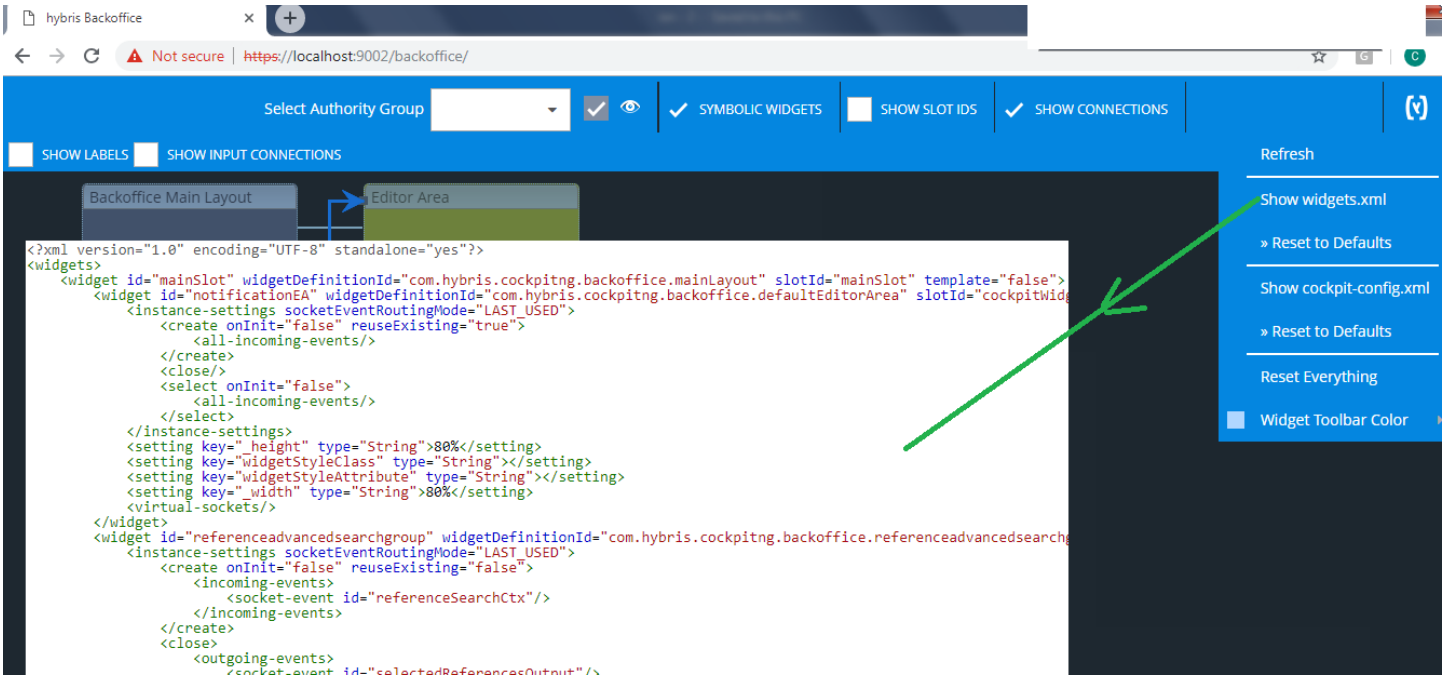
In Backoffice – Press F4 == It will take you to “Backoffice – Orchestration”





➔ 1 widget is connected another widget.

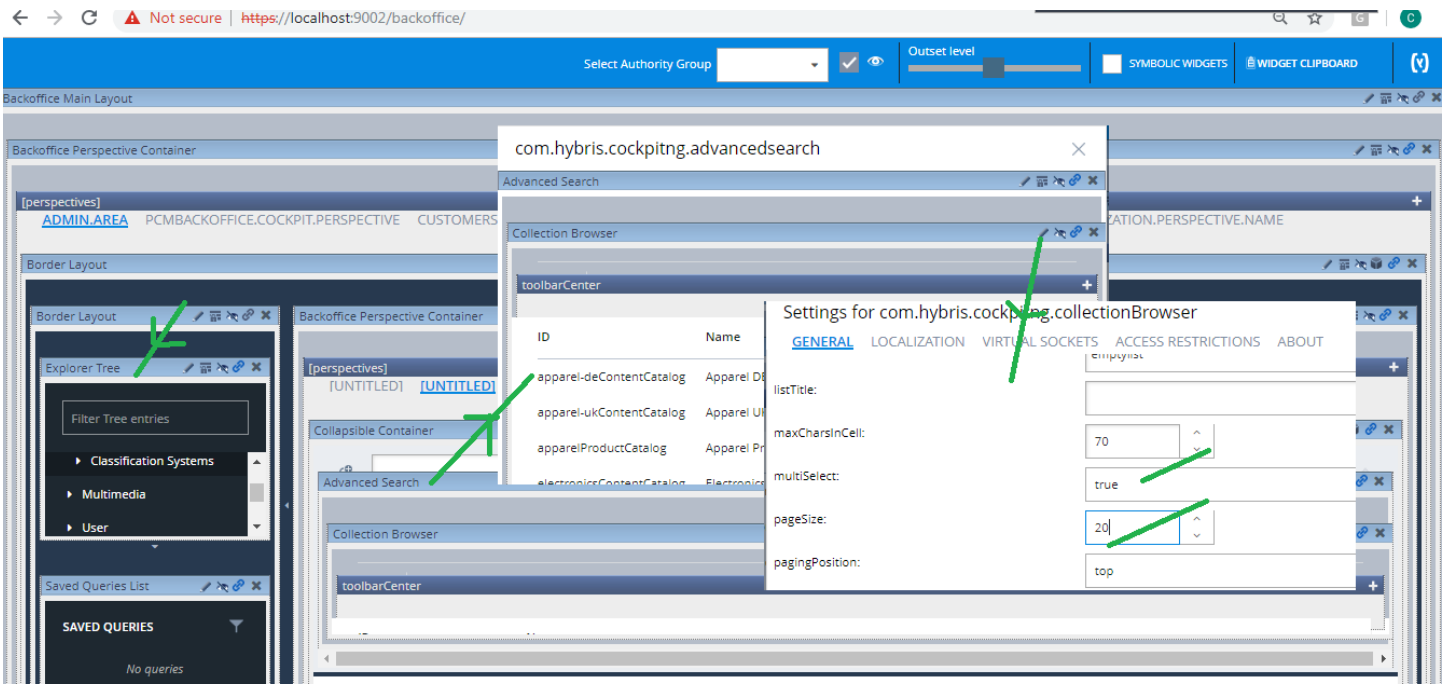


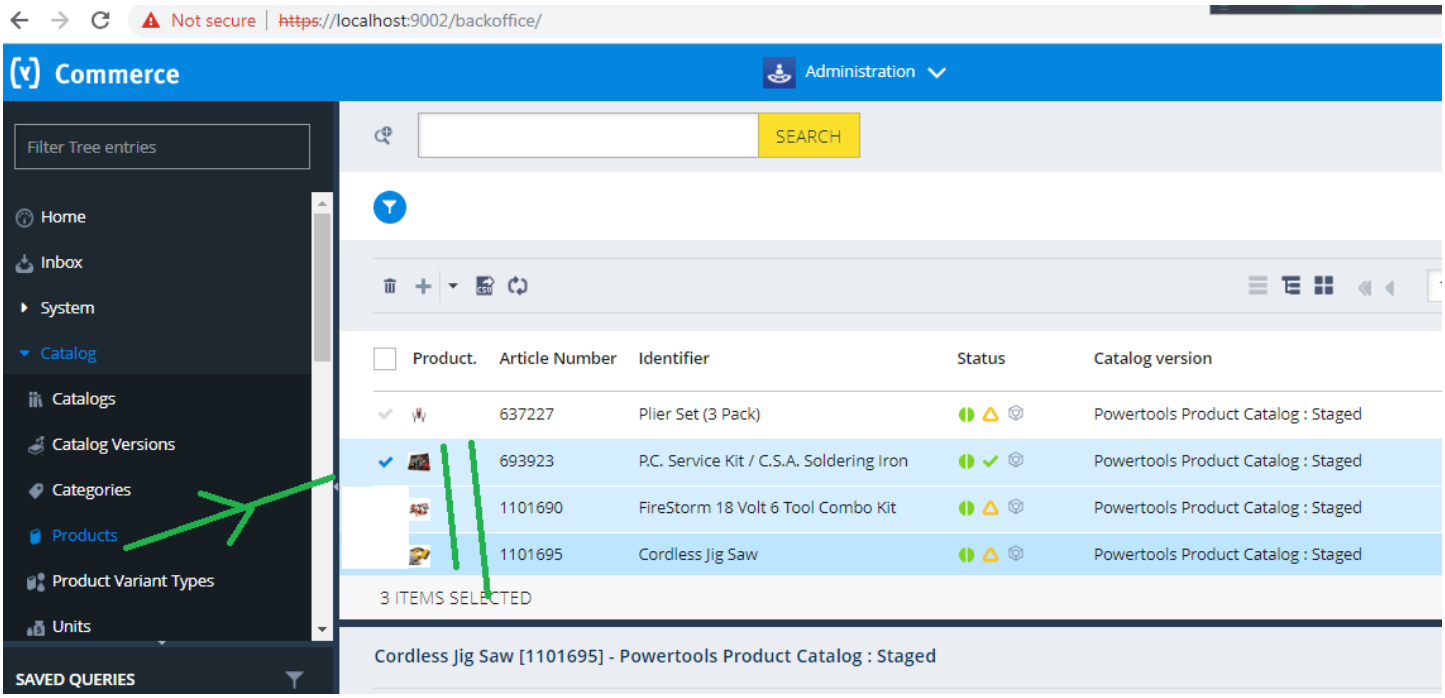


Q1 = By default multi product selection allowed.

How to Disable “Multiple Product Selection”?

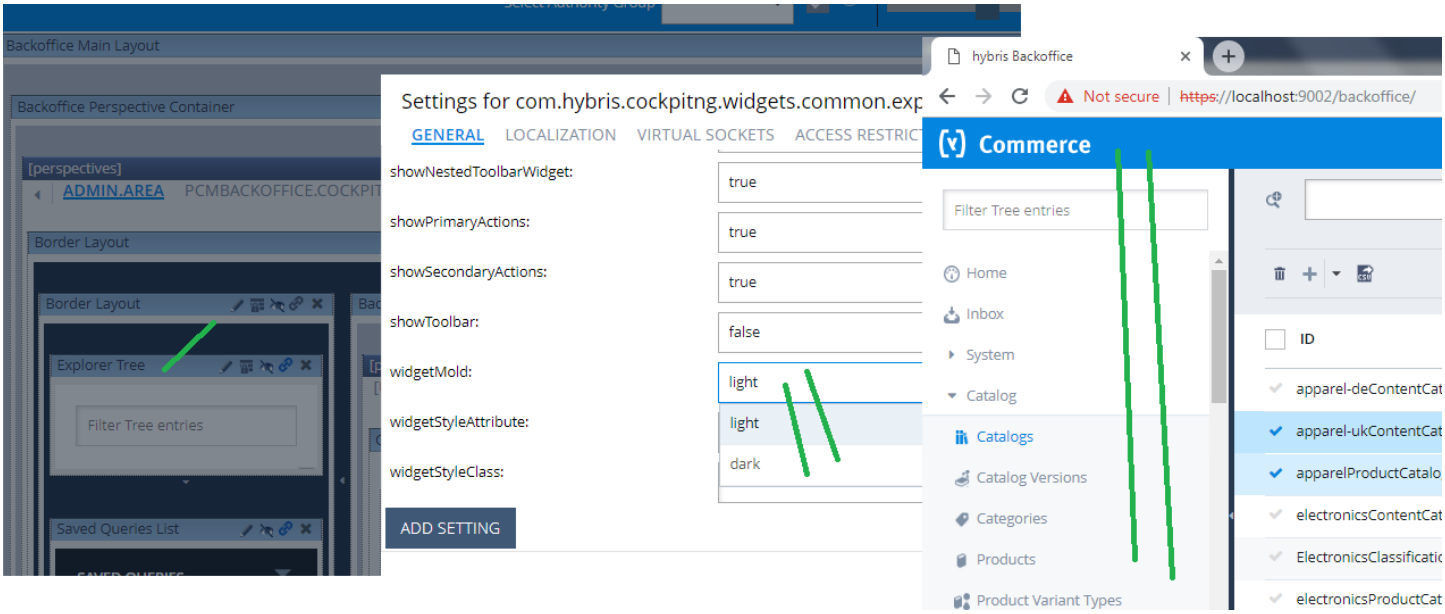
In Backoffice → Press F4 → Explorer Tree → Advanced Search → Edit





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

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



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

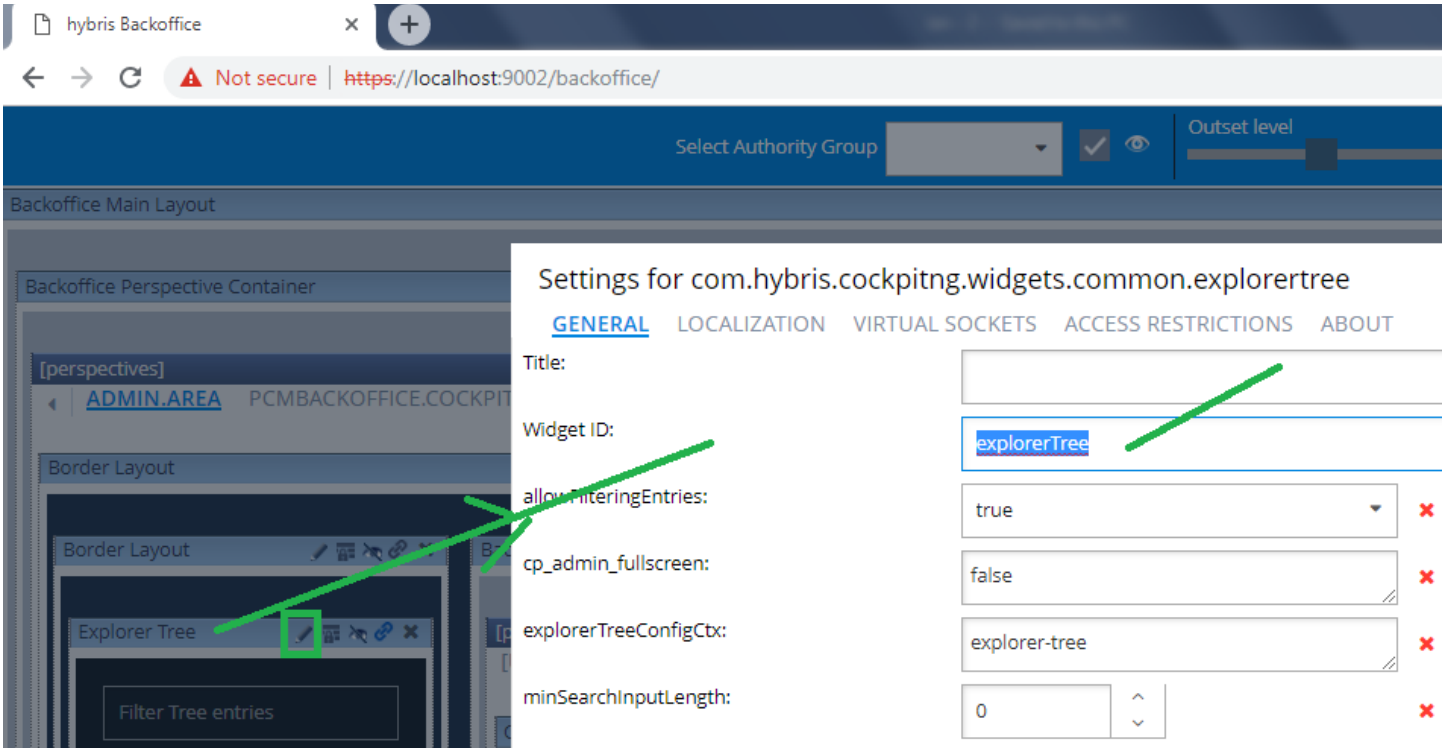
How to Disable that?

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

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

Solution =

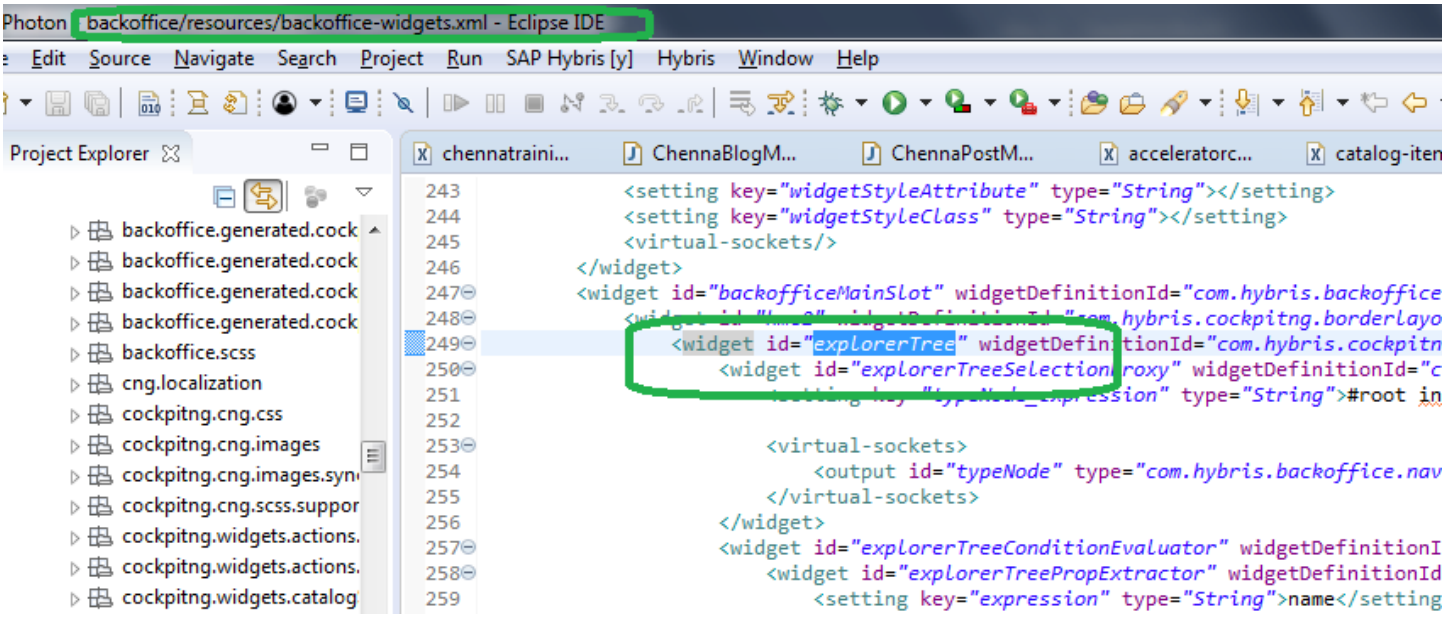
Step 1 = From Orchestration mode – Find the Widget ID = **explorerTree**

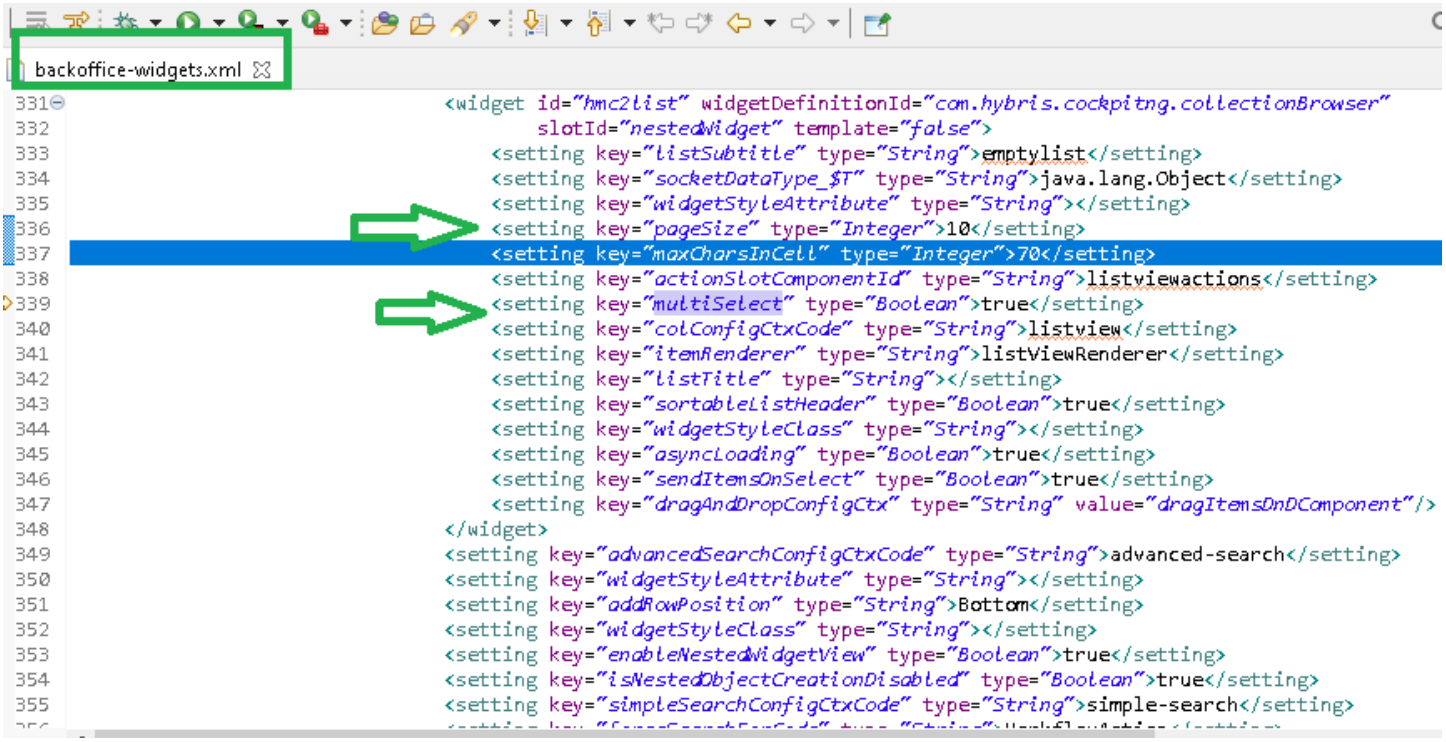


Step 2 = Take the widget ID & go to "**backoffice-widget.xml** (or) ***-widget.xml**" & do the corresponding code changes.

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

Then do the corresponding changes.


Photon backoffice/resources/backoffice-widgets.xml - Eclipse IDE
Edit Source Navigate Search Project Run SAP Hybris [y] Hybris Window Help
Project Explorer
backoffice.generated.cock
backoffice.generated.cock
backoffice.generated.cock
backoffice.generated.cock
backoffice.scss
cng.localization
cockpitng.cng.css
cockpitng.cng.images
cockpitng.cng.images.syn
cockpitng.cng.scss.suppor
cockpitng.widgets.actions.
cockpitng.widgets.actions.
cockpitng.widgets.catalog
chennatraini...
ChennaBlogM...
ChennaPostM...
acceleratorc...
catalog-iten
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="explorerTree" widgetDefinitionId="com.hybris.cockpitng.borderlayo
249 <widget id="explorerTree" widgetDefinitionId="com.hybris.cockpitn
250 <widget id="explorerTreeSelectionProxy" widgetDefinitionId="c
251 type="String">#root in
252
253 <virtual-sockets>
254 <output id="typeNode" type="com.hybris.backoffice.nav
255 </virtual-sockets>
256 </widget>
257 <widget id="explorerTreeConditionEvaluator" widgetDefinitionI
258 <widget id="explorerTreePropExtractor" widgetDefinitionId
259 <setting key="expression" type="String">name</setting


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_$T" 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>


backoffice-widgets.xml
organization-backoffice-widgets.xml
commerceservicesbackoffice-backoffice-widgets.xml
41 <virtual-sockets/>
42 </widget>
43 <setting key="listSubtitle" type="String"></setting>
44 <setting key="socketDataType_$T" 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.**

Above scenario similar to below…