

Layers: -

In “SAP Comm” – We have structured way of writing the code.

You take any application – there will be **2 flows**: -

1st Flow = Registration (You Enter the data in the Form)

= **Uni-Directional** Flow

PL (JSP file / Tag file / HTML file / JS file / CSS file /) → Controller → Façade Layer (FL) → Service Layer (SL) → Perform the DB Save Operation.

Controller = Java file

Q: When can you say a Java file = Controller?

@Controller (Annotation)

```
83 */  
84 @Controller Annotation  
85 @RequestMapping(value = "/cart")  
86 public class CartPageController extends AbstractCartPageController  
87 {  
88     public static final String SHOW_CHECKOUT_STRATEGY_OPTIONS = "storef  
89         public static final String FDDDD_MCG_TVDF = "errorMen".
```

1 Controller can call N number of other controllers.

FL = Java file.

1 FL can call another FL. So, we can have N number of FL.

SL = Java file.

1 SL can call another SL. So, we can have N number of SL.

We can have 1 / more JSP / Tag files for the same.

Q: When to write code in JSP File & When to write code in Tag file?

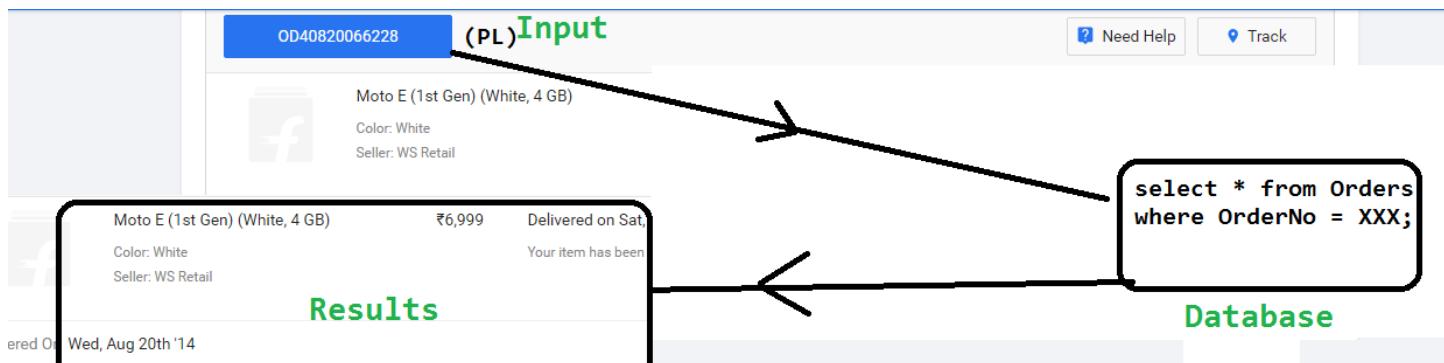
Let's say -- U have XXX code & you want to re-use that code in more than 1 place then put that XXX code in Tag file.

Else -- You can create JSP File

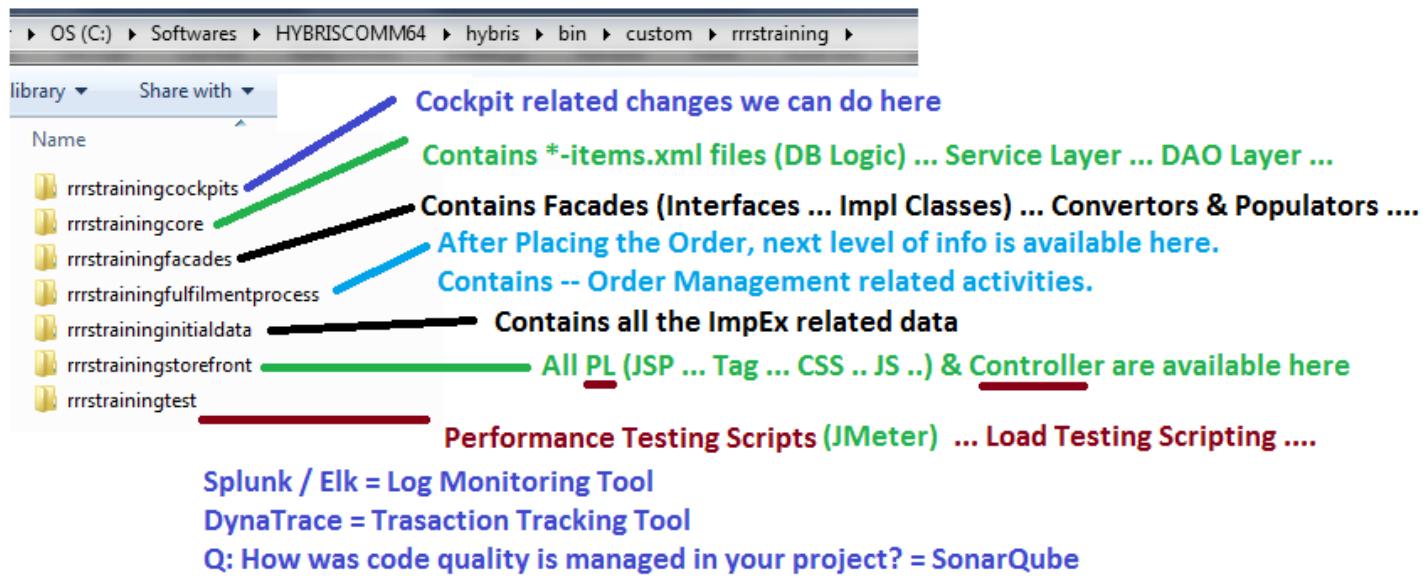
2nd Flow = Order Status (Enter Order No & Display the status)

= Bi-Directional Flow

PL (JSP file / Tag file / HTML file / JS file / CSS file /) → Controller → Façade Layer (FL) → Service Layer (SL) → DAO Layer (You Write the Flexible Search Query) → Service Layer → Façade Layer → Controller → Presentation Layer (PL) & Display the results.



Note: - In “SAP Comm” projects, we will be having 7 (or) More Exts.



Note: - “SAP Comm” is having different layers for coding.

Q = Can I write all the code in 1 file? = Yes, you can.

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Q = Why do we need to write coding in those many layers?

= **Loosely Coupling.**

DAO Layer (Here you write the Flexible Search Queries = FSQ)? →

If you take any DB, what can we do? =

1) **DDL** = Create table ... Create Cols

For DDL Operation, in “SAP Comm” what do we have? = *-**items.xml**

2) Insert Record ... Update Record ... Delete Record

For this DML Operations, in “SAP Comm” what do we have? = **ImpEx**

3) Get the Records (Select Command)

For this DML operation, in “SAP Comm” what do we have? = FSQ

Oracle – Select == “**SAP Comm**” – **FSQ**

Note: - In “SAP Comm” – “80 - 90%” of the code already available. Only “20 - 10%” of code we write based on customers need.

Now – It’s import for us to find out existing code & write the new code on top of it.

Note: - In eCom Site – We will be having different pages. Example: -

1) Cart Page 2) Checkout Page

3) Quick Order Page 4) Product List Page 5)

For every page there will be 1 Controller.

For Cart Page → We have “**CartPageController.java**”

For Checkout Page → We have “**MultiStepCheckoutController.java**”

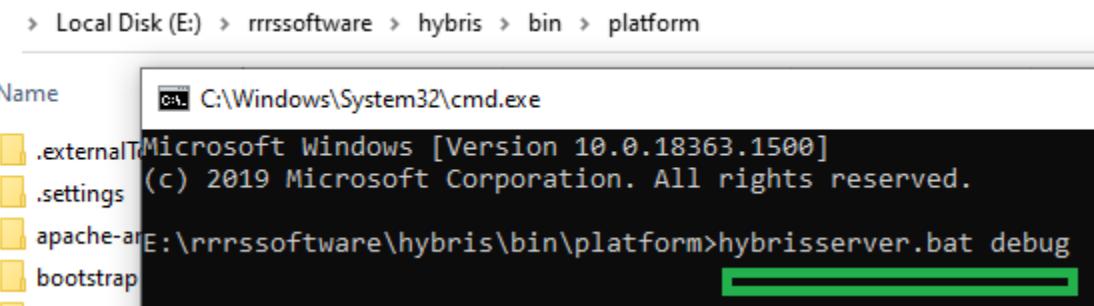
For Quick Order Page → We have “**QuickOrderPageController.java**”

Q = How to identify right files? = Put debug in code & find out.

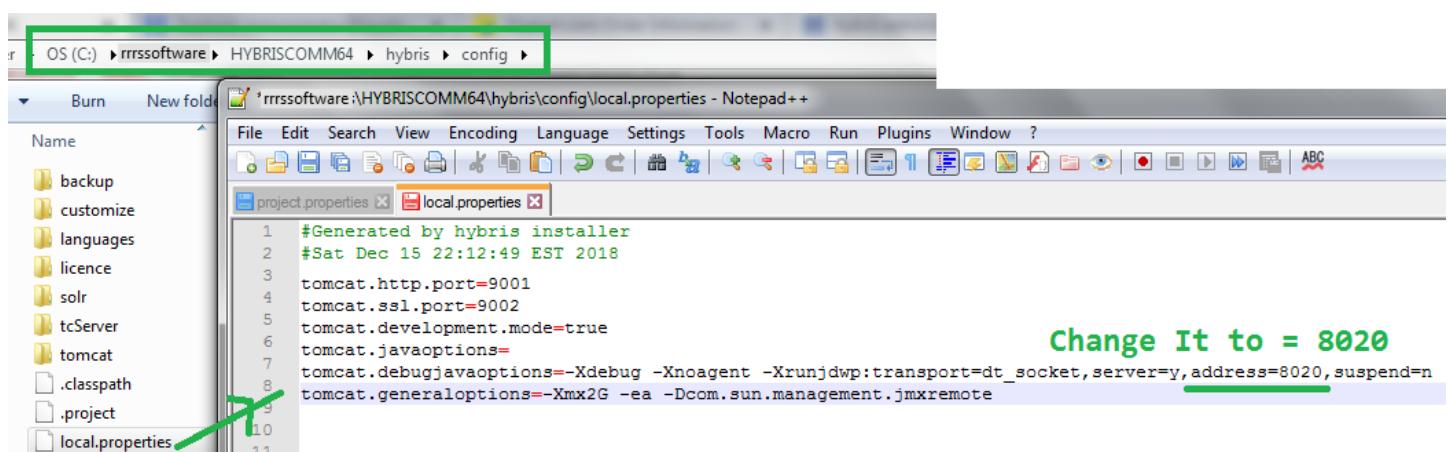
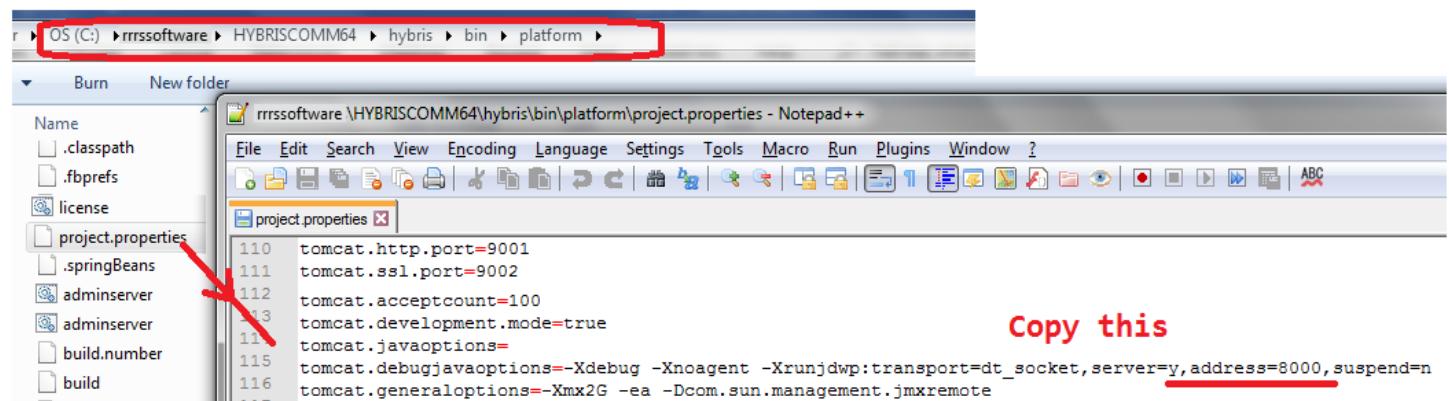
Q = How to debug the code? =

Step 1 = Start the server in debug mode

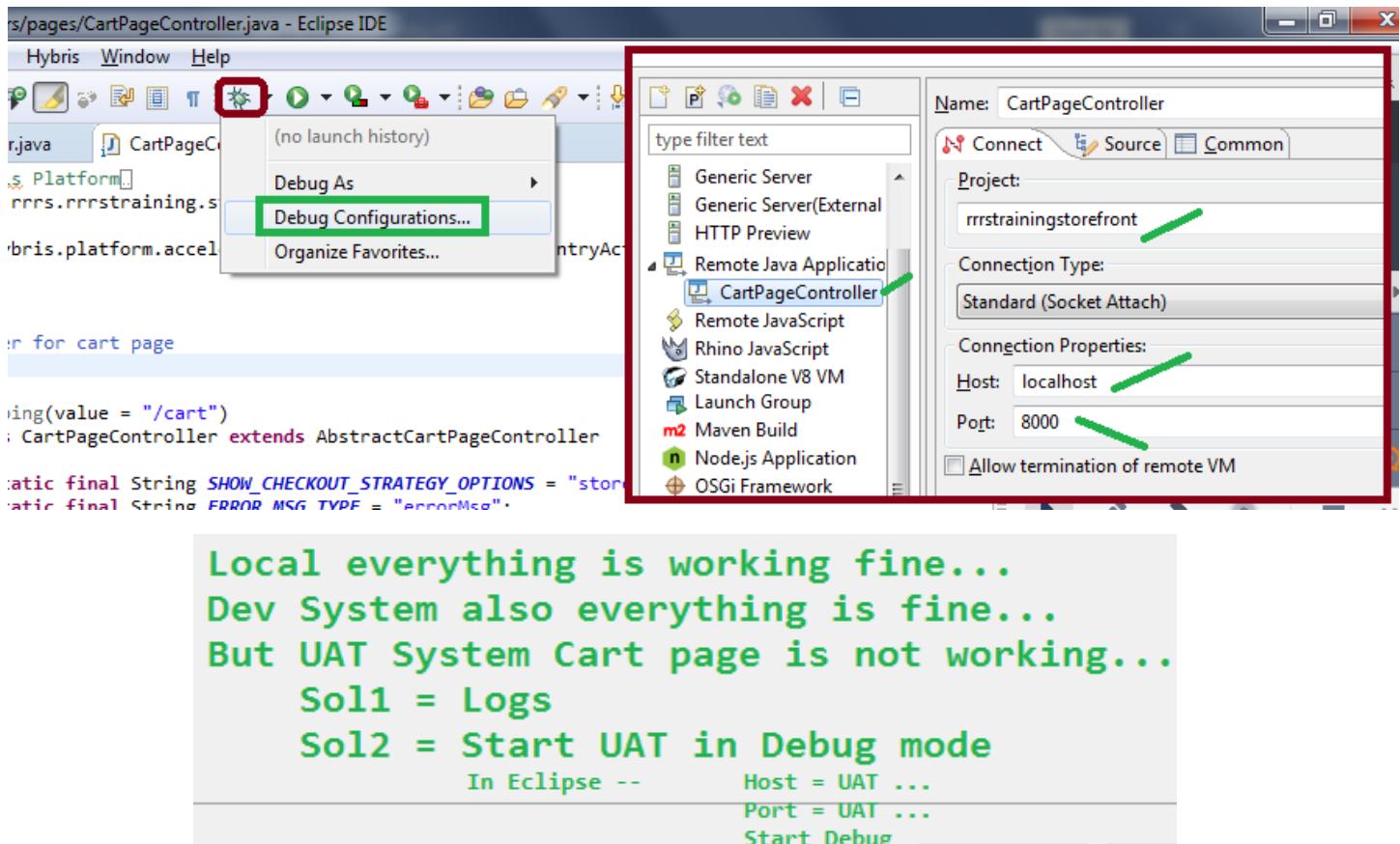
C:\rrrsssoftwares\hybris\bin\platform>**hybrisserver.bat debug**



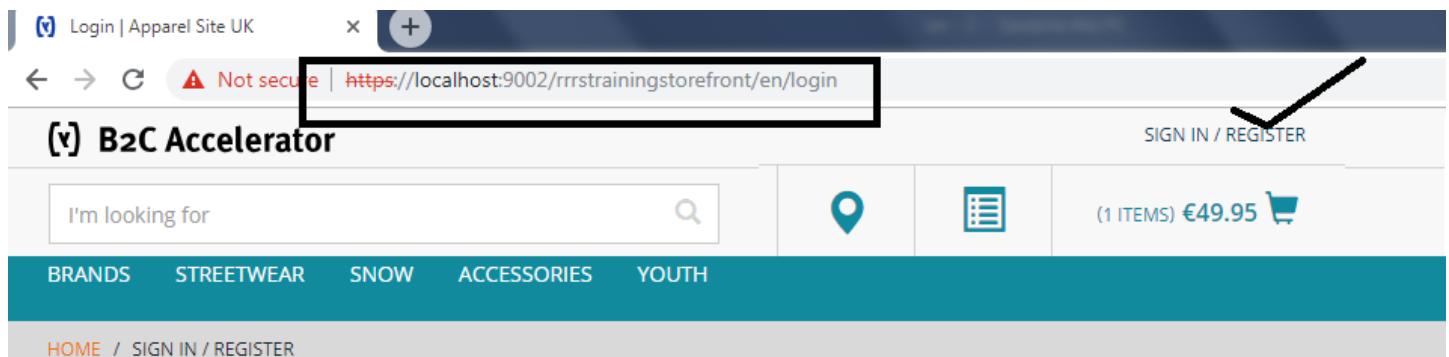
Note: - debug runs at specific port (=8000 default). Sometimes, you get error saying that port is already used. So – How to change the port / How find the current port?



Step 2 = In Eclipse, Configure the debug settings



1st Flow = Registration Example: -



Contact Us = ChennaReddyTraining@RRRS.CO.IN

Req1 = How to add another filed called “Enter Address” after 1st Name & Last Name?

Step 1 = Q - How to find existing PL layer?

Sol1 = We know that, PL logic is available in “***storefront**” (*=RRRSTraining) Ext.

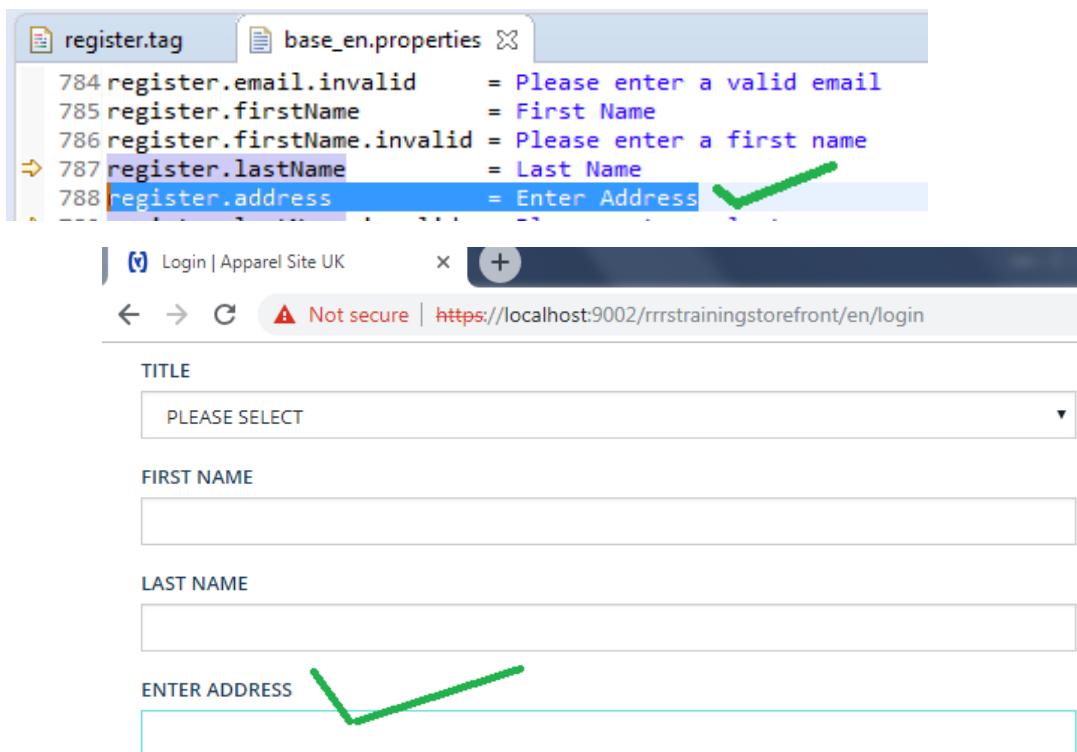
Note: - Hybris 6.X version on words – It's responsive.

So, go to ***storefront** Ext & find the corresponding file & do the changes.

`/*storefront/web/webroot/WEBINF/tags/responsive/user/register.tag`



`/*storefront/web/webroot/WEBINF/messages/base_en.properties`



Sol2 = Go to the required place where you want the changes, RBM → Inspect Element, identify some unique element (ID name or Class or ...)

The screenshot shows a web browser window with a login form titled "rrrstrainingstorefront/en/login". The form includes fields for TITLE, FIRST NAME, LAST NAME, and EMAIL ADDRESS. A callout arrow points from the text "form#registerForm" in the browser's developer tools to the "TITLE" field of the form. The developer tools interface is visible at the bottom, with the "Elements" tab selected.

The screenshot shows the SAP Hybris IDE interface. A search dialog is open in the center, with the "Search" tab highlighted. The search term "registerForm" is entered in the "Containing text:" field. Below it, the "File name patterns (separated by comma):" field contains "*jsp, *tag". The search results pane on the right shows a tree structure under "rrrstrainingstorefront/web", indicating 6 matches found.

The screenshot shows the SAP Hybris IDE interface with the search results for "registerForm" displayed. The results show 6 matches found across the workspace. The search term "registerForm" is highlighted in the search bar and the results list. The results pane shows the file structure where the matches were found.

**6 matches found.
Now, identify which is yours**

From 6 matches, find which our current requirement file is. Open that file
→ Delete the code. Save it. Refresh the browser & see the results.

Q = What are we trying to do in PL? = We are entering lots of field values (FName, LName, Address and ...)

LAST NAME
LName

EMAIL ADDRESS
abcd@gmail.com

PASSWORD
.....
Minimum length is 6 characters

CONFIRM PASSWORD
.....

Enter the Data

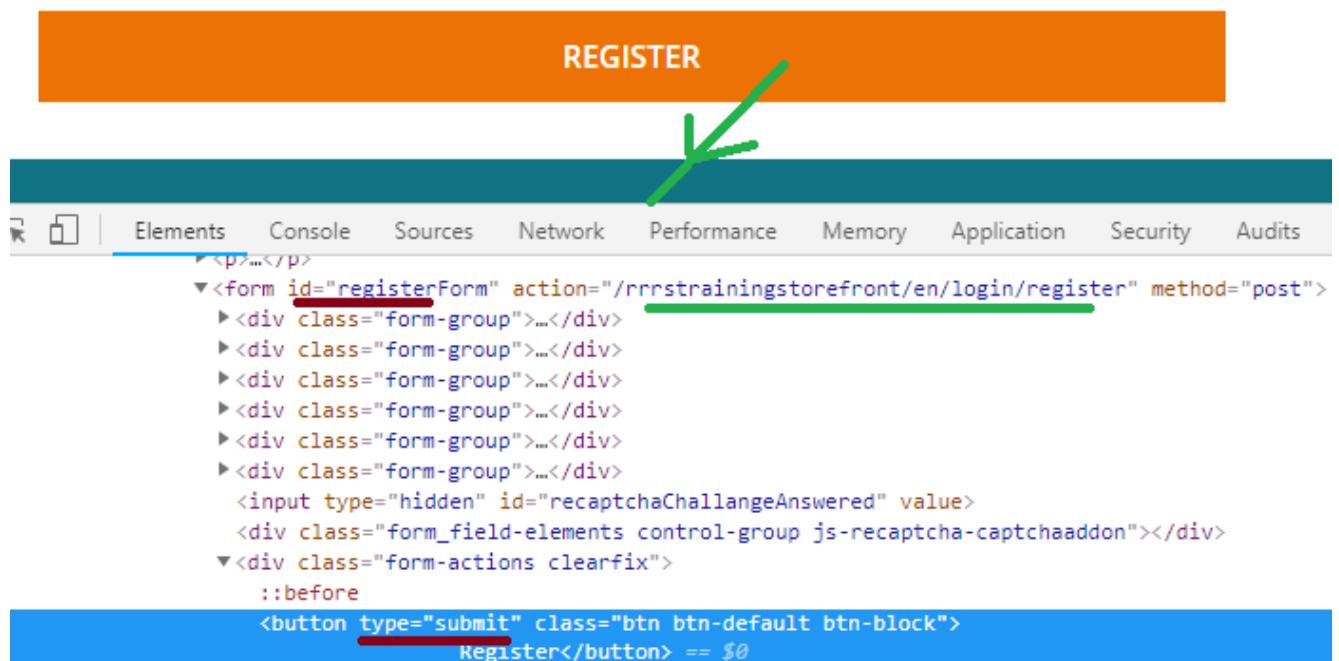
When you click on Register Controller is called.

REGISTER

Note = After entering all the data → **Register, Controller** will be called.

Controller code / file will be available in ***storefront Ext.**

Step 2 = Q: How to find out corresponding Controller?



Contact Us = ChennaReddyTraining@RRRS.CO.IN

action="/rrrstrainingstorefront/en/login/register"

action="/rrrstrainingstorefront/en/login/register"

We already in
Storefront

Lang Code

/register = This will be a request mapping at method level.

/login = This will be a request mapping at class level.

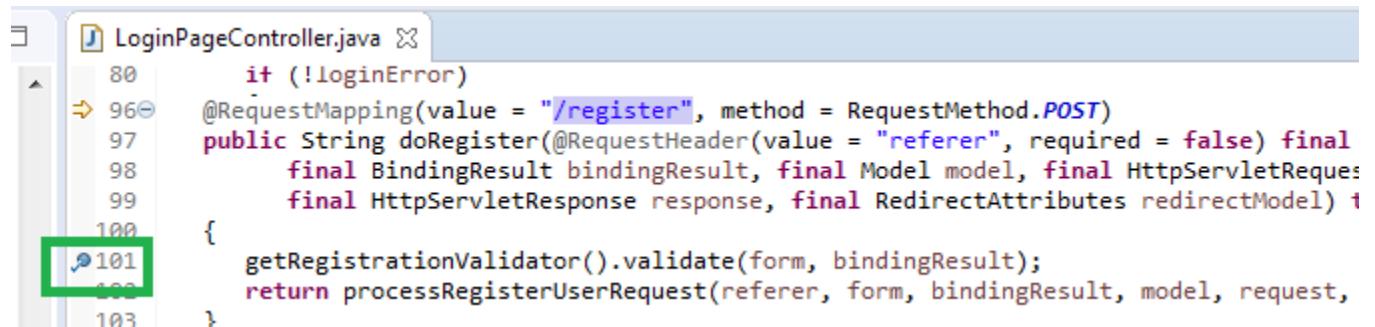


The screenshot shows the Eclipse IDE interface with the Project Explorer on the left and the code editor on the right. The code editor displays the `LoginPageController.java` file. A red checkmark is placed next to the `LoginPageController.java` entry in the Project Explorer. The code in the editor is as follows:

```
36 /**
37  * Login Controller. Handles login and register for the account flow.
38 */
39 @Controller
40 @RequestMapping(value = "/login")
41 public class LoginPageController extends AbstractLoginPageController
42 {
43     private HttpSessionRequestCache httpSessionRequestCache;
44
45     @RequestMapping(value = "/register", method = RequestMethod.POST)
46     public String doRegister(@RequestHeader(value = "referer", required = false) final
47         final BindingResult bindingResult, final Model model, final HttpServletRequest
48         final HttpServletResponse response, final RedirectAttributes redirectModel) {
49
50         getRegistrationValidator().validate(form, bindingResult);
51     }
52 }
```

Q = What is the guarantee that above file is the Correct Controller?

Start the server debug mode & see the flow is coming in this file.



The screenshot shows the Eclipse IDE code editor with the `LoginPageController.java` file open. A red rectangle highlights line 101, which contains the breakpoint annotation (`①`). The code is identical to the one shown in the previous screenshot:

```
80     if (!loginError)
81
82     @RequestMapping(value = "/register", method = RequestMethod.POST)
83     public String doRegister(@RequestHeader(value = "referer", required = false) final
84         final BindingResult bindingResult, final Model model, final HttpServletRequest
85         final HttpServletResponse response, final RedirectAttributes redirectModel) {
86
87         getRegistrationValidator().validate(form, bindingResult);
88         return processRegisterUserRequest(referer, form, bindingResult, model, request,
89     }
90 }
```

Note = When we click on Register button in **storefront**, if the flow coming here then it's the file & method.

```

96 @RequestMapping(value = "/register", method = RequestMethod.POST)
97 public String doRegister(@RequestHeader(value = "referer", required = false) final String referer, final RegisterForm form,
98     final BindingResult bindingResult, final Model model, final HttpServletRequest request,
99     final HttpServletResponse response, final RedirectAttributes redirectModel) throws CMSItemNotFoundException
100 {
101     getRegistrationValidator().validate(form, bindingResult);
102     return processRegisterUserRequest(referer, form, bindingResult, model, request, response, redirectModel);
103 }

104 @RequestMapping(value = "/register", method = RequestMethod.POST)
105 public String doRegister(@RequestHeader(value = "referer", required = false) final String referer, final RegisterForm form,
106     final BindingResult bindingResult, final Model model, final HttpServletRequest request,
107     final HttpServletResponse response, final RedirectAttributes redirectModel)
108 {
109     getRegistrationValidator().validate(form, bindingResult);
110     return processRegisterUserRequest(referer, form, bindingResult, model, request, response, redirectModel);
111 }
112 }
113 
```

Console Problems Debug Shell Search Debug Servers Variables

Note = Whatever we entered in UI layer is available in **RegisterForm Obj.**

Q = What exactly we are doing in controller?

Q1 = In PL, we are entering PWD & Confirm PWD. If these 2 are not same, then is it required for us to continue next steps? = No.

Q2 = In PL, we are entering email ID. If email ID does not have @ symbol, then is it required for us to continue next steps? = No.

===== In PL, we are entering lots of data. It's time to validate those.

Note: - For validation purpose – We have **Validator FWK**.

Validator FWK is having **validate ()** method.

We need to override **validate ()** method.

```

96 @RequestMapping(value = "/register", method = RequestMethod.POST)
97 public String doRegister(@RequestHeader(value = "referer", required = false) final String referer, final RegisterForm form,
98     final BindingResult bindingResult, final Model model, final HttpServletRequest request,
99     final HttpServletResponse response, final RedirectAttributes redirectModel)
100 {
101     getRegistrationValidator().validate(form, bindingResult);
102     return processRegisterUserRequest(referer, form, bindingResult, model, request, response, redirectModel);
103 }

104 @Override
105 public void validate(final Object object, final Errors errors)
106 {
107     final RegisterForm registerForm = (RegisterForm) object;
108     final String titleCode = registerForm.getTitleCode();
109     final String firstName = registerForm.getFirstName();
110     final String lastName = registerForm.getLastName();
111     final String email = registerForm.getEmail();
112     final String pwd = registerForm.getPassword();
113     final String checkPwd = registerForm.getCheckPassword();

114     validateTitleCode(errors, titleCode);
115     validateName(errors, firstName, "first");
116     validateName(errors, lastName, "last");
117 }
```

```

38    @Override
39    public void validate(final Object object, final Errors errors)
40    {
41        final RegisterForm registerForm = (RegisterForm) object;
42        final String lastName = registerForm.getLastName();
43        final String email = registerForm.getEmail(); // Line 45 highlighted in green
44        final String pwd = registerForm.
45        final String checkPwd = registerForm.
46
47        validateTitleCode(errors, titleCode);
48        validateName(errors, firstName);
49        validateName(errors, lastName, "lastName");
50
51        if (StringUtils.length(firstName) < 2)
52        {
53            errors.rejectValue("lastName", "mustNotBeEmpty");
54            errors.rejectValue("firstName", "mustNotBeEmpty");
55        }
56    }
57

```

RMB -- Inspect

- registerForm.getEmail() = "abcd@gmail.com"
 - hash=0
 - value= (id=2094)

You can see the value
abcd@gmail.com

Note = After validation is successful – Do below: -

Q3 = In PL, assume that you entered the **Confirm PWD**.

Are we really storing the Confirm PWD DB? = No.

Q4 = In PL, assume that you entered the **captcha**.

Are we storing the captcha in DB? = No.

===== In PL, we are entering lots of data & we may not store all the data in DB. So, it is not required to carry the data to next steps if not stored in DB.

So, create another Obj & put required into in this.

This Obj is called “**Data Obj**”.

```

@RequestMapping(value = "/register", method = RequestMethod.POST)
public String doRegister(@RequestHeader(value = "referer", required = false)
final BindingResult bindingResult, final Model model, final HttpServletRequest request, final HttpServletResponse response, final RedirectAttributes redirectAttributes)
{
    getRegistrationValidator().validate(form, bindingResult);
    return processRegisterUserRequest(referer, form, bindingResult, model);
}

```

1 Controller can call another Controller

```

133 */
134 protected String processRegisterUserRequest(final String referer, final RegisterForm form, final BindingResult bindingResult, final Model model, final HttpServletRequest request, final HttpServletResponse response, final RedirectAttributes redirectModel) throws CMSItemNotFoundException
135 {
136     if (bindingResult.hasErrors())
137     {
138         model.addAttribute(form);
139         model.addAttribute(new LoginForm());
140         model.addAttribute(new GuestForm());
141         GlobalMessages.addErrorMessage(model, FORM_GLOBAL_ERROR);
142         return handleRegistrationError(model);
143     }
144
145     final RegisterData data = new RegisterData();
146

```

form= RegisterForm (id=423)

- captcha= null
- checkPwd= "12341234" (id=428)
- email= "abcd" (id=432)
- firstName= "FName" (id=433)
- lastName= "LName" (id=434)
- mobileNumber= null
- pwd= "12341234" (id=436)
- titleCode= "mr" (id=437)

```
3④ * --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN!.
17 package de.hybris.platform.commercefacades.user.data;
18
19 import java.io.Serializable;
20
21 public class RegisterData implements Serializable
22 {
23
24     /** Default serialVersionUID value. */
25
26     private static final long serialVersionUID = 1L;
27
28     /** <i>Generated property</i> for <code>RegisterData.login</code> property defined at extension <code>com
29
30     private String login;
31
32     /** <i>Generated property</i> for <code>RegisterData.password</code> property defined at extension <codi
33
34     private String password;
35
36     /** <i>Generated property</i> for <code>RegisterData.titleCode</code> property defined at extension <co
37
38     private String titleCode;
39
40 }
```

Once Data Obj is ready then call the **Façade Layer**.

After Data Obj is ready = Call the FL = `getCustomerFacade().register(data);`

Q1 = What is the input for Controller?

= Form Obj

Q2 = What we did in Controller? =

- 1) Validations
- 2) Creating Data Obj

Q3 = What is the output of the Controller?

= Data Obj & Call Façade Layer.

The screenshot shows a Java code editor with three tabs: LoginPageController.java, RegistrationValidator.java, and AbstractRegisterPageController.java. The current tab is AbstractRegisterPageController.java. The code is as follows:

```

146     final RegisterData data = new RegisterData();
147     data.setFirstName(form.getFirstName());
148     data.setLastName(form.getLastName());
149     data.setLogin(form.getEmail());
150     data.setPassword(form.getPwd());
151     data.setTitleCode(form.getTitleCode());
152
153     try
154     {
155         getCustomerFacade().register(data);
156         getAutoLoginStrategy().login();
157
158         GlobalMessages.addFlashMessage("registration.confirm");
159     }
160     catch (final DuplicateUidException e)
161     {
162         LOGGER.warn("registration failed due to duplicate uid");
163     }

```

A tooltip is displayed over the line `getCustomerFacade().register(data);`, showing the details of the `data` object:

- `data= RegisterData (id=2042)`
- `firstName= "FName" (id=2045)`
- `lastName= "LName" (id=2046)`
- `login= "abcd@gmail.com" (id=2047)`
- `password= "12341234" (id=2048)`
- `titleCode= "mr" (id=2049)`

Q = How to call the Façade Layer?

= **getCustomerFacade().register(data);**

General Syntax =

getXXXFacade().method(==);

XXX = Customer / Order / Cart /

Step 3 = Façade Layer: -

The screenshot shows a Java code editor with four tabs: LoginPageController.java, RegistrationValidator.java, AbstractRegisterPageController.java, and *DefaultCustomerFacade.java. The current tab is *DefaultCustomerFacade.java. The code is as follows:

```

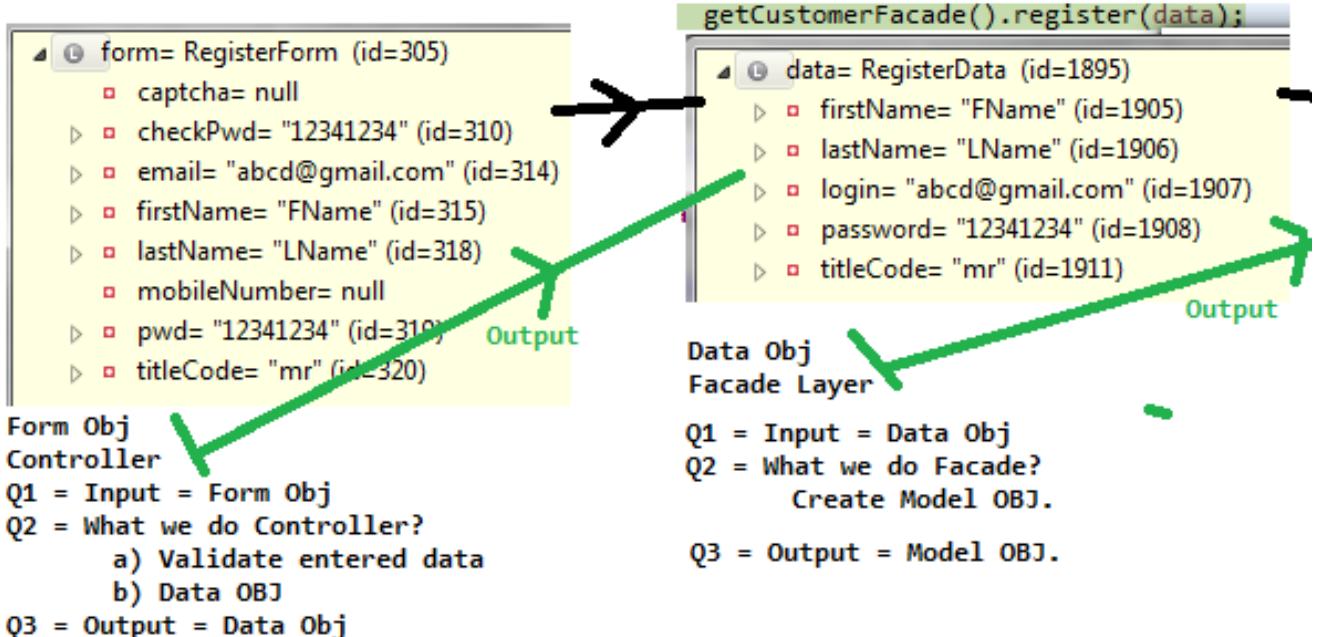
119
120     @Override
121     public void register(final RegisterData registerData) throws DuplicateUidException
122     {
123         validateParameterNotNullStandardMessage(registerData);
124         Assert.hasText(registerData.getFirstName());
125         Assert.hasText(registerData.getLastName());
126         Assert.hasText(registerData.getLogin());
127
128         try
129         {
130             de.hybris.platform.commercefacades.user.data.RegisterData@781c5ad1
131         }
132         catch (final DuplicateUidException e)
133         {
134             throw e;
135         }
136     }

```

A tooltip is displayed over the line `registerData= RegisterData (id=2042)`, showing the details of the `registerData` object:

- `registerData= RegisterData (id=2042)`
- `firstName= "FName" (id=2045)`
- `lastName= "LName" (id=2046)`
- `login= "abcd@gmail.com" (id=2047)`
- `password= "12341234" (id=2048)`
- `titleCode= "mr" (id=2049)`

Contact Us = ChennaReddyTraining@RRRS.CO.IN



ID:	abcd@gmail.com	Name:	FName LName
Customer ID:	58e00a9f-42cc-492d-85a5-d592cda71b50		
Original UID:	abcd@gmail.com		
Standard Language:	English		
Standard Currency:	GBP		

Q5 = In Data Obj, we have First Name & Last Name. But in DB, we only have Name & it's concatenation of 1st name & last name.

Q6 = In Data Obj, we don't have Lang ... Currency But in DB, we have Lang & Currency and

==> That means, Ur Data Obj is not TRUE representation of DB table customer record.

Now, it's time to create another Obj called "**Model Obj**", which will be TRUE representation of your DB table customer record.

For Customer → We have **CustomerModel**

For Order → We have **OrderModel**

→ General Syntax == For **XXX** item type ... We will have **XXXModel**

XXX = Customer / Order / Cart / Address /

Note: - Once Model Obj is ready then call the **Service Layer**.

The screenshot shows an IDE interface with two main panes. On the left, there is a code editor with Java code related to customer account registration. On the right, there is a debugger view showing a stack trace and variable information for a CustomerModel object.

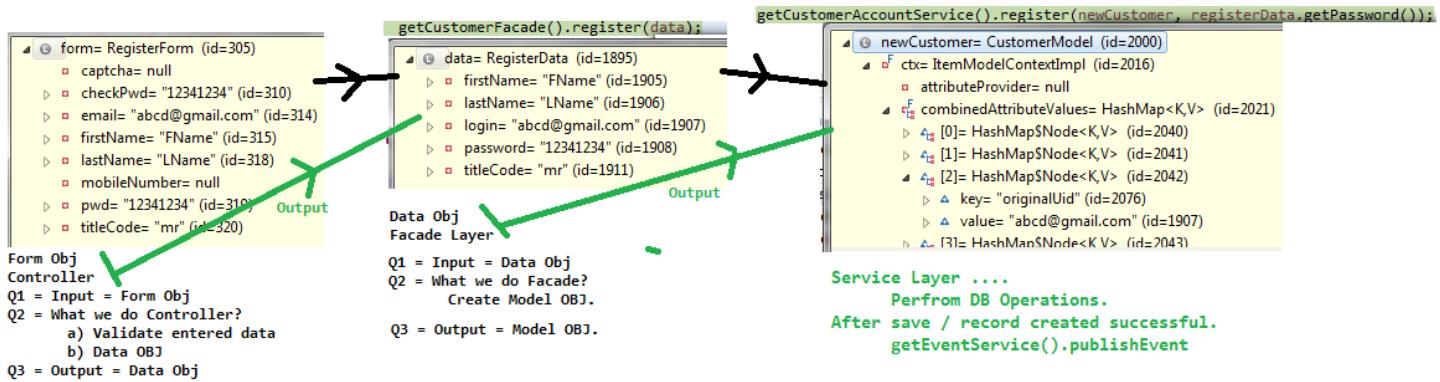
```
.37     setCustomerOrGuestUser(registerData, newCustomer);
.38     newCustomer.setSessionLanguage(getCommonI18NService().getCurrentLanguage());
.39     newCustomer.setSessionCurrency(getCommonI18NService().getCurrentCurrency());
.40     getCustomerAccountService().register(newCustomer, registerData.getPassword());
.41 }
.42
.43 @Override
.44 public void createGuestUserForAnonymous()
.45 {
.46     validateParameterNotNullStandardMessage();
.47     final CustomerModel guestCustomer =
.48     final String guid = generateGUID();
.49
.50     //takes care of localizing the name
.51 }
```

Stack Trace (from bottom to top):

```
uid=abcd@gmail.com, passwordEncoding=*, ldapaccount=false, name=FName LName, groups=
```

Variables (CustomerModel object):

- newCustomer= CustomerModel (id=2277)
- ctx= ItemModelContextImpl (id=2285)
 - attributeProvider= null
 - combinedAttributeValues= HashMap<K,V> (id=2312)
 - combinedLocalizedAttributeValues= HashMap<K,V> (id=2314)
 - dynamicAttributesProvider= DefaultDynamicAttributesProvider (id=2315)
 - fetchStrategy= DefaultFetchStrategy (id=2319)
 - itemType= "Customer" (id=2260)
 - locProvider= DefaultLocaleProvider (id=2322)
 - newPK= null
- uid=abcd@gmail.com, passwordEncoding=*, ldapaccount=false, name=FName LName, groups=



Q1 = What is the input for Façade Layer? = Data Obj

Q2 = What we did in Façade Layer? = Creating the Model Obj

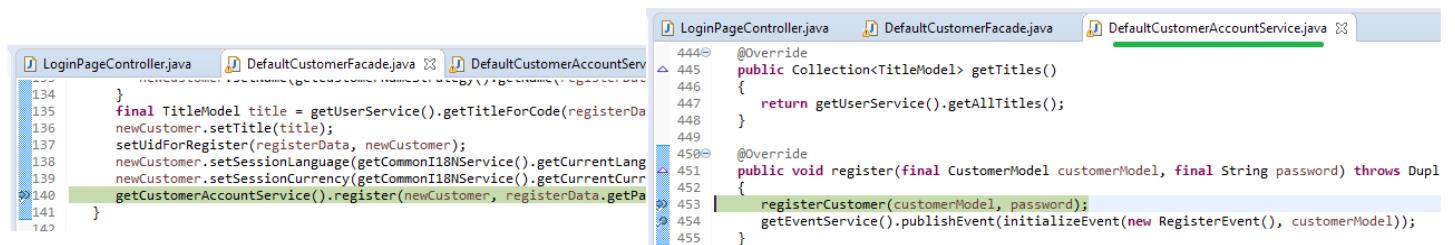
Q3 = What is Façade Layer output?

= Model Obj & also calling Service Layer.

Q4 = How to call the Service Layer? =

```
getCustomerAccountService().register(newCustomer,  
registerData.getPassword());
```

Step 4 = Service Layer: -



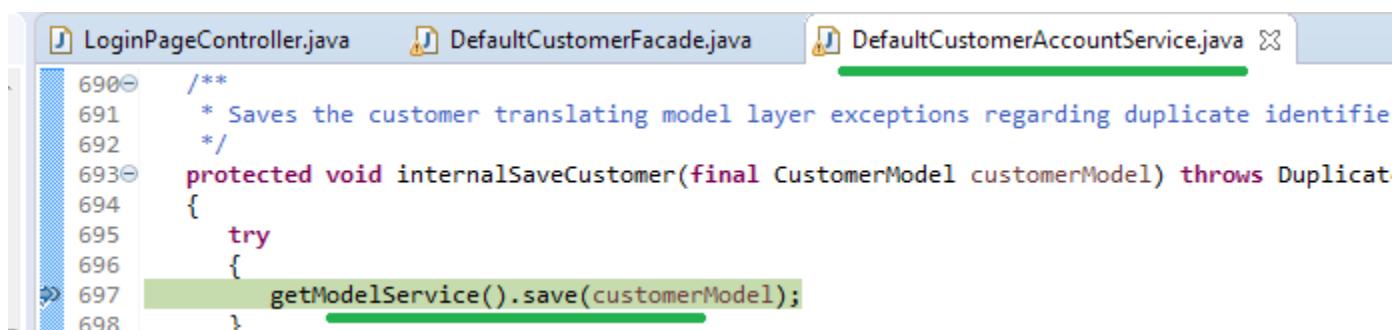
```
444    @Override  
445    public Collection<TitleModel> getTitles()  
446    {  
447        return getUserService().getAllTitles();  
448    }  
449  
450    @Override  
451    public void register(final CustomerModel customerModel, final String password) throws DuplicateCustomerException  
452    {  
453        registerCustomer(customerModel, password);  
454        getEventService().publishEvent(initializeEvent(new RegisterEvent(), customerModel));  
455    }  
456
```

General Syntax: -

For Cart → We have *Cart*Service.java

For Customer → We have *Customer*Service.java

For Order → We have *Order*Service.java



```
690    /**  
691     * Saves the customer translating model layer exceptions regarding duplicate identification  
692     */  
693    protected void internalSaveCustomer(final CustomerModel customerModel) throws DuplicateCustomerException  
694    {  
695        try  
696        {  
697            getModelService().save(customerModel);  
698        }
```

Q = What is the general Syntax for creating the Record?

getModelService().save(**XXXModel**);

XXX = Customer / Cart / Address / Order /

The screenshot shows a software application's user interface. On the left is a navigation tree with the following structure:

- Multimedia
- User
 - Companies
 - User Groups
 - Employees
 - Customers**
 - Addresses
 - Titles
 - Agreements
- Order
- Price Settings
- Internationalization
- Marketing
- Cockpit
- Scripting

The "Customers" node is highlighted with a green checkmark.

The main area contains a "Customer" detail form with the following fields:

ID:	abcd@gmail.com	Name:	FName LName
Customer ID:	daea3940-594e-43d2-993b-ad72825c66f0		
Original UID:	abcd@gmail.com		
Properties			
Description:			
Standard Language:	English	Standard Currency:	GBP

Q = What happen if getModelService().save() – Fails to perform the operation?

ModelSavingException

Q = What happen if getModelService().save() – Success to perform the operation?

Record will be inserted in the DB.

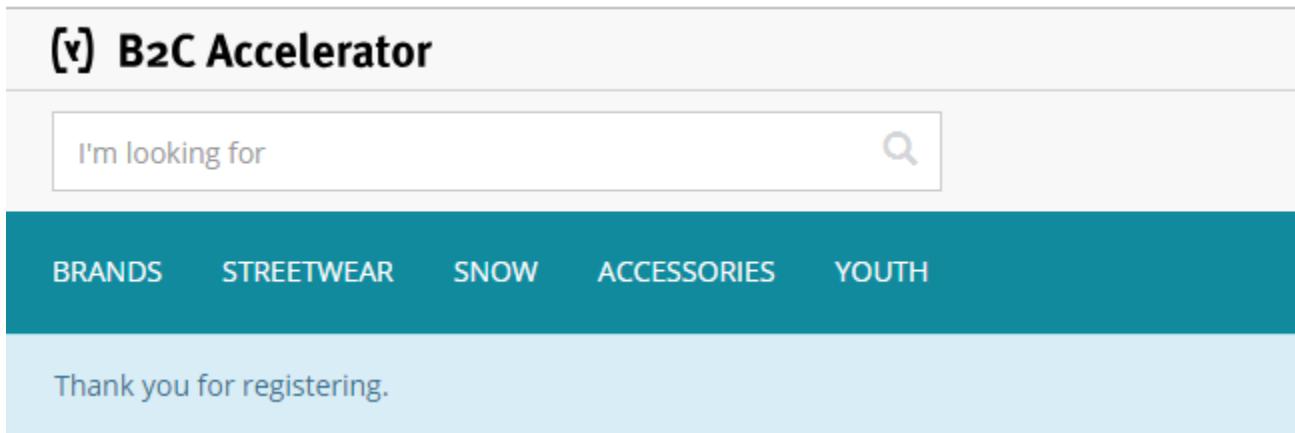
An event will be published stating that – Record insert is successful.

```

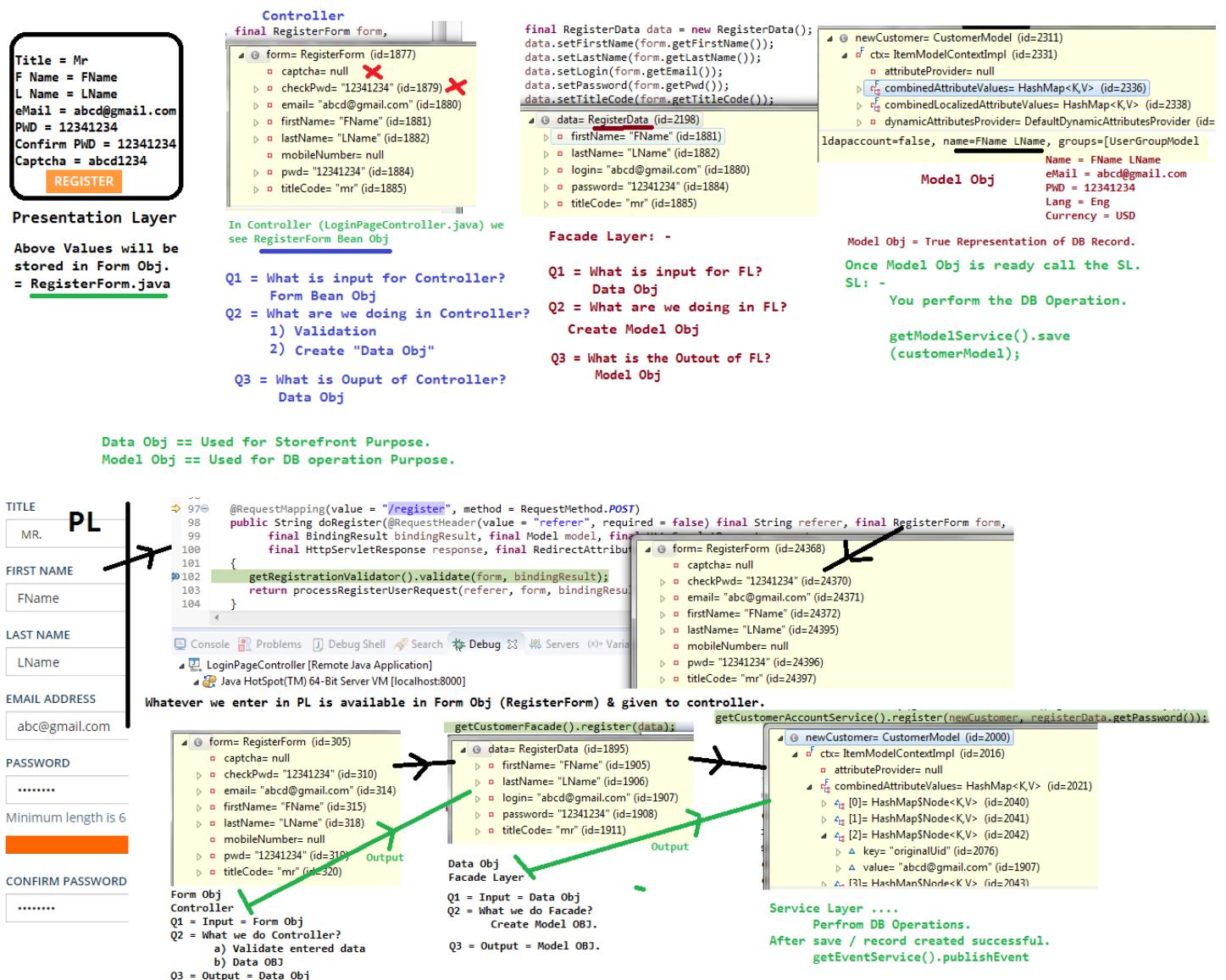
LoginPageController.java           DefaultCustomerFacade.java           DefaultCustomerAccountService.java
444@Override
445  public Collection<TitleModel> getTitles()
446  {
447      return getUserService().getAllTitles();
448  }
449
450@Override
451  public void register(final CustomerModel customerModel, final String password) throws DuplicateUidException
452  {
453      registerCustomer(customerModel, password);
454      getEventService().publishEvent(initializeEvent(new RegisterEvent(), customerModel));
455  }

```

Results in Storefront: -



Overral Flow:-

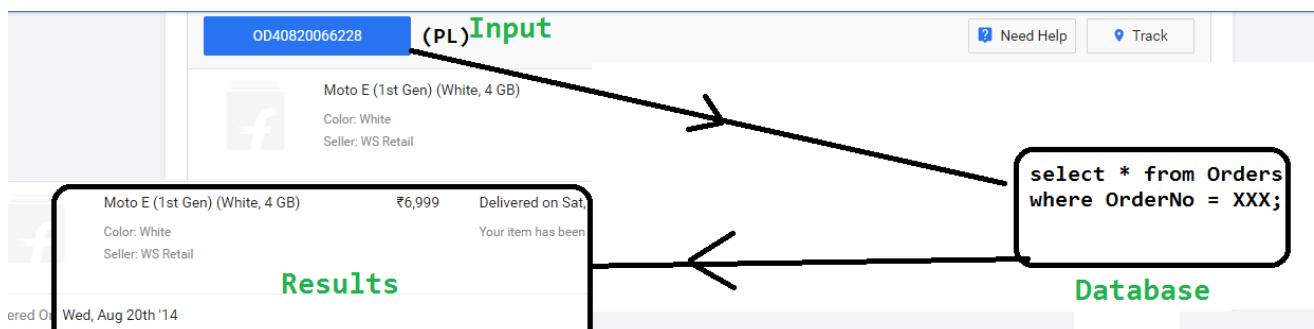


Contact Us = ChennaReddyTraining@RRRS.CO.IN

2nd Flow = Order Status (Enter Order No & Display the status)

= Bi-Directional Flow

PL (JSP file / Tag file / HTML file / JS file / CSS file /) → Controller → Façade Layer (FL) → Service Layer (SL) → DAO Layer (You Write Flexible Search Query) → Service Layer → Façade Layer → Controller → Presentation Layer (PL) & Display the results.



Q: How to place the Order from Backoffice / hmc?

The screenshot shows the SAP Backoffice (hmc) interface for placing an order. The left sidebar navigation tree includes: Multimedia, User, Order (selected), Order Entries, Order History Entry, Consignment, Consignment Entry, Fraud reports, B2B Booking Entry, Price Settings, Internationalization, Marketing, Cockpit, Scripting, WCMS, Ticket System, Base Commerce, Deeplink URLs, B2B Commerce, and B2B Approval Process. The main screen displays the 'Order' creation form. The top bar includes Save, Reload, Copy, Delete, Calculate with promotions, Recalculate completely, and Recalculate totals. The 'Positions and Prices' tab is active. The form fields include:

- User: abcd@gmail.com
- Order Nr.: 00002000
- Currency: GBP
- Date: 03/12/2019 12:00:00 AM
- Pricing: Gross
- Calculation is up to date: Yes
- Site: Apparel Site UK
- Store: Apparel Store UK
- Sales Application:
- Language:

The 'Positions' section shows three entries:

Product	Quantity	Unit	Base Price	Total Price
300441355 - Beacon Jacket glacier M - (ap)	3	pieces	101.21	303.63
300460184 - Last Mission Jacket snapper L	4	pieces	64.76	259.04
300392200 - Boardshort youth summer M -	5	pieces	36.41	182.05

At the bottom, there is a 'Global Discounts' section.

The second part of the screenshot shows the 'Payment and Delivery' tab selected. The form fields are identical to the first screenshot, with the addition of the 'Status' section:

Orderstatus:	Open
Export Status:	
Status Info:	

Order History | Apparel Site UK Explorer: Administrator@localhost

Not secure | https://localhost:9002/rrrstrainingstorefront/en/my-account/orders

B2C Accelerator

I'm looking for

WELCOME FNAME | MY ACCOUNT | SIGN OUT

BRANDS STREETWEAR SNOW ACCESSORIES YOUTH

HOME / ORDER HISTORY

1 Order

ORDER NUMBER	ORDER STATUS	DATE PLACED	TOTAL
00002000	Open	Mar 12, 2019 12:00 AM	£744.72

Order Details | Apparel Site UK

Not secure | https://localhost:9002/rrrstrainingstorefront/en/my-account/order/00002000

SORT BY: DATE

1 Order

ORDER NUMBER	ORDER STATUS
00002000	Open

FSQ

DB

ITEM (STYLE NUMBER)	PRICE
Beacon Jacket glacier M 300441355 In Stock	£101.21
Last Mission Jacket snapper L 300460184 In Stock	£64.76

Step 1 = How to identify the PL?

Order History | Apparel Site UK

Not secure | https://localhost:9002/rrrstrainingstorefront/en/my-account/orders

Your Company Req -- Display Order Discount if any

1 Order

ORDER NUMBER	ORDER STATUS	DATE PLACED	RRRSDiscount	TOTAL
00002000	Open	Mar 12, 2019 12:00 AM	18%	£744.72

Contact Us = ChennaReddyTraining@RRRS.CO.IN

base_en.properties accountOrderHistoryPage.jsp

```

29
30         <div class="account-order-history-paginat&gt;
31             <nav:pagination top="true" msgKey="text.account.orderHistory.page" showCurrentPageInfo="true" />
32         </div>
33     <div class="account-overview-table">
34         <table class="orderhistory-list-table responsive-table">
35             <tr class="account-orderhistory-table-head responsive-table-head hidden-xs">
36                 <th><spring:theme code="text.account.orderHistory.orderNumber" /></th>
37                 <th><spring:theme code="text.account.orderHistory.orderStatus" /></th>
38                 <th><spring:theme code="text.account.orderHistory.datePlaced" /></th>
39                 <th><spring:theme code="text.account.orderHistory.RRRSDiscoun&gt;
40                 <th><spring:theme code="text.account.orderHistory.total" /></th>
41             </tr>
42             <c:forEach items="${searchPageData.results}" var="order">
43                 <tr class="responsive-table-item">

```

base_en.properties accountOrderHistoryPage.jsp

1020 text.account.order.qty	= Qty
1021 text.account.order.warning.storePickUpItems	= Reminder - Please
1022 text.account.order.yourOrder	= Your Order
1023 text.account.order.totalSavings	= You saved {0} on
1024 text.account.order.back.btn	= Back
1025 text.account.orderHistory	= Order History
1026 text.account.orderHistory.back	= Back
1027 text.account.orderHistory.actions	= Actions
1028 text.account.orderHistory.datePlaced	= Date Placed
1029 text.account.orderHistory.RRRSDiscoun>	= RRRS Discount
1030 text.account.orderHistory.mobile.page.currentPage	= {0} of {1}

base_en.properties accountOrderHistoryPage.jsp accountMyQuotesPage.jsp

```

48
49
50         </td>
51         <td class="hidden-sm hidden-md hidden-lg"><spring:theme code="text.account.orderHistory.orderStatus" /></td>
52         <td class="status">
53             <spring:theme code="text.account.order.status.display.${order.statusDisplay}" />
54         </td>
55         <td class="hidden-sm hidden-md hidden-lg"><spring:theme code="text.account.orderHistory.datePlaced" /></td>
56         <td class="responsive-table-cell">
57             <fmt:formatDate value="${order.placed}" dateStyle="medium" timeStyle="short" type="both" />
58         </td>
59         <td class="responsive-table-cell"> 18%
60
61
62         <td class="hidden-sm hidden-md hidden-lg"><spring:theme code="text.account.orderHistory.total" /></td>
63         <td class="responsive-table-cell responsive-table-cell-bold">
64             ${fn:escapeXml(order.total.formattedValue)}
65         </td>
66     </vcommerce:testId>

```

Step 2 = Find out the Controller.

Once we click on the “Order No”, the corresponding controller will be called.

The screenshot shows a table with columns: ORDER NUMBER, ORDER STATUS, DATE PLACED, and RRRS DISCOUNT. One row is highlighted with a grey background, showing '00002000' in the ORDER NUMBER column, 'Open' in ORDER STATUS, 'Mar 12, 2019 12:00 AM' in DATE PLACED, and '18%' in RRRS DISCOUNT. A green arrow points from the text 'When you Hover Order No' to the '00002000' link. Below the table, the browser's developer tools are open, specifically the Elements tab, showing the HTML structure of the page. The URL in the address bar is https://localhost:9002/rrrstrainingstorefront/en/my-account/order/00002000. The DOM tree shows a table with a tbody containing tr and td elements, and a specific td element for the order number contains an anchor tag with href="/rrrstrainingstorefront/en/my-account/order/00002000".

Q: What are the different OOTB functionality are there in “My-Account” page?

The screenshot shows a web page titled 'B2C Accelerator'. At the top, there is a navigation bar with links for 'Order History | Apparel Site UK', 'Not secure | https://localhost:9002/rrrstrainingstorefront/en/my-account/orders', 'WELCOME FNAME', 'MY ACCOUNT' (which is highlighted with a green box), and 'SIGN OUT'. Below the navigation bar, there is a grid of links: 'Personal Details' (under 'Personal Details'), 'Email Address' (under 'Email Address'), 'Payment Details' (under 'Payment Details'); 'Password' (under 'Password'), 'Order History' (under 'Order History'), 'Address Book' (under 'Address Book'); and 'Saved Carts' (under 'Saved Carts'), 'Support Tickets' (under 'Support Tickets'), 'Returns History' (under 'Returns History').

/order/ = Find mapping at method level. & /my-account = Find mapping at class level.

```

1 AccountPageController.java
2+ * [y] hybris Platform
3 package com.rrrs.rrrstraining.storefront.controllers.pages;
4
5 import de.hybris.platform.acceleratorfacades.ordergridform.OrderGridFormFacade;
6
7 @Controller
8 @RequestMapping("/my-account")
9 public class AccountPageController extends AbstractSearchPageController
10 {
11     private static final String TEXT_ACCOUNT_ADDRESS_BOOK = "text.account.addressBook";
12     private static final String BREADCRUMBS_ATTR = "breadcrumbs";
13
14     @RequestMapping(value = "/order/" + ORDER_CODE_PATH_VARIABLE_PATTERN, method = RequestMethod.GET)
15     @RequireHardLogIn
16     public String order(@PathVariable("orderCode") final String orderCode, final Model model,
17                         final RedirectAttributes redirectModel) throws CMSItemNotFoundException
18     {
19         try
20         {
21             final OrderData orderDetails = orderFacade.getOrderDetailsForCode(orderCode);
22             model.addAttribute("orderData", orderDetails);
23
24         }
25     }
26
27 }

```

In Controller – What did we do yesterday?

- 1) We did the validation

Q: Do we need the validation for current scenario? = No need.

Yesterday (**Registration**) – Means, User is entering the data. So, there is possibility of user can enter wrong data. Hence did the validation.

Today (**Order Status**) – Means, User just click on Order No (User is not entering the order no), so no need the validation.

- 2) We have created the Data Obj

Step 3 = Façade Layer: -

```

1 @RequestMapping(value = "/order/" + ORDER_CODE_PATH_VARIABLE_PATTERN, method = RequestMethod.GET)
2 @RequireHardLogIn
3 public String order(@PathVariable("orderCode") final String orderCode, final Model model,
4                     final RedirectAttributes redirectModel) throws CMSItemNotFoundException
5 {
6     try
7     {
8         final OrderData orderDetails = orderFacade.getOrderDetailsForCode(orderCode);
9         model.addAttribute("orderData", orderDetails);
10
11         final List<Breadcrumb> breadcrumbs = accountBr
12         breadcrumbs.add(new Breadcrumb("/my-account/or
13             getI18nService().getCurrentLocale(), nu
14
15
16
17
18

```

For Cart – We have cartFacade

For Order – We have orderFacade

For User – We have userFacade

The screenshot shows a Java code editor with three tabs: AccountPageController.java, DefaultOrderFacade.java, and DefaultCustomerAccountService.java. The code in AccountPageController.java is:

```
49
50    @Override
51    public OrderData getOrderDetailsForCode(final String code)
52    {
53        final BaseStoreModel baseStoreModel = getBaseSto
54
55        OrderModel orderModel = null;
56        if (getCheckoutCustomerStrategy().isAnonymousCheckou
57        {
58            orderModel = getCustomerAccountService().getOrde
59        }
60        else
61        {
62            try
63            {
64                orderModel = getCustomerAccountService().getOrderForCode((CustomerModel) getUserService().getCurrentUser(), code,
65                baseStoreModel);
66            }
67        }
68    }
```

A code completion tooltip for 'getOrderForCode' is open, showing the method signature and parameters:

```
code= "00002000" (id=2310)
  hash= 0
  value= (id=2311)
```

The parameter 'code' is highlighted with a green arrow.

Step 4 = Service Layer

The screenshot shows a Java code editor with three tabs: AccountPageController.java, DefaultOrderFacade.java, and DefaultCustomerAccountService.java. The code in DefaultCustomerAccountService.java is:

```
548
549
550    /**
551     * @deprecated Since 5.0.
552     */
553    @Override
554    @Deprecated
555    public OrderModel getOrderForCode(final CustomerModel customerModel, final String code, final BaseStoreModel store)
556    {
557        validateParameterNotNull(customerModel, "Customer model cannot be null");
558        validateParameterNotNull(code, "Order code cannot be null");
559        validateParameterNotNull(store, "Store must not be null");
560        return getCustomerAccountDao().findOrderByCustomerAndCodeAndStore(customerModel, code, store);
561    }
```

A code completion tooltip for 'getCustomerAccountDao().findOrderByCustomerAndCodeAndStore' is open, showing the method signature and parameters:

```
code= "00002000" (id=2310)
  hash= 0
  value= (id=2311)
```

The parameter 'code' is highlighted with a green arrow.

Step 5 = DAO Layer

In DAO Layer – We write Flexible Search Queries

Oracle --- Select == “SAP Comm” – FSQ

```
116
117  @Override
118  public OrderModel findOrderByCustomerAndCodeAndStore(final CustomerModel customerModel, final String code,
119          final BaseStoreModel store)
120  {
121      validateParameterNotNull(customerModel, "Customer must not be null");
122      validateParameterNotNull(code, "Code must not be null");
123      validateParameterNotNull(store, "Store must not be null");
124      final Map<String, Object> queryParams = new HashMap<String, Object>();
125      queryParams.put("customer", customerModel);
126      queryParams.put("code", code);
127      queryParams.put("store", store);
128      final OrderModel result = getFlexibleSearchService().searchUnique(
129          new FlexibleSearchQuery(FIND_ORDERS_BY_CUSTOMER_CODE_STORE_QUERY, queryParams));
130      return result;
131  }
```

Q: How to execute the Flexible Search Queries? = FlexibleSearchService

Note =

For **DB interaction purpose** – We have **Model Obj.**

For **Storefront interaction purpose** – We have **Data Obj.**

General Syntax is →

If itemtype = **XXX** then we have **XXXModel** & **XXXData**

For Cart – We have CartModel & CartData

For Order – We have OrderModel & OrderData =====

Note = Above Results (OrderModel) – Return to Service Layer ... Service Layer returns to Façade Layer...

Step 6 = Once you get the results into Façade Layer...

Now it's time to convert **OrderModel** into **OrderData**.

That means, General Syntax = **XXXModel** into **XXXData**.

Q = How to convert Model Obj into Data Obj? = We can use **Converters.**

`getOrderConverter().convert(orderModel);`

General Syntax = `getXXXConverter().convert(xxxModel);`

Note = Converter internally calls the **populator.**

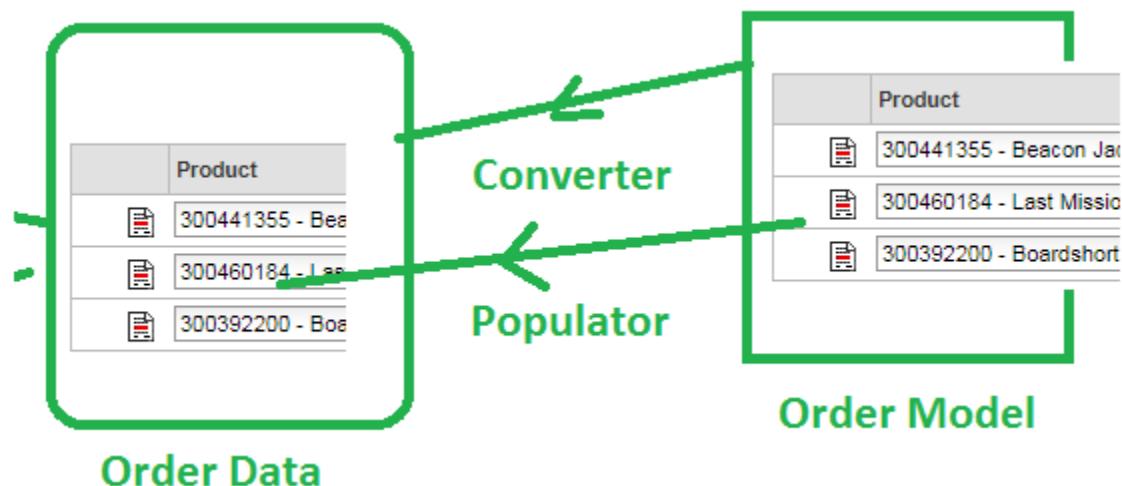
Q = What is the purpose of the Populator?

Populator **populate** the data from **Model obj** into **Data Obj**.

That means, Populator takes data from Model Obj & Put in Data Obj.

Populator – Will have **Populate()** method.

```
AccountPageController.java
69         throw new UnknownIdentifierException(String.format("Unknown identifier %s", identifier));
70     }
71 }
72 if (orderModel == null)
73 {
74     throw new UnknownIdentifierException(String.format("Unknown identifier %s", identifier));
75 }
76 return getOrderConverter().convert(orderModel);
77 }
78 }
```



```

150
151@ Override
152 public void populate(final AbstractOrderEntryModel source, final OrderEntryData target)
153 {
154     Assert.notNull(source, "Parameter source cannot be null.");
155     Assert.notNull(target, "Parameter target cannot be null.");
156
157     addCommon(source, target);
158     addProduct(source, target);
159     addTotals(source, target); green arrow here
160     addDeliveryMode(source, target);
161     addConfigurations(source, target);
162     addEntryGroups(source, target);
163     addComments(source, target);
164 }

```



```

226
227protected void addProduct(final AbstractOrderEntryModel orderEntry, final OrderEntryData entry)
228{
229    entry.setProduct(getProductConverter().convert(orderEntry.getProduct()));
230}
231
232protected void addTotals(final AbstractOrderEntryModel orderEntry, final OrderEntryData entry)
233{
234    if (orderEntry.getBasePrice() != null)
235    {
236        entry.setBasePrice(createPrice(orderEntry, orderEntry.getBasePrice())); green arrow here
237    }
238    if (orderEntry.getTotalPrice() != null)
239    {
240        entry.setTotalPrice(createPrice(orderEntry, orderEntry.
241    }
242}
243
244protected void addComments(final AbstractOrderEntryModel orde

```

So – Data Obj = **OrderEntryData**

Step 7 = Once – Convertor & Populator are executed, your will flow return to Controller.

```

306@ RequestMapping(value = "/order/" + ORDER_CODE_PATH_VARIABLE_PATTERN, method = RequestMethod
307 @RequireHardLogIn
308 public String order(@PathVariable("orderCode") final String orderCode, final Model model,
309     final RedirectAttributes redirectModel) throws CMSItemNotFoundException
310 {
311     try
312     {
313         final OrderData orderDetails = orderFacade.getOrderDetailsForCode(orderCode);
314         model.addAttribute("orderData", orderDetails); green arrow here
315     }

```

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

Step 8 = Now Use the “OrderData and OrderEntryData” Objects data UI in Layer & display the results.

ITEM (STYLE NUMBER)	PRICE
 Beacon Jacket glacier M 300441355 In Stock	£101.21
 Last Mission Jacket snapper L 300460184 In Stock	£64.76

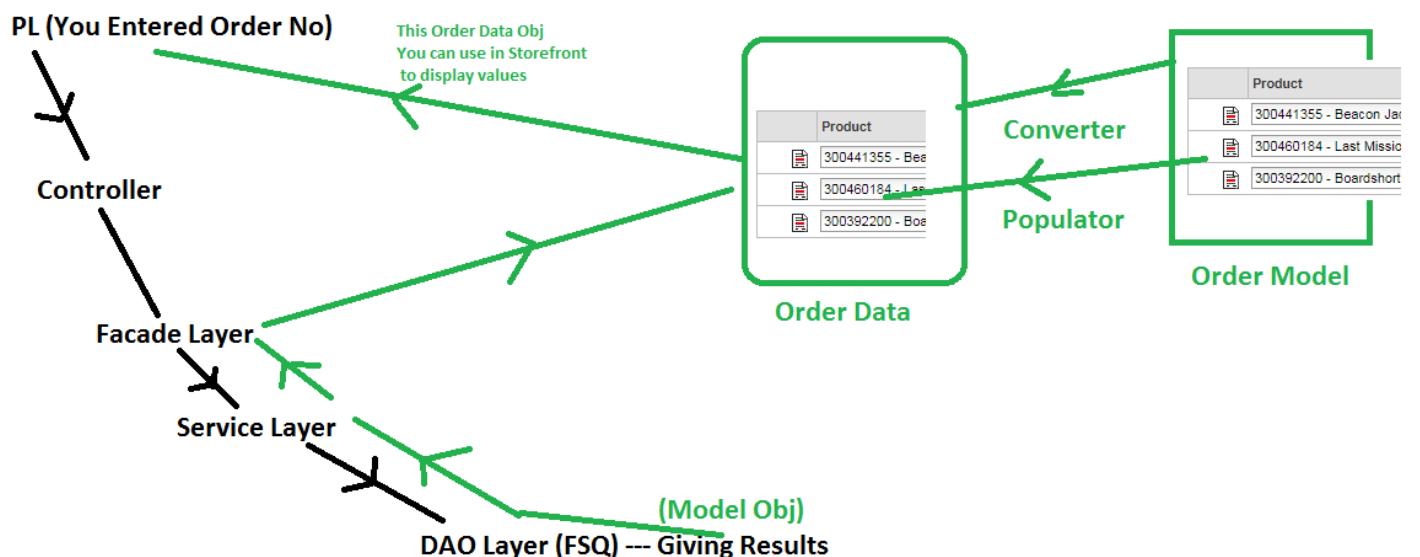
OrderEntryData

Note: - Let's verify how Data Object (**OrderEntryData**) is used in UI Layer?

```

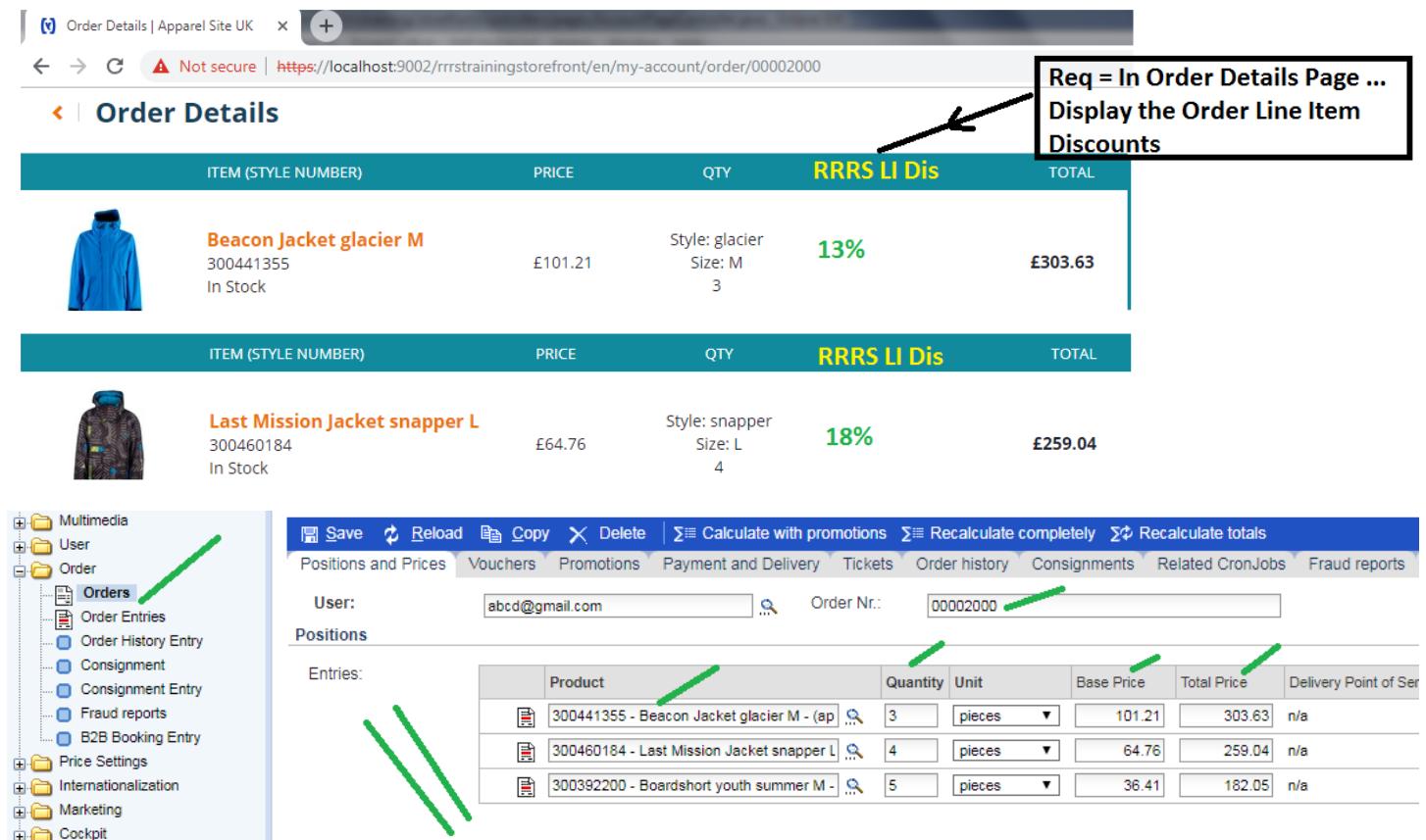
1 <%@ tag body-content="empty" trimDirectiveWhitespaces="true" %>
2 <%@ attribute name="order" required="true" type="de.hybris.platform.commercefacades.order.data.AbstractOrderData" %>
3 <%@ attribute name="orderEntry" required="true" type="de.hybris.platform.commercefacades.order.data.OrderEntryData" %>
4 <%@ attribute name="consignmentEntry" required="false"
   type="de.hybris.platform.commercefacades.order.data.ConsignmentEntryData" %>
...
109 <%-- price --%>
110<div class="item_price">
111   <span class="visible-xs visible-sm"><spring:theme code="basket.page.itemPrice"/></span>
112<ycommerce:TestId code="orderDetails_productItemPrice_label">
113   <order:orderEntryPrice orderEntry="${orderEntry}" />
114</ycommerce:TestId>
115</div>

```



Business Request = In Order Details Page, display “Order Line Item Discount (Example: - 13% ... 18% ...)” from DB field.

Also, display Order Line Item Label = **RRRS LI Dis.**



The screenshot shows the 'Order Details' page for order number 00002000. It displays two items:

- Beacon Jacket glacier M**: Price £101.21, Qty 3, Style: glacier, Size: M. The discount is labeled **13%** and the total is £303.63.
- Last Mission Jacket snapper L**: Price £64.76, Qty 4, Style: snapper, Size: L. The discount is labeled **18%** and the total is £259.04.

A callout box with an arrow points to the top right of the table, containing the text: "Req = In Order Details Page ... Display the Order Line Item Discounts".

Left Sidebar: Shows a navigation tree with categories like Multimedia, User, Order, Orders, Order Entries, Order History Entry, Consignment, Consignment Entry, Fraud reports, B2B Booking Entry, Price Settings, Internationalization, Marketing, and Cockpit. A green arrow points from the 'Orders' link in the sidebar to the 'Entries' section of the main table.

Table Headers: The table has columns: ITEM (STYLE NUMBER), PRICE, QTY, RRRS LI Dis, and TOTAL.

Table Data:

ITEM (STYLE NUMBER)	PRICE	QTY	RRRS LI Dis	TOTAL
Beacon Jacket glacier M 300441355 In Stock	£101.21	3	Style: glacier Size: M 13%	£303.63
Last Mission Jacket snapper L 300460184 In Stock	£64.76	4	Style: snapper Size: L 18%	£259.04

Toolbar: Includes Save, Reload, Copy, Delete, Calculate with promotions, Recalculate completely, Recalculate totals, and links to Positions and Prices, Vouchers, Promotions, Payment and Delivery, Tickets, Order history, Consignments, Related CronJobs, and Fraud reports.

Database View: A screenshot of a database tool showing the structure of the 'ORDERS' table. The table has columns: COLUMN_NAME, TYPE_N..., IS_NULLA..., DECIMAL_DIGI..., and COLUMN. Several columns are highlighted with red arrows pointing to them: ORDERENTRIES (highlighted in blue), P_BASEPRICE, P_QUANTITY, and P_TOTALPRICE.

Step 1 = Create New Column called “RRRSLIDis”



```
core-items.xml
2527<itemtype code="AbstractOrderEntry"
2528     extends="GenericItem"
2529     jaloClass="de.hybris.platform.jalo.order.AbstractOrderEntry"
2530     abstract="true"
2531     autoCreate="true"
2532     generate="true">
2533<attributes>
2534     <attribute autocreate="true" qualifier="basePrice" type="java.lang.Double">
2535         <persistence type="property">
2536             <columnType>
2537                 <value>java.math.BigDecimal</value>
2538             </columnType>
2539         </persistence>
2540         <modifiers read="true" write="true" search="false" optional="true"/>
2541         <defaultValue>Double.valueOf(0.0d)</defaultValue>
2542     </attribute>
```

Q = Can we go & add new attribute directly in this “core-items.xml”? =
Yes, we can do. But this is **not best practice**.

Why this is not best practice? = “core-items.xml” file is “SAP Comm” provided & available in platform folder. If we do the changes here, it will work, but tomorrow if any upgrade happens then our **changes will be lost**.

Q = Where can we do the changes (creating new DB field)?



Photon - rrrtrainingcore/resources/rrrtrainingcore-items.xml - Eclipse IDE

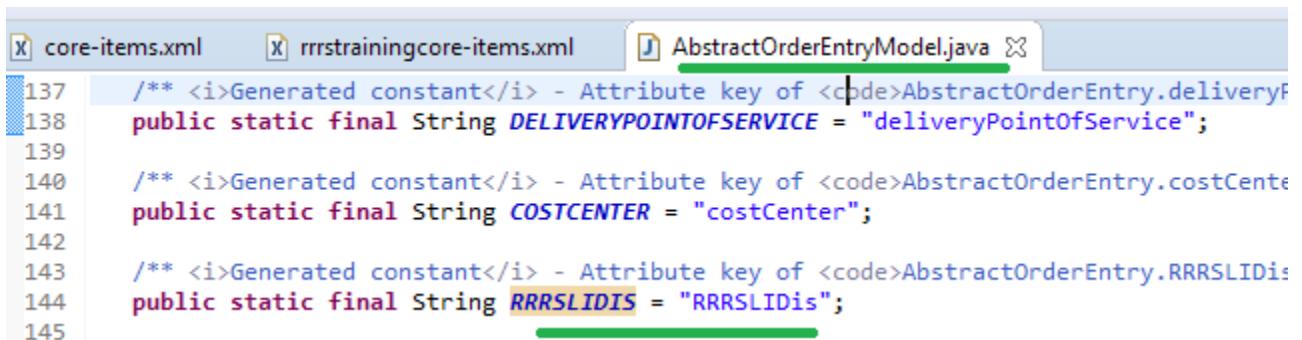
```
File Edit Navigate Search Project Run SAP Hybris [y] Hybris Design Window Help
```

Project Explorer

- rrrtrainingcockpits
- rrrtrainingcore
 - JRE System Library [jre1.8.0_191]
 - Referenced Libraries
 - resources
 - cockpitng.widgets.actions.pickupconfig
 - localization
 - rrrtrainingcore
 - rrrtrainingcore.integration
 - rrrtrainingcore.messages
 - rrrtrainingcore.processes
 - rrrtrainingcore.processes.quote
 - rrrtrainingcore.test
 - test
 - rrrtrainingcore-backoffice-config.xml
 - rrrtrainingcore-items.xml
 - rrrtrainingcore-spring.xml

Step 2 = Do the build (ant clean all).

After completion of build – If you open the Model class, it should your attribute.

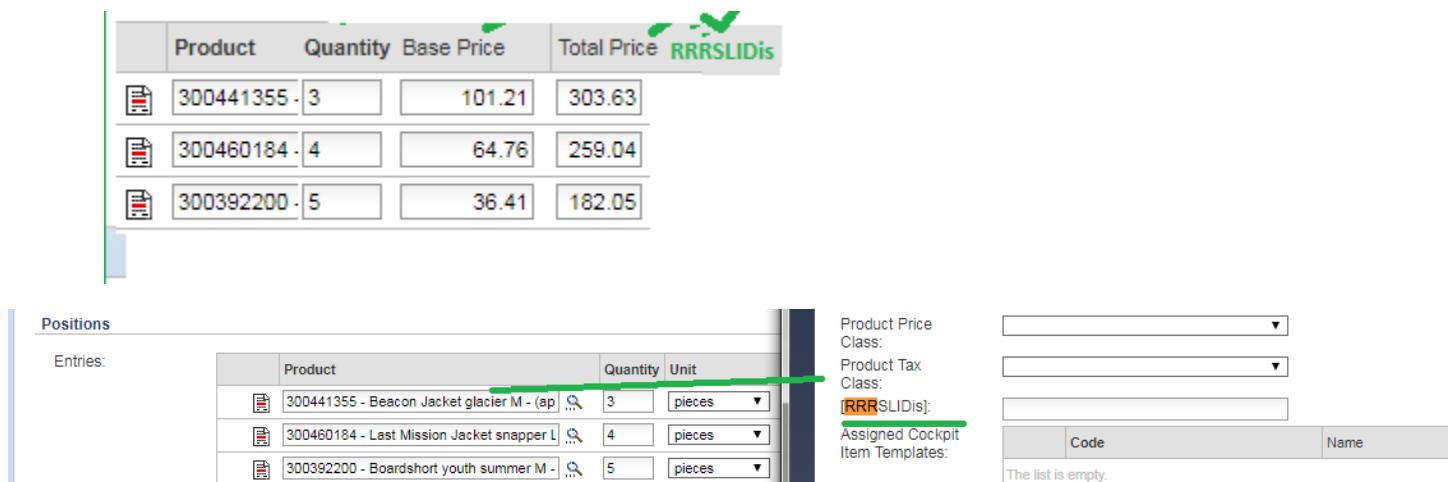


```
137  /** <i>Generated constant</i> - Attribute key of <code>AbstractOrderEntry.deliveryPointOfService</code>
138  public static final String DELIVERYPOINTOFSERVICE = "deliveryPointOfService";
139
140  /** <i>Generated constant</i> - Attribute key of <code>AbstractOrderEntry.costCenter</code>
141  public static final String COSTCENTER = "costCenter";
142
143  /** <i>Generated constant</i> - Attribute key of <code>AbstractOrderEntry.RRRSLIDis</code>
144  public static final String RRRSLIDIS = "RRRSLIDis";
145
```

Step 3 = Start the Server (hybrisserver.bat)

Step 4 = Perform the Platform – hAC Update

Step 5 = After the update successful – Check the results (New field created in DB).



The screenshot shows the Hybris Admin Console interface. At the top, there is a header with tabs: core-items.xml, rrrstrainingcore-items.xml, and AbstractOrderEntryModel.java. Below the header, there is a code editor window displaying Java code related to the AbstractOrderEntryModel class. In the main content area, there is a table representing a shopping cart with three items:

	Product	Quantity	Base Price	Total Price
	300441355	3	101.21	303.63
	300460184	4	64.76	259.04
	300392200	5	36.41	182.05

Below the table, there is a section titled "Positions" with a table titled "Entries". This table lists the three items from the shopping cart:

	Product	Quantity	Unit
	300441355 - Beacon Jacket glacier M - (ap)	3	pieces
	300460184 - Last Mission Jacket snapper L	4	pieces
	300392200 - Boardshort youth summer M	5	pieces

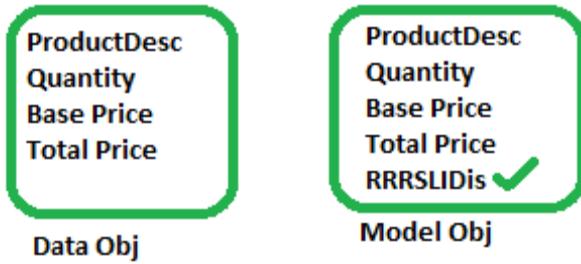
To the right of the "Entries" table, there are several configuration fields:

- Product Price Class: [dropdown]
- Product Tax Class: [dropdown]
- [RRRSLIDis]: [dropdown] (highlighted in green)
- Assigned Cockpit Item Templates: [list box]
The list is empty.

Note = RRRSLIDis values we need to display in Storefront.

For DB interaction purpose – We have **Model Obj** (or) Model Class.

For PL interaction purpose – We have **Data Obj** (or) Data Class.



Note = So, now it's time to create attribute in “**Data class / Data Obj**”.

If we want new attribute in **Model class (or) Model Obj** – then we need to do the changes in “***core-items.xml**”.

If we want new attribute in **Data Class (or) Data Obj** – then we need to do the changes in “***facades-beans.xml**”.

Step 6 = Create new attribute called “**ChennaLIDis**” in Data class (or) Data Obj.

The screenshot shows the Eclipse IDE interface with the title bar "Photon - commercefacades/resources/commercefacades-beans.xml - Eclipse IDE". The menu bar includes File, Edit, Source, 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 shows files like commercefacades-beans.xml, commercefacades-spring.xml, commercefacades.build.number, src, testsrc, gensrc, classes, eclipsebin, lib, resources, buildcallbacks.xml, extensioninfo.xml, external-dependencies.xml, project.properties, ruleset.xml, and commerceorqaddon. The central editor area shows the XML code for "AbstractOrderEntryModel.java". The code defines a bean for "OrderEntryData" with properties: placedBy (String), quoteCode (String), entryNumber (Integer), quantity (Long), basePrice (PriceData), totalPrice (PriceData), product (ProductData), updateable (boolean), deliveryMode (DeliveryModeData), deliveryPointOfService (PointOfServiceData), entries (List<OrderEntryData>), configurationInfos (ConfigurationInfoData), and statusSummaryMap (Map<ProductInfoStatus, Integer>).

```

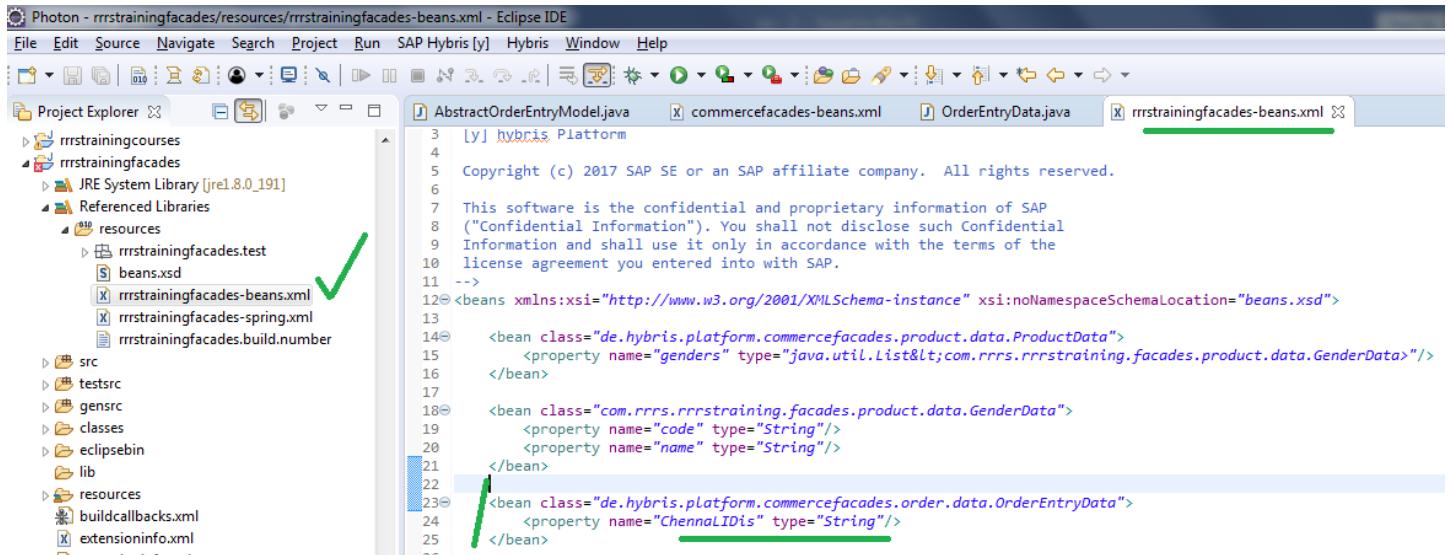
<property name="placedBy" type="String"/>
<property name="quoteCode" type="String"/>
</bean>
<bean class="de.hybris.platform.commercefacades.order.data.OrderEntryData">
<property name="entryNumber" type="Integer"/>
<property name="quantity" type="Long"/>
<property name="basePrice" type="de.hybris.platform.commercefacades.product.data.PriceData"/>
<property name="totalPrice" type="de.hybris.platform.commercefacades.product.data.PriceData"/>
<property name="product" type="de.hybris.platform.commercefacades.product.data.ProductData"/>
<property name="updateable" type="boolean"/>
<property name="deliveryMode" type="de.hybris.platform.commercefacades.order.data.DeliveryModeData"/>
<property name="deliveryPointOfService"
          type="de.hybris.platform.commercefacades.storelocator.data.PointOfServiceData"/>
<property name="entries"
          type="java.util.List<de.hybris.platform.commercefacades.order.data.OrderEntryData>"/>
<property name="configurationInfos"
          type="java.util.List<de.hybris.platform.commercefacades.order.data.ConfigurationInfoData>"/>
<property name="statusSummaryMap" type="java.util.Map<de.hybris.platform.catalog.enums.ProductInfoStatus, Integer>"/>

```

Q = Can we go & add new attribute directly in this “**commercefacades-beans.xml**”? = Yes, we can do. But this is **not best practice**.

Why this is not best practice? = “**commercefacades-beans.xml**” file is “SAP Comm” provided. If we do the changes here, it will work, but tomorrow if any upgrade happens then our **changes will be lost**.

Q = Where can we do the changes?



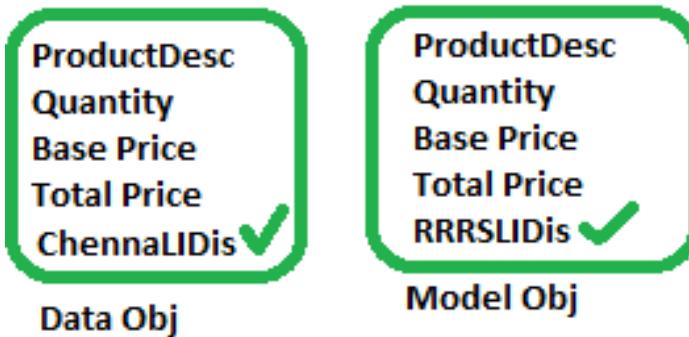
```
3  ly] hybris Platfrom
4
5 Copyright (c) 2017 SAP SE or an SAP affiliate company. All rights reserved.
6
7 This software is the confidential and proprietary information of SAP
8 ("Confidential Information"). You shall not disclose such Confidential
9 Information and shall use it only in accordance with the terms of the
10 license agreement you entered into with SAP.
11 -->
12<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="beans.xsd">
13
14<bean class="de.hybris.platform.commercefacades.product.data.ProductData">
15    <property name="genders" type="java.util.List<com.rrrs.rrrstraining.facades.product.data.GenderData>"/>
16</bean>
17
18<bean class="com.rrrs.rrrstraining.facades.product.data.GenderData">
19    <property name="code" type="String"/>
20    <property name="name" type="String"/>
21</bean>
22
23<bean class="de.hybris.platform.commercefacades.order.data.OrderEntryData">
24    <property name="ChennaLIDis" type="String"/>
25</bean>
```

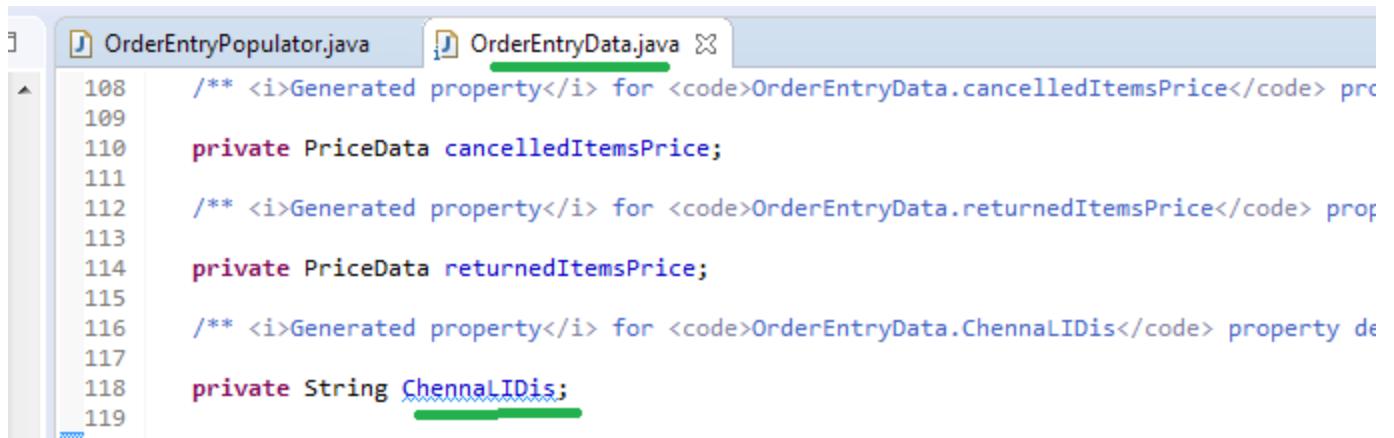
Note = Model Obj (or) Model class & Data Obj (or) Data class can have same attribute names.

Step 7 = Do the build (**ant clean all**)

Note = After build successful –

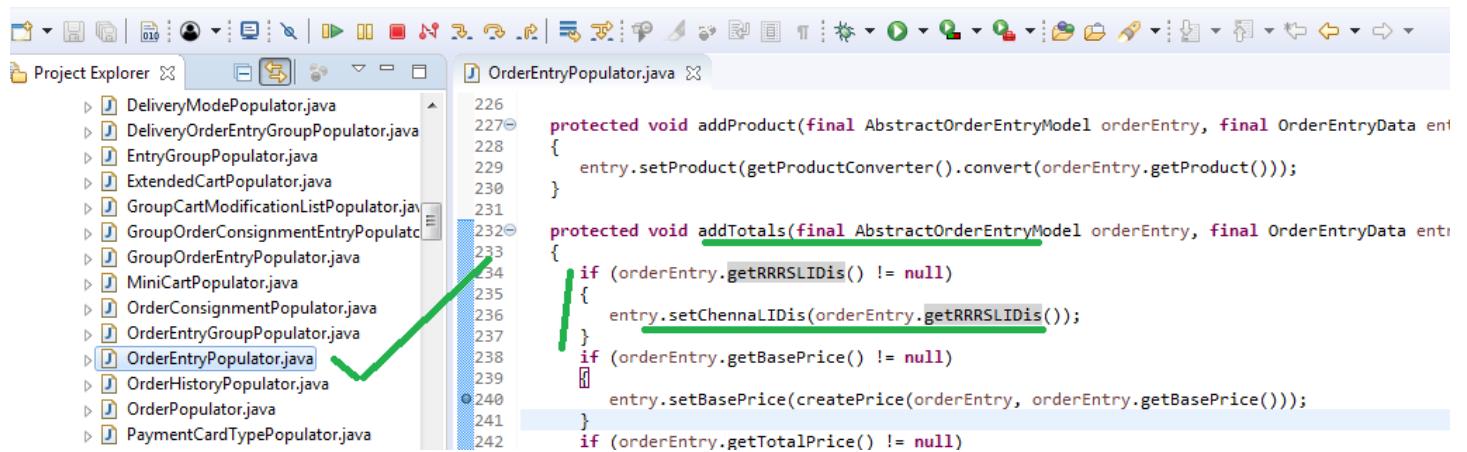
Check results (New Attribute ChennaLIDis created).





```
108     /** <i>Generated property</i> for <code>OrderEntryData.cancelledItemsPrice</code> pro
109
110     private PriceData cancelledItemsPrice;
111
112     /** <i>Generated property</i> for <code>OrderEntryData.returnedItemsPrice</code> prop
113
114     private PriceData returnedItemsPrice;
115
116     /** <i>Generated property</i> for <code>OrderEntryData.ChennaiLIDis</code> property de
117
118     private String ChennaiLIDis;
119
```

Step 8 = Populate the Model Obj data into Data Obj.



```
226     protected void addProduct(final AbstractOrderEntryModel orderEntry, final OrderEntryData ent
227     {
228         entry.setProduct(getProductConverter().convert(orderEntry.getProduct()));
229     }
230
231     protected void addTotals(final AbstractOrderEntryModel orderEntry, final OrderEntryData ent
232     {
233         if (orderEntry.getRRRSЛИDis() != null)
234         {
235             entry.setChennaiLIDis(orderEntry.getRRRSЛИDis());
236         }
237         if (orderEntry.getBasePrice() != null)
238         {
239             entry.setBasePrice(createPrice(orderEntry, orderEntry.getBasePrice()));
240         }
241         if (orderEntry.getTotalPrice() != null)
```

Step 9 = Do the build (ant clean all)

Step 10 = Start the Server (hybrisserver.bat)

Step 11 = Do the UI (or) PL layer changes.

Photon - rrrstrainingstorefront/web/webroot/WEB-INF/tags/responsive/order/orderUnsignedEntries.tag - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run SAP Hybris [y] Hybris Window Help
Project Explorer base_en.properties orderUnsignedEntries.tag
  theme-lambda-desktop_de.properties
  theme-lambda-desktop_en.properties
  theme-lambda-desktop_ja.properties
  theme-lambda-desktop_zh.properties
  tags
    addons
    desktop
    responsive
      action
      address
      cart
      checkout
      common
      formElement
      grid
      nav
      order
        accountOrderDetailOrderTo...
        accountOrderDetailsItem.ta...
  33   <order:addressItem address="${orderData.deliveryAddress}" />
  34   </div>
  35   </div>
  36   </div>
  37   </div>
  38   </c:otherwise>
  39   </c:choose>
  40   </div>
  41
  42
  43   <ul class="item_list">
  44     <li class="hidden-xs hidden-sm">
  45       <ul class="item_list--header">
  46         <li class="item_toggle"></li>
  47         <li class="item_image"></li>
  48         <li class="item_info"><spring:theme code="basket.page.item"/></li>
  49         <li class="item_price"><spring:theme code="basket.page.price"/></li>
  50         <li class="item_quantity"><spring:theme code="basket.page.qty"/></li>
  51
  52       <li class="item_quantity"><spring:theme code="rrrs.chenna.dis"/></li>
  53
  54     <li class="item_total--column"><spring:theme code="basket.page.total"/></li>
  55   </ul>

```

Photon - rrrstrainingstorefront/web/webroot/WEB-INF/messages/base_en.properties - Eclipse IDE

```

File Edit Navigate Search Project Run SAP Hybris [y] Hybris Window Help
Project Explorer base_en.properties orderUnsignedEntries.tag
  base_en.properties
  base_ja.properties
  base_zh.properties
  site-apparel_de_de.properties
  site-apparel_de_en.properties
  site-apparel_uk_en.properties
  site-electronics_de.properties
  site-electronics_en.properties
  site-electronics_ja.properties
  137 basket.page.shipping.pickup = Pickup
  138 basket.page.shipping.ship = Ship
  139 basket.page.title = Item
  140 basket.page.title.pickupFrom = Pick Up From:
  141 basket.page.title.yourDeliveryItems = Your Delivery Items
  142 basket.page.title.yourDeliveryItems.pickup = Order items to be picked up wil
  143 basket.page.title.yourItems = Your Cart
  144 basket.page.title.yourPickUpItems = Your Pick Up Items
  145 basket.page.total = Total
  146 rrrs.chenna.dis = RRRS LI Dis.
  147 basket.page.totals.delivery = Delivery:

```

Photon - rrrstrainingstorefront/web/webroot/WEB-INF/tags/responsive/order/orderEntryDetails.tag - Eclipse IDE

```

File Edit Source Refactor Navigate Search Project Run SAP Hybris [y] Hybris Window Help
Project Explorer orderEntryDetails.tag
  theme-blue-desktop_en.properties
  theme-blue-desktop_ja.properties
  theme-blue-desktop_zh.properties
  theme-lambda-desktop_de.properties
  theme-lambda-desktop_en.properties
  theme-lambda-desktop_ja.properties
  theme-lambda-desktop_zh.properties
  tags
    addons
    desktop
    responsive
  147 <%-- RRRS LI Dis --%>
  148   <div class="item_total hidden-xs hidden-sm">
  149     ${orderEntry.chennaLIDis}
  150   </div>
  151
  152
  153 <%-- total --%>
  154   <div class="item_total hidden-xs hidden-sm">
  155     <ycommerce:TestId code="orderDetails_productTotalPrice_">
  156       <format:price priceData="${orderEntry.totalPrice}" />
  157     </ycommerce:TestId>
  158   </div>
  159

```

Step 12 = Enter Discount Data

Product	Quantity	Base Price	Total Price	RRRS Li Dis
300441355	3	101.21	303.63	13%
300460184	4	64.76	259.04	18%
300392200	5	36.41	182.05	14%

Step 13 = Test the Results in Storefront now: -

Req = In Order Details Page ...
Display the Order Line Item Discounts

ITEM (STYLE NUMBER)	PRICE	QTY	RRRS LI Dis	TOTAL
Beacon Jacket glacier M 300441355 In Stock	£101.21	3	Style: glacier Size: M 13%	£303.63
Last Mission Jacket snapper L 300460184 In Stock	£64.76	4	Style: snapper Size: L 18%	£259.04

See Overall Steps: -

OrderEntries (Table): -

Product	Quantity	Base Price	Total Price	RRRS Li Dis
300441355	3	101.21	303.63	
300460184	4	64.76	259.04	
300392200	5	36.41	182.05	

Step 1 = Create new Col called "RRRSLiDis"

```
xml <rrstrainingcore-items.xml>
<itemtype code="AbstractOrderEntry"
  extends="AbstractOrderEntry"
  abstract="true"
  autoreate="false"
  generate="true">
  <attributes>
    <attribute qualifier="RRRSLiDis"
      type="java.lang.String"
      <persistence type="property"/>
  </attribute>
</itemtype>
```

Step 2 = Do the build (ant clean all)

```
AbstractOrderEntryModel.java
144 public static final String
145     RRRSLIDIS = "RRRSLiDis";
146
```

Step 3 = Start the Server (hybrisserver.bat)
Step 4 = Perform Platform -- Update (hAC)
Step 5 = Check the Results

Product	Quantity	Base Price	Total Price	RRRS Li Dis
300441355	3	101.21	303.63	
300460184	4	64.76	259.04	
300392200	5	36.41	182.05	

Note:- For DB Purpose -- We have Model Obj.

For PL Purpose -- We have Data Obj

ProductDesc
Quantity
Base Price
Total Price
RRRSLiDis
Data Obj

ProductDesc
Quantity
Base Price
Total Price
RRRSLiDis
Model Obj

Step 7 = Do the build (ant clean all)
Check the results.

ProductDesc
Quantity
Base Price
Total Price
ChennaLiDis
Data Obj

ProductDesc
Quantity
Base Price
Total Price
RRRSLiDis
Model Obj

Step 8 = Populate Model Obj data into Data Obj.

```
*OrderEntryPopulator.java
protected void addTotals(AbstractOrderEntryModel
  orderEntry, final OrderEntryData entry)
{
  if (orderEntry.getRRRSLiDis() != null)
    entry.setChennaLiDis(orderEntry.getRRRSLiDis())
}
```

Step 9 = Do the build (ant clean all)
Step 10 = Start the Server (hybrisserver.bat)

Step 11 = Do the PL / UI changes

```
orderUnsignedEntries.tag
<ul class="item_list">
  <li class="hidden-xs hidden-sm">
    <ul class="item_list--header">
      <li class="item_quantity"><spring:theme
        code="rrs.chenna.dis"/></li>
    
```

Step 12 = Enter the Data (Discount Data)

Product	Quantity	Base Price	Total Price	RRRS Li Dis
300441355	3	101.21	303.63	13%
300460184	4	64.76	259.04	18%
300392200	5	36.41	182.05	14%

Step 13 = Check the Results in Storefront.

Order Details | Apparel Site UK

https://localhost:9002/rrstrainingstorefront/en/my-

ITEM (STYLE NUMBER)	PRICE	QTY	RRRS Li Dis	TOTAL
Beacon Jacket glac 300441355 In Stock	£101.21	3	13	£303.63

```

OrderEntry
=====
Desc Price QTY Total
-----
D1 10 3 49
D2 13 4 45

Step 1 = Create an attribute called
"chennaDis" in "AbstractOrderEntry"
(core-items.xml)
emtype code="AbstractOrderEntry"
extends="GenericItem"
jaloClass="de.hybris.platform.jalo."
abstract="true" autocreate="true"
<attributes>
<attribute qualifier="chennaDis" ty=
<persistence type="property" />
<modifiers read="true" write="true"
</attribute>
Step 2 = ant clean all ....
Start the Server .... hAC --> Update
== Output: -
OrderEntry
=====
Desc Price QTY Total chennaDis
-----
D1 10 3 49
D2 13 4 45

```

Step 3 = Enter the chennaDis Data

```

OrderEntry
=====
Desc Price QTY Total chennaDis
-----
D1 10 3 49 13%
D2 13 4 45 18%

```

Note: - Our Goal is to display the chennaDis in Storefront.

Output after the build (ant clean all)

```

AbstractOrderEntryModel.java
56  /**
57   * Generated relation code constant for
58   */
59  public final static String _CONSIGNMENTID =
60  /**
61   * Generated constant - Attribute key of <attribute
62   * name="chennaDis" type="string" />
63  */
64  public static final String CHENNADIS = "chennaDis";

```

Desc = D1
Price = 10
QTY = 3
Total = 49
chennaDis = 13%
=====

Model Obj = AOEM

For DB Purpose -- We have Model Obj (AOEM)
For UI Purpose -- We have Data Obj (OED)
AOEM = Abstract Order Entry Model
OED = Order Entry Data

After Build ==> Output (OED)

Desc	Price	QTY	Total
chennaDisData			

Step 4 = Create new attribute called "chennaDisData" (Note: - We can also have the same name as Model Obj attribute).

--> To have 1 attribute in Model Obj, we need to do the changes in "-items.xml" file.
--> To have 1 attribute in Data Obj, we need to do the changes in "facades-beans.xml"
--> Whenever we write *DAO file / *Service file, we need to define those as beans inside "spring.xml"

Step 6 = Do the Data populate

Desc	Price	QTY	Total
D1	10	3	49
chennaDisData			

OED

Step 7 = Build .. Start Server ...

Step 8 = Identify the UI Files & Use the OrderEntryData fields & Display in Storefront as per the Requirement....

orderUnsignedEntries.tag
orderEntryDetails.tag

Step 9 = Test the Results....

OrderEntries [Table]

PCode	Price	QTY	Total
P1	123	3	369
P2	234	4	936
P3	345	5	1725

Step 1 = Create new Col = RRRSLIDs

*core-items.xml

Not Best Practice

```

<itemtype code="AbstractOrderEntry">
<attributes>
<attribute qualifier="basePrice" ty=
<attribute qualifier="product" ty=
<attribute qualifier="quantity" ty=
<attribute qualifier="taxValues" ty=
<attribute qualifier="totalPrice" ty=
<attribute qualifier="RRRSLIDs" t=

```

This will work ... But not best practice.

Q = Then what to do?

Step 2 = Do the build [ant clean all]

AbstractOrderEntryModel

PCode	Price	QTY	Total
= RRRSLIDs			

OrderEntryData

PCode	Price	QTY	Total
Create variable here			

AbstractOrderEntryModel

PCode	Price	QTY	Total
= RRRSLIDs			

Step 4 = Create variable in **data class / **data Obj.

Note =

For DB purpose -- We have *Model class / *Model Obj. These are created based on *.items.xml.

For UI purpose -- We have *Data class / *Data Obj. These are created based on *facades-beans.xml.

*commercefacades-beans.xml

Not Best Practice

```

<bean class="de.hybris.platform.commercefacades.order.data.
<property name="quantity" type="Long"/>
<property name="basePrice" type="de.hybris.platform.com
<property name="totalPrice" type="de.hybris.platform.com
<property name="product" type="de.hybris.platform.comme
<property name="ChennaLIDis" type="java.lang.String"/>

```

This will work ... But not best Practice.

*chenarrstrainingfacades-beans.xml

Best Practice

```

<!-- Example 8 -->
<bean class="de.hybris.platform.commercefacades.order.
<property name="ChennaLIDis" type="String"/>

```

Step 5 = Do the Build [ant clean all]

OrderEntryData

PCode	Price	QTY	Total
= ChennaLIDis			

AbstractOrderEntryModel

PCode	Price	QTY	Total
= RRRSLIDs			

Step 6 = Populate *Model Obj data into *Data Obj

OrderEntryPopulator.java

```

@Override
public void populate(final AbstractOrderEntryModel
source, final OrderEntryData target)
{
    addTotals(source, target);
}
protected void addTotals(final AbstractOrderEntryModel
entry, final OrderEntryData entry)
{
    if (entry.getRRRSLIDs() != null)
    {
        entry.setChennaLIDis(entry.getRRRSLIDs());
    }
}
Put Data in *Data Obj Get Data from *Model Obj

```

Step 7 = Take data from *Data Obj & Display in UI

orderUnsignedEntries.tag

```

.48<div class="item_total hidden-sm">
.49    <-- total -->
.50    ${orderEntry.chennaLIDis}
.51  </div>
.52
.53

```

orderEntryDetails.tag

Step 8 = Results [Test it]

Conclusion = In this scenario –

We already have the table and columns.

We already have the “SAP Comm” code [DAO File ... Service File ... Façade File ... Converter ... Populator ... Controller ...]

We already have the “UI Code”.

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Then what we are trying to do?

We created new column & display that column data in UI layer.

High level steps =

Step 1 = Create new column in existing table [`*core-items.xml`]

Step 2 = Create new variable in Data Obj [`*facades-beans.xml`]

Step 3 = Do Populator changes

Step 4 = Do UI Changes

Step 5 = Start the Server ... "hAC – Update".

Step 6 = Results [Test it]

Business Requirement =

Save Cart – Will have Description. Let's say we entered some Desc today, after 1 week, if we don't want then we are not able to remove.

The screenshots show a sequence of interactions with a web application:

- Step 1:** A screenshot of a "Save Cart" dialog box. It shows fields for "NAME" (FirstCart) and "DESCRIPTION" (Cart Desc Not Emp'). A red annotation at the top left says "Step 1 = Saved the Cart with some Desc".
- Step 2:** A screenshot of an "Edit Saved Cart" dialog box. It shows the same "NAME" field and a note: "You removed the Desc & --> Save". A red annotation at the top left says "Step 2 = After 1 week, your company asked you to removed the Desc.". A large red X is drawn over the "SAVE" button.
- Saved Cart Details:** A screenshot of a table showing the saved cart details. The "DESCRIPTION" field is listed as "Cart Desc Not Empty" with a crossed-out icon. A red annotation at the bottom right says "Even after removing the Desc, still it's coming."

Step 1 = Find the UI code

The screenshot shows a web application interface for editing a saved cart. The main title is "Edit Saved Cart". On the left, there's a sidebar with "BRANDS" and "STREETWEAR" links, and a message "Cart FirstCart was successfully created". The main content area has two input fields: "NAME" with value "FirstCart" and "DESCRIPTION" with value "Cart Desc Not Empty". A red arrow points to the "DESCRIPTION" field. Below this is a table with columns "NAME", "ID", and "DATE SAVED". At the bottom, the browser's developer tools are open, specifically the "Elements" tab, which displays the HTML structure of the page, including the form and its fields.

UI File = saveCartModel.tag

```
saveCartModal.tag
22<
23    <label class="control-label" for="name">
24        <spring:theme code="basket.save.cart.name" />
25    </label>
26    <form:input cssClass="form-control" id="saveCartName" path="name" maxlen="255" type="text" />
27    <div class="help-block right-cartName" id="remain">
28        Characters Left : 255
29    </div>
30    <div class="form-group">
31        <label class="control-label" for="description">
32            <spring:theme code="basket.save.cart.description" />
33        </label>
34        <form:textarea cssClass="form-control" id="saveCartDescription" path="description" type="text" />
35        <div class="help-block" id="remainTextArea">
36        </div>
37    </div>
38
39    <div class="modal-footer">
40        <button type="button" class="btn btn-primary" data-dismiss="modal" href="#">Save
41    </div>
42
43</div>
```

```

251     return getSaveCartTextGenerationStrategy().generateSaveCartName(cartModel);
252 }
253
254
255 protected String generateSaveCartDescription(final CartModel cartModel, final String description)
256 {
257     if (StringUtils.isNotEmpty(description))
258     {
259         return description;
260     }
261     else if (StringUtils.isNotEmpty(cartModel.getDescription()))
262     {
263         return cartModel.getDescription();
264     }
265
266
267     return getSaveCartTextGenerationStrategy().generateSaveCartDe
268 }
```

This code (Else) is not required.
DefaultSaveCartFacade.java is
file available in commercefacades
(Standard Ext)

In Above method Logic Is: -

If Desc (UI Entered value) is not empty, then return the same Desc.

Else we are returning Cart Mode – Desc (Which is previously saved).

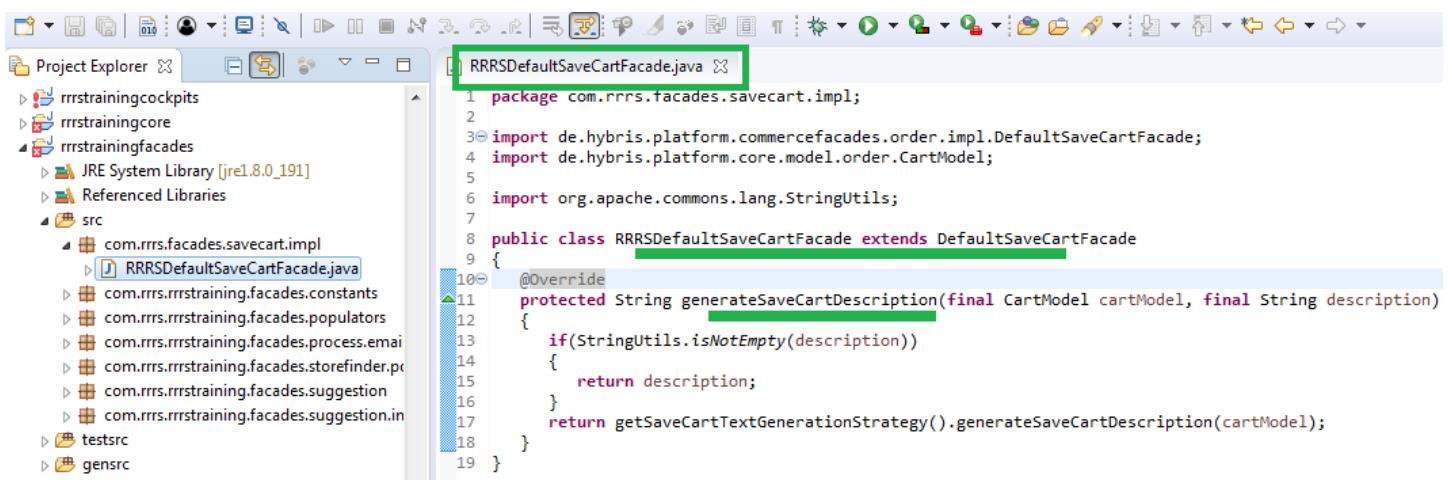
Step 2 = How to fix this.

C:\rrrssoftwares\HYBRISCOMM64\hybris\bin\ext-commerce\commercefacades\src\de\hybris\platform\commercefacades\order\impl\DefaultSaveCartFacade.java (DSCF)

DSCF.java is available in commercefacades Ext (This is provided by “SAP Comm”).

Thumb Rule == Generally we don't modify anything directly inside “SAP Comm” provided platform Exts.

So → U need to Override above Java file methods in your Exts (**rrrstrainingfacade**).



The screenshot shows the Eclipse IDE interface with the Project Explorer on the left and the RRRSDefaultSaveCartFacade.java editor on the right. The code is as follows:

```
1 package com.rrrs.facades.savecart.impl;
2
3 import de.hybris.platform.commercefacades.order.impl.DefaultSaveCartFacade;
4 import de.hybris.platform.core.model.order.CartModel;
5
6 import org.apache.commons.lang.StringUtils;
7
8 public class RRRSDefaultSaveCartFacade extends DefaultSaveCartFacade
9 {
10     @Override
11     protected String generateSaveCartDescription(final CartModel cartModel, final String description)
12     {
13         if(StringUtils.isNotEmpty(description))
14         {
15             return description;
16         }
17         return getSaveCartTextGenerationStrategy().generateSaveCartDescription(cartModel);
18     }
19 }
```

Step 3 = Whenever you write Façade File / Service File / [Java File] → You need to define this in ***-spring.xml**

If you want any DB Changes → ***core-items.xml**

If you want any Data Obj Fields → ***facades-beans.xml**

If you write any Service File / Façade File / ... → ***-spring.xml**

The screenshot shows the Eclipse IDE interface with the Project Explorer on the left and the editor on the right. The editor displays the file `commercefacades-spring.xml`. A large red 'X' is drawn over the code area, indicating that this configuration is incorrect or incomplete. The code itself defines a bean for a save cart facade:

```
<!-- Save Cart -->
<alias name="defaultSaveCartFacade" alias="saveCartFacade"/>
<bean id="defaultSaveCartFacade" class="de.hybris.platform.commercefacades.DefaultSaveCartFacade">
    <property name="commerceSaveCartService" ref="commerceSaveCartService"/>
    <property name="saveCartTextGenerationStrategy" ref="commerceSaveCartTextGenerationStrategy"/>
    <property name="configurationService" ref="configurationService"/>

```

The screenshot shows the Eclipse IDE interface with the Project Explorer on the left and the editor on the right. The editor displays the file `rrstrainingfacades-spring.xml`. A green checkmark is drawn next to the file in the Project Explorer, indicating that this configuration is correct. The code defines a bean for a save cart facade:

```
<alias name="rrrsDefaultSaveCartFacade" alias="saveCartFacade"/>
<bean id="rrrsDefaultSaveCartFacade" class="com.rrrs.facades.savecart.impl.RRRSDefaultSaveCartFacade" parent="defaultSaveCartFacade"/>
```

Step 4 = Do the Build (ant clean all)

- Start the Server
- Test the Results.

Saved Cart Details | Apparel Site

Not secure | https://localhost:9002/chennatrainingstorefront/en/my-account/saved-carts/00003000

BRANDS STREETWEAR SNOW ACCESSORIES YOUTH

HOME / SAVED CARTS / SAVED CART 00003000

◀ | Saved Cart Details

NAME	ID	DATE SAVED	QTY
FirstCart	00003000	Jan 18, 2019 11:16 PM	1
DESCRIPTION	-		

Conclusion = In this scenario –

We already have the table and columns.

We already have the “SAP Comm” code [DAO File ... Service File ... Façade File ... Converter ... Populator ... Controller ...]

We already have the “UI Code”.

Then what we are trying to do?

Some of the logic whatever “SAP Comm” given is not working as per customer needs. Q = How to override the given logic / method?

Step 1 = Find Existing Logic (or) Code

Step 2 = Override the Logic (or) Code (or) Method

Step 3 = Register newly created class [*-spring.xml]

Step 4 = Build ... Start ... Test

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Q = “SAP Comm” commercefacades extension facades mostly return?

- a) Data objects
- b) Model objects
- c) Data model
- d) none

Q = “SAP Comm” ServiceLayer Models should not be used as part of a facade interface, maintaining a clean abstraction of the?

- a) Business layer and the persistence layer
- b) Business layer and the presentation layer**
- c) Persistence layer and the presentation layer
- d) All

Q: All converters should be Spring configured only and should use the -----
----- base class

- a) AbstractConverter**
- b) DefaultConverter
- c) a & b
- d) none of above

Q = -----Is an implementation of a Populator pipeline where each population step is evaluated against a Set of Enum values passed by the caller.

- a) DefaultConfigurablePopulator**
- b) AbstractPopulatingConverter
- c) AbstractConverter

Q = Product Populators are a little unique in that they typically extend a ----- class that is variant aware and supports the ability of falling back to a variants parent product for attribute values in the event of the source product value being null

- a) DefaultProductPopulator b) AbstractProductPopulatingConverter
- c) AbstractProductPopulator** d) None

Q =extension is the template shipped with the “SAP Comm” Accelerator that you can use as a starting point for your own extension.

- a) yacceleratorfacades** b) commercefacades
- c) both d) none

Q = Data Transfer Objects (DTOs) are objects created to contain....

- a) Values** b) Business logic c) Both d) none of them

Q =template method that allows a concrete sub-class to pick the appropriate target data object implementation.

- a) createListTarget **b) createTarget**
- c) createSourceToTarget d) None of the above

Q = Facades are which scoped Spring managed beans

- a) yrequest **b) singleton** c) tenant d) request

Q = Provides access to various internationalization switches that the user can make when they visit a specific storefront

- a) **Store Session façade**
- b) Store Locator façade
- c) User façade
- d) User Locale façade

Q = Explain different types of interceptors?

Model Interceptors = Intercept the behavior of the lifecycle of Models. Model lifecycle consists of loading from Database, saving to Database and deleting or removing from the Database.

In lifecycle of a model, interceptors can intercept & modify model data. It includes auditing, validating & even restricting the data from being removed/deleted from database if certain conditions are not met.

There are 5 types of Interceptors: -

- a) LoadInterceptor = Invoked whenever a model is loaded from the database
- b) InitDefaultsInterceptor = Invoked during modelService.create() & modelService.initDefaults()
- c) PrepareInterceptor = Invoked before model is saved to database & before ValidateInterceptor
- d) ValidateInterceptor = Invoked before a model is saved to database & after PrepareInterceptor
- e) RemoveInterceptor = Invoked before a model is removed from database.

End – End Scenarios: -

Not secure | localhost:9002/chennarrstrainingstorefront/en/my-account/saved-carts

B2C Accelerator

I'm looking for

BRANDS STREETWEAR SNOW ACCESSORIES YOUTH

HOME / SAVED CARTS

1 Saved Cart

NAME	ID	DATE SAVED	DESCRIPTION
1stCart123	00004000	Jun 4, 2021 3:30 AM	This is Tejaswini 1st Order ===

Query
Cart

Not secure | https://localhost:9002/chennarrstrainingstorefront/en/Categories/Snowwear-men/Snow-Jackets/c/190100

B2C Accelerator

I'm looking for

BRANDS STREETWEAR SNOW ACCESSORIES YOUTH

HOME / CATEGORIES / SNOWWEAR MEN / SNOW JACKETS

Shop by Stores
or

Query
Category
Products

SAP Comm DB

Not secure | localhost:9002/chennarrstrainingstorefront/en/my-account/orders

B2C Accelerator

I'm looking for

BRANDS STREETWEAR SNOW ACCESSORIES YOUTH

HOME / ORDER HISTORY

Order History

1 Order

ORDER NUMBER	ORDER STATUS	DATE PLACED
00003000	Created	Jun 3, 2021 3:30 AM

Query
Order

SAP Comm

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Requirement = Create an itemtype called “ChennaRRRCourses” & Table name = “RRRSTrainingCourses”. Insert the Records ... Display the records in **Storefront**.

The screenshot shows the hybris Backoffice interface. On the left, a sidebar menu is open under the 'Commerce' tab, with 'Types' selected. In the main content area, a search bar at the top contains 'ChennaRRRCourses'. Below it, the title '(ItemType)' is displayed. A table titled 'Table Records' lists four entries:

ChennaRRRCourses.duration	ChennaRRRCourses.amount	ChennaRRRCourses.code	ChennaRRRCourses.name
60 Hrs	500	1013	C Lang
50 Hrs	500	1012	Data Hub
80 Hrs	500	1011	Hybris

A green arrow points from the sidebar's 'Types' link to the search bar. Two black diagonal slashes are overlaid on the table area, pointing towards the 'Table Records' text.

Step 1 = Create itemtype called “ChennaRRRCourses”

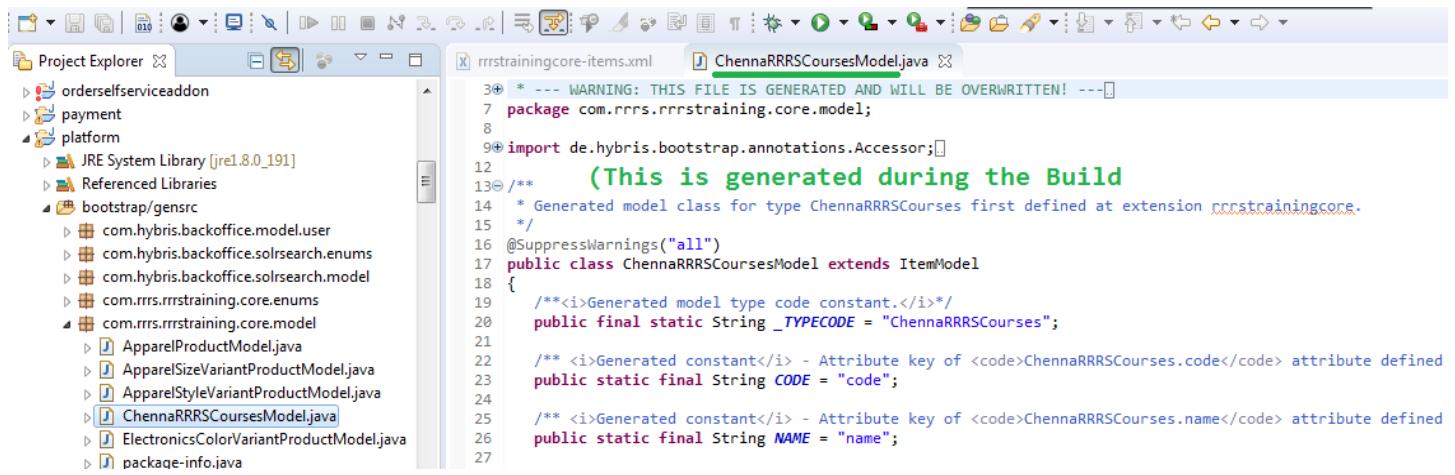
The screenshot shows the Eclipse IDE with the 'Project Explorer' view on the left and an XML editor on the right. The XML file is named 'rrrstrainingcore-items.xml'. A vertical green line highlights the code within the 'itemtype' tag. The code defines the itemtype 'ChennaRRRCourses' with attributes: 'code', 'name', 'duration', and 'amount'. The 'code' attribute is set to 'ChennaRRRCourses', 'name' to 'ChennaRRRCourses', 'duration' to '60 Hrs', and 'amount' to '500'.

```

<itemtype code="ChennaRRRCourses" generate="true" autoCreate="true" >
    <deployment table="RRRSTrainingCourses" typecode="10128" />
    <attributes>
        <attribute qualifier="code" type="java.lang.String">
            <modifiers optional="false" unique="true" />
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="name" type="java.lang.String">
            <modifiers optional="false" />
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="duration" type="java.lang.String">
            <modifiers optional="false" />
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="amount" type="java.lang.Integer">
            <modifiers optional="false" />
            <persistence type="property"/>
        </attribute>
    </attributes>
</itemtype>

```

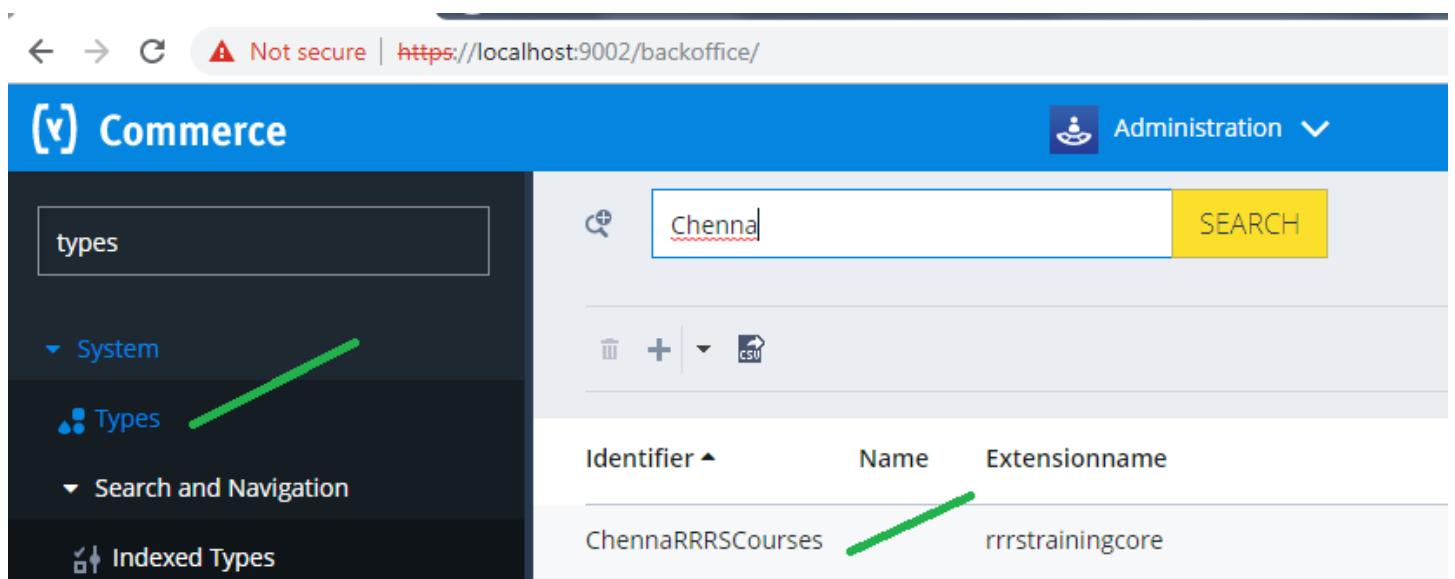
Step 2 = Build (ant clean all) ...



The screenshot shows the Eclipse IDE interface. The Project Explorer view on the left lists several projects and their components. The Java code editor on the right displays the file `ChennaRRRCoursesModel.java`. The code is annotated with green text indicating it was generated during the build process. The code defines a class `ChennaRRRCoursesModel` extending `ItemModel`, with static final strings for `_TYPECODE`, `CODE`, and `NAME`.

```
* --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN! ---
package com.rrrs.rrrstraining.core.model;
import de.hybris.bootstrap.annotations.Accessor;
/**
 * Generated model class for type ChennaRRRCourses first defined at extension rrrstrainingcore.
 */
@SuppressWarnings("all")
public class ChennaRRRCoursesModel extends ItemModel
{
    /**Generated model type code constant.</i> */
    public final static String _TYPECODE = "ChennaRRRCourses";
    /**
     *Generated constant</i> - Attribute key of <code>ChennaRRRCourses.code</code> attribute defined
    public static final String CODE = "code";
    /**
     *Generated constant</i> - Attribute key of <code>ChennaRRRCourses.name</code> attribute defined
    public static final String NAME = "name";
}
```

Step 3 = Start the Server (Hybrisserver.bat) ... hAC – Update



The screenshot shows the Hybris Administration Console (hAC) interface. The left sidebar has a navigation menu with items like System, Types, Search and Navigation, and Indexed Types. A green arrow points from the 'Types' link in the sidebar to the search bar in the main content area. The search bar contains the text 'Chenna'. The main content area displays a table with columns Identifier, Name, and Extensionname. One row in the table is highlighted with a green arrow pointing to the Extensionname column, which contains the value 'rrrstrainingcore'.

Identifier	Name	Extensionname
ChennaRRRCourses		rrrstrainingcore

Step 4 = Insert the records into RRRSTrainingCourses table.

The screenshot shows the hybris administration console interface. On the left, there is a sidebar with a search bar at the top containing 'types'. Below it, under 'System', is a 'Types' section with a plus sign icon. Under 'Search and Navigation', there is an 'Indexed Types' section. Under 'Catalog', there is a 'Product Variant Types' section. In the center, a modal window titled 'Create New ChennaRRRCourses' is open. It has a header with a magnifying glass icon and a 'SEARCH' button. Below the header, it says 'Provide All Mandatory Fields'. There are three input fields: '[name]:' with a placeholder '60 Hrs', '50 Hrs', and '80 Hrs'; 'Time created:' with a placeholder 'Training1'; and 'ChennaRRRCourses' with a placeholder 'Hybris'. A green diagonal line highlights the 'Types' icon in the sidebar and the 'name:' field in the modal.

(OR) – Write ImpEx & Execute from hAC

(or) Keep ImpEx in *Initialdata Ext

```
1 insert_update ChennaRRRCourses;code[unique=true];name;duration;amount
2 ;Training1;Hybris;90;500
3 ;Training2;DataHub;55;500
4 ;Training3;Salesforce;60;5000
```

The screenshot shows the hybris administration console with a blue header bar. The header includes the title '(*) hybris administration console', a 'logout' button, and a message 'You're Administrator'. Below the header is a navigation bar with tabs: 'Platform', 'Monitoring', 'Maintenance', and 'Console'. The 'Console' tab is selected. The main content area is titled 'ImpEx Import'. At the top of this area are two buttons: 'Import content' and 'Import script'. The 'Import content' button is highlighted with a blue background. Below these buttons is a section titled 'Import content' with a dashed border. Inside this section is the same ImpEx code as shown in the previous screenshot:
1 insert_update ChennaRRRCourses;code[unique=true];name;duration;amount
2 ;Training1;Hybris;90;500
3 ;Training2;DataHub;55;500
4 ;Training3;Salesforce;60;5000

Contact Us = ChennaReddyTraining@RRRS.CO.IN

tenantID

Master
true

Update running system
 Clear the hMC configuration from the database
 Create essential data
 Localize types

Project data settings

Results =

ChennaRRRCourses.duration	ChennaRRRCourses.amount	ChennaRRRCourses.code	ChennaRRRCourses.name
60	5000	Training3	Salesforce
55	500	Training2	DataHub
90	500	Training1	Hybris

Step 5 = Write Flexible Search Query to get (or) fetch the Above table records.

```

1 package com.rrrs.rrstraining.core.courses.dao.impl;
2
3 import de.hybris.platform.servicelayer.internal.dao.AbstractItemDao;
4
5 public class ChennaRRRCoursesDao extends AbstractItemDao
6 {
7     private static final String COURSES_QUERY = "SELECT {C.PK} FROM {ChennaRRRCourses AS C}";
8
9     public List<ChennaRRRCoursesModel> getAllCourses()
10    {
11        final FlexibleSearchQuery query = new FlexibleSearchQuery(COURSES_QUERY);
12
13        final SearchResult<ChennaRRRCoursesModel> result = getFlexibleSearchService().search(query);
14
15        return result.getResult();
16    }
17
18 }
19
20
21
22
23
24
25

```

```
rrstrainingcore-items.xml ChennaRRRCoursesDao.java rrstrainingcore-spring.xml
543         </set>
544     </property>
545     <property name="commerceQuoteDao" ref="commerceQuoteDao"/>
546     <property name="eventService" ref="eventService"/>
547     <property name="timeService" ref="timeService" />
548 </bean>
549
550
551 <bean id="chennaRRRCoursesDao" class="com.rrrs.rrrstraining.core.courses.dao.impl.ChennaRRRCoursesDao" parent="abstractItemDao">
552     <property name="modelService" ref="modelService"/>
553     <property name="flexibleSearchService" ref="flexiblesearchService"/>
554 </bean>
555
556 <bean id="chennaRRRCoursesService" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRCoursesService" />
557
558
559 <import resource="/rrrstrainingcore/processes/quote/quote-buyer-process-spring.xml"/>
560 <import resource="/rrrstrainingcore/processes/quote/quote-salesrep-process-spring.xml"/>
561 <import resource="/rrrstrainingcore/processes/quote/quote-seller-approval-process-spring.xml"/>
562
563 <import resource="/rrrstrainingcore/processes/quote/quote-to-expire-soon-email-process-spring.xml"/>
564 <import resource="/rrrstrainingcore/processes/quote/quote-expired-email-process-spring.xml"/>
```

Step 6 = Write Service Layer Class & Call the DAO Class

The screenshot shows the Eclipse IDE interface. The left side displays the Project Explorer with several projects and their source code structures. The right side shows the Java code editor for the file `ChennaRRRSCoursesService.java`. The code implements a service layer for managing courses, utilizing a DAO layer and a resource injection mechanism.

```
package com.rrrs.rrrstraining.core.courses.service;
import java.util.List;
import javax.annotation.Resource;
import com.rrrs.rrrstraining.core.courses.dao.impl.ChennaRRRSCoursesDao;
import com.rrrs.rrrstraining.core.model.ChennaRRRSCoursesModel;
public class ChennaRRRSCoursesService
{
    @Resource
    private ChennaRRRSCoursesDao chennaRRRSCoursesDao;
    public List<ChennaRRRSCoursesModel> getAllCourses()
    {
        return chennaRRRSCoursesDao.getAllCourses();
    }
}
```

Project Explorer Package Explorer

rrrstrainingcockpits
rrrstrainingcore
JRE System Library [jre1.8.0_191]
Referenced Libraries
src
tests
gensrc
classes
eclipsebin
lib
resources
cockpitng
localization
rrrstrainingcore
test
items.xsd
rrrstrainingcore-backoffice-config.xml
rrrstrainingcore-items.xml
rrrstrainingcore-spring.xml

ChennaRRRCoursesService.java rrrstrainingcore-spring.xml

```
</bean>
<bean id="chennaRRRCoursesDao" class="com.rrrs.rrrstraining.core.courses.dao.impl.ChennaRRRCoursesDao" parent="abstractCoursesDao">
    <property name="modelService" ref="modelService"/>
    <property name="flexibleSearchService" ref="flexibleSearchService"/>
</bean>

<bean id="chennaRRRCoursesService" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRCoursesService" />

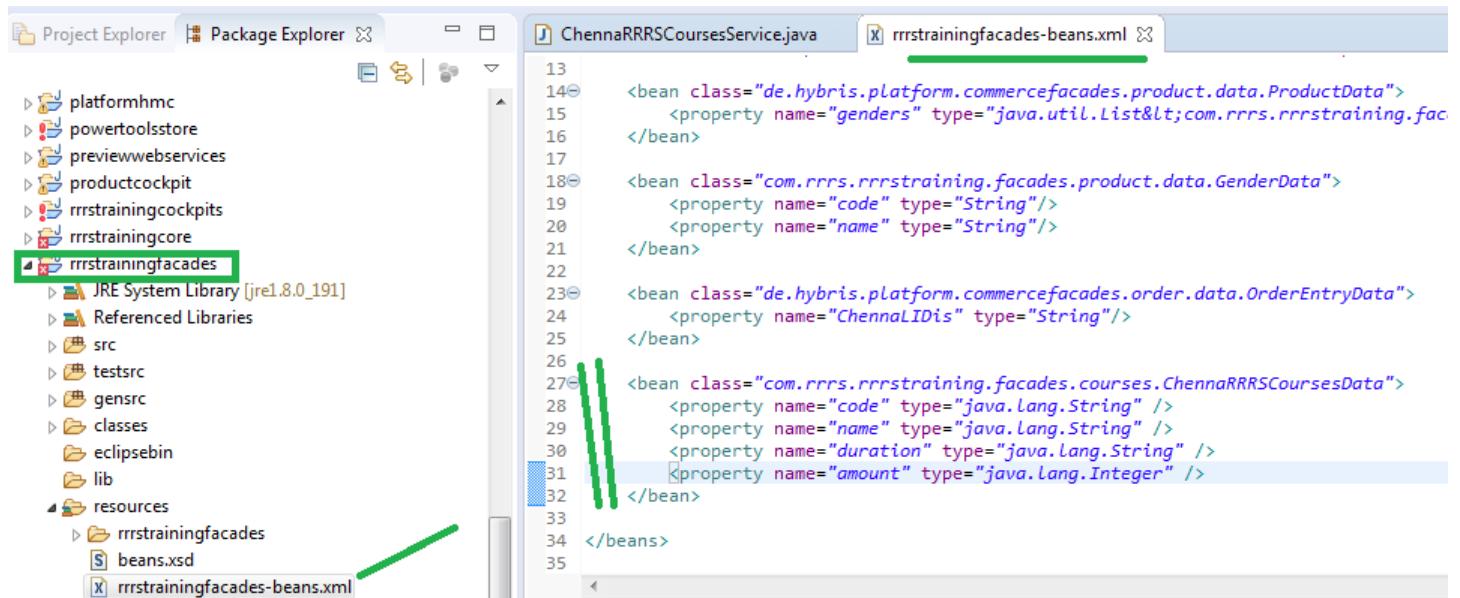
<import resource="/rrrstrainingcore/processes/quote/quote-buyer-process-spring.xml"/>
<import resource="/rrrstrainingcore/processes/quote/quote-salesrep-process-spring.xml"/>
<import resource="/rrrstrainingcore/processes/quote/quote-seller-approval-process-spring.xml"/>

<import resource="/rrrstrainingcore/processes/quote/quote-to-expire-soon-email-process-spring.xml"/>
<import resource="/rrrstrainingcore/processes/quote/quote-expired-email-process-spring.xml"/>

<import resource="/rrrstrainingcore/processes/quote/quote-post-cancellation-process-spring.xml"/>
```

Step 7 = Create the Data Class (or) Data Obj Class

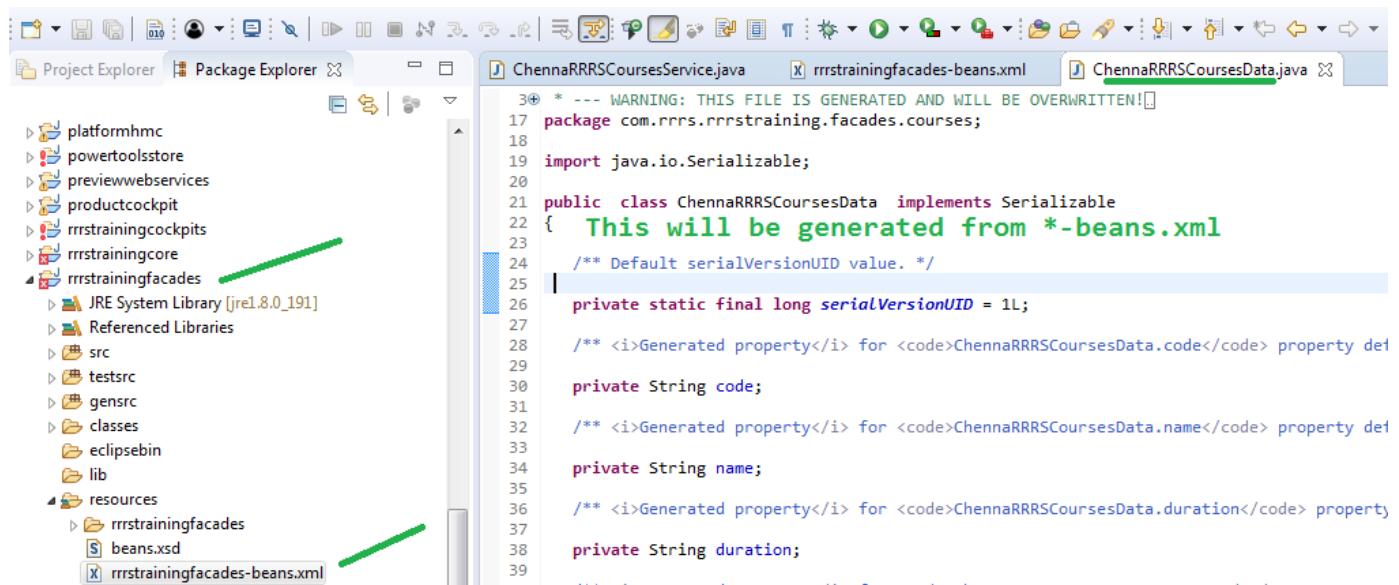
= ChennaRRRCoursesData



```
ChennaRRRCoursesService.java rrstrainingfacades-beans.xml

13     <bean class="de.hybris.platform.commercefacades.product.data.ProductData">
14         <property name="genders" type="java.util.List<com.rrrs.rrrstraining.facade.GenderData>" />
15     </bean>
16
17     <bean class="com.rrrs.rrrstraining.facades.product.data.GenderData">
18         <property name="code" type="String"/>
19         <property name="name" type="String"/>
20     </bean>
21
22     <bean class="de.hybris.platform.commercefacades.order.data.OrderEntryData">
23         <property name="ChennaLIDis" type="String"/>
24     </bean>
25
26     <bean class="com.rrrs.rrrstraining.facades.courses.ChennaRRRCoursesData">
27         <property name="code" type="java.lang.String" />
28         <property name="name" type="java.lang.String" />
29         <property name="duration" type="java.lang.String" />
30         <property name="amount" type="java.lang.Integer" />
31     </bean>
32
33     </beans>
34
35 
```

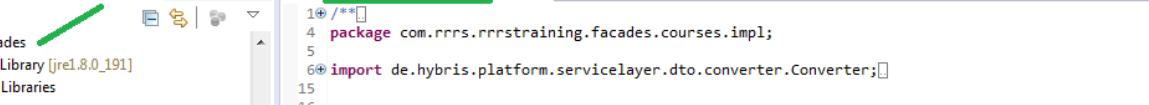
Step 8 = Do the build (ant clean all)



```
ChennaRRRCoursesService.java rrstrainingfacades-beans.xml ChennaRRRCoursesData.java

30 * --- WARNING: THIS FILE IS GENERATED AND WILL BE OVERWRITTEN!
31 package com.rrrs.rrrstraining.facades.courses;
32
33 import java.io.Serializable;
34
35 public class ChennaRRRCoursesData implements Serializable
36 {
37     This will be generated from *.beans.xml
38
39     /** Default serialVersionUID value. */
40
41     private static final long serialVersionUID = 1L;
42
43     /** <i>Generated property</i> for <code>ChennaRRRCoursesData.code</code> property definition
44     private String code;
45
46     /** <i>Generated property</i> for <code>ChennaRRRCoursesData.name</code> property definition
47     private String name;
48
49     /** <i>Generated property</i> for <code>ChennaRRRCoursesData.duration</code> property definition
50     private String duration;
51 }
```

Step 9 = Write the Façade Class & Service Layer Class



```
ChennaRRRSCoursesFacade.java
10 /**
11  * 
12  */
13 package com.rrrs.rrrstraining.facades.courses.impl;
14 
15 import de.hybris.platform.servicelayer.dto.converter.Converter;
16 
17 public class ChennaRRRSCoursesFacade
18 {
19     @Resource
20     private ChennaRRRSCoursesService chennaRRRSCoursesService;
21     private Converter<ChennaRRRSCoursesModel, ChennaRRRSCoursesData> coursesConverter;
22 
23     public List<ChennaRRRSCoursesData> getCoursesData()
24     {
25         final List<ChennaRRRSCoursesModel> chennaRRRSCoursesModels = chennaRRRSCoursesService.getAllCourses();
26 
27         return getCoursesConverter().convertAll(chennaRRRSCoursesModels);
28     }
29 
30     /**
31      * 
32      */
33 }
```

Calling Service Layer Method

Convert Model Obj into Data Obj

```
ChennaRRRSCoursesFacade.java rrstrainingfacades-spring.xml
206     </bean> -->
207
208
209     <alias alias="chennaRRRSCoursesFacade" name="defaultChennaRRRSCoursesFacade"/>
210     <bean id="defaultChennaRRRSCoursesFacade" class="com.rrrs.rrrstraining.facades.courses.impl.ChennaRRRSCoursesFacade">
211         <property name="coursesConverter" ref="defaultChennaRRRSCoursesConverter"/>
212     </bean>
213
214     <alias name="defaultChennaRRRSCoursePopulator" alias="chennaRRRSCoursesConverter"/>
215     <bean id="defaultChennaRRRSCoursePopulator" parent="abstractPopulatingConverter">
216         <property name="targetClass" value="com.rrrs.rrrstraining.facades.courses.ChennaRRRSCoursesData"/>
217         <property name="populators">
218             <list>
219                 <ref bean="chennaRRRSCoursePopulator"/>
220             </list>
221         </property>
222     </bean>
223
```

Step 10 = Write the Populator

```
1  /**
2  * 
3  */
4 package com.rrrs.rrrstraining.facades.courses.populators;
5
6 import de.hybris.platform.converters.Populator;
7
8 public class ChennaRRRCoursesPopulator implements Populator<ChennaRRRCoursesModel, ChennaRRRCoursesData>
9 {
10     @Override
11     public void populate(final ChennaRRRCoursesModel source, final ChennaRRRCoursesData target)
12         throws ConversionException
13     {
14         target.setCode(source.getCode());
15         target.setName(source.getName());
16         target.setDuration(source.getDuration());
17         target.setAmount(source.getAmount());
18     }
19 }
20
21
22
23 }
```

Step 10 = Results

ChennaiRRRSCoursesData				Training1	Hybris	90	500
Training1	Hybris	90	500	Training2	Datahub	55	500
Training2	DataHub	55	500	Training3	Salesforce	60	500
Training3	Salesfo	60	500				

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Project Explorer Package Explorer

rrstrainingfacades

- JRE System Library [jre1.8.0_191]
- Referenced Libraries
- src
- tests
- gensrc
- classes
- eclipsebin
- lib
- resources
 - rrstrainingfacades
 - beans.xml
 - rrstrainingfacades-beans.xml
 - rrstrainingfacades-spring.xml
 - rrstrainingfacades.build.number

rrstrainingfacades-spring.xml ChennaRRRCoursesPopulator.java

```
206     </bean> -->
207
208
209     <alias alias="chennaRRRCoursesFacade" name="defaultChennaRRRCoursesFacade"/>
210     <bean id="defaultChennaRRRCoursesFacade" class="com.rrrs.rrstraining.facades.courses.impl.ChennaRRRCoursesFacade">
211         <property name="coursesConverter" ref="defaultChennaRRRCoursesConverter"/>
212     </bean>
213
214     <alias name="defaultChennaRRRCoursesConverter" alias="chennaRRRCoursesConverter"/>
215     <bean id="defaultChennaRRRCoursesConverter" parent="abstractPopulatingConverter">
216         <property name="targetClass" value="com.rrrs.rrstraining.facades.courses.ChennaRRRCoursesData"/>
217         <property name="populators">
218             <list>
219                 <ref bean="chennaRRRCoursesPopulator"/>
220             </list>
221         </property>
222     </bean>
223
224     <alias name="defaultChennaRRRCoursesPopulator" alias="chennaRRRCoursesPopulator"/>
225     <bean id="defaultChennaRRRCoursesPopulator" class="com.rrrs.rrstraining.facades.courses.populators.ChennaRRRCoursesPopulator"/>
```

Overall Results so far

```
public class ChennaRRRCoursesDao extends AbstractItemDao
{
    private static final String COURSES_QUERY = "SELECT {C.PK} FROM {ChennaRRRCourses AS C}";
    public List<ChennaRRRCoursesModel> getAllCourses()
    {
        final FlexibleSearchQuery query = new FlexibleSearchQuery(COURSES_QUERY);
        final SearchResult<ChennaRRRCoursesModel> result = getFlexibleSearchService().search(query);
```

After Step 8 = Results
ChennaRRRSCoursesData

Step 5 = Results

It contains All 3 Records

ChennaiRRRSCoursesData		It Services Pvt Ltd		
Code	Name	Duration	Amount	
	Training1		Hybris	90 500
	Training2		Databhub	55 500
	Traininge3		Salesforce	60 500

Step 3 = Results

Table = RRRSTrainingCourses

Code	Name	Duration	Amount
------	------	----------	--------

Step 4 =

Results

Step 10 = Results

ChennaRRRSCoursesModel

ChennaRRRSCoursesData				Training1	Hybris	90	500
				Training2	Datahub	55	500
				Training3	Salesforce	60	500
Training1	Hybris	90	500				
Training2	DataHub	55	500				
Training3	Salesfo	60	500				

Database

Step 11 = Write the Controller & Call the Façade Layer

```
1 package com.rrrs.rrrstraining.storefront.courses.controller;
2
3 import de.hybris.platform.acceleratorstorefrontcommons.annotations.RequireHardLogIn;
4
5 @Controller
6 @RequestMapping("/myaccount")
7 public class ChennaRRRSCoursesController
8 {
9     @Resource(name = "chennaRRRSCoursesFacade")
10    private ChennaRRRSCoursesFacade chennaRRRSCoursesFacade;
11
12    @RequestMapping(value = "/courses", method = RequestMethod.GET)
13    @RequireHardLogIn
14    public String getCourses(final Model model) throws CMSItemNotFoundException
15    {
16        System.out.println("Begin controller");
17        final List<ChennaRRRSCoursesData> chennaRRRSCoursesDetails = chennaRRRSCoursesFacade.get_coursesData();
18        System.out.println("Courses Details Count = " + chennaRRRSCoursesDetails.size());
19        model.addAttribute("chennaRRRSCoursesDetails", chennaRRRSCoursesDetails);
20
21        return ControllerConstants.Views.Pages.Account.CoursesPage;
22    }
23}
```

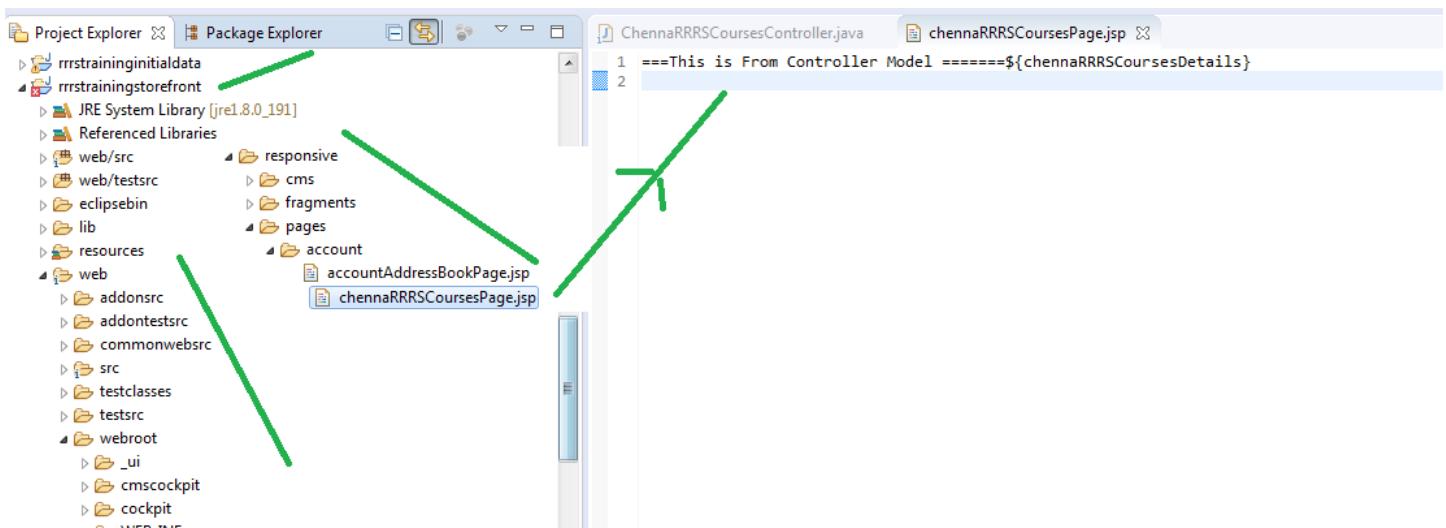
Contact Us = ChennaReddyTraining@RRRS.CO.IN

```

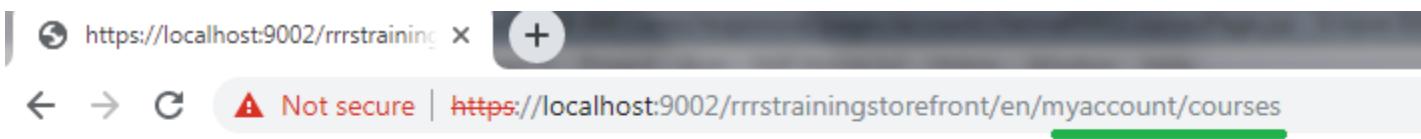
ChennaRRRSCoursesFacade.java ChennaRRRSCoursesController.java ControllerConstants.java
88     String AccountProfileEmailEditPage = "pages/account/accountProfileEmailEditPage"; // NOSONAR
89     String AccountChangePasswordPage = "pages/account/accountChangePasswordPage"; // NOSONAR
90     String AccountAddressBookPage = "pages/account/accountAddressBookPage"; // NOSONAR
91     String AccountEditAddressPage = "pages/account/accountEditAddressPage"; // NOSONAR
92     String AccountPaymentInfoPage = "pages/account/accountPaymentInfoPage"; // NOSONAR
93     String AccountRegisterPage = "pages/account/accountRegisterPage"; // NOSONAR
94
95     String CoursesPage = "pages/account/chennaRRRSCoursesPage"; // NOSONAR
96
97 }
98

```

Step 12 = Write the UI Layer (or) Presentation Layer



Results: -



====This is From Controller Model ===== [com.rrrs.rrrstraining.facades.courses.ChennaRRRSCoursesData com.rrrs.rrrstraining.facades.courses.ChennaRRRSCoursesData@3e4ddfa2, com.rrrs.rrrstraining.facades.courses.ChennaRRRSCoursesData@5b6de700, com.rrrs.rrrstraining.facades.courses.ChennaRRRSCoursesData@5b6de700]

Project Explor... Package Expl... ChenraRRRCoursesController.java chennaRRRCoursesPage.jsp

```

1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
2 <%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
3
4
5@:forEach items="${chennaRRRCoursesDetails}" var="chennaRRRCoursesData">
6 <br />
7 ${chennaRRRCoursesData.code} --- ${chennaRRRCoursesData.name} --- ${chennaRRRCoursesData.duration} --- ${chennaRRRCoursesData.amount}
8 </c:forEach>

```

https://localhost:9002/rrrstraining x +

← → C Not secure | https://localhost:9002/rrrstrainingstorefront/en/myaccount/courses

1011 --- Hybris --- 80 Hrs --- 500
 1012 --- Data Hub --- 50 Hrs --- 500
 1013 --- C Lang --- 60 Hrs --- 500
 Training1 --- Hybris --- 90 --- 500
 Training2 --- DataHub --- 55 --- 500
 Training3 --- Salesforce --- 60 --- 5000

```

public class ChenraRRRCoursesDao extends AbstractItemDao
{
    private static final String COURSES_QUERY = "SELECT {C.PK} FROM {ChennaRRRCourses AS C}";
    public List<ChennaRRRCoursesModel> getAllCourses()
    {
        final FlexibleSearchQuery query = new FlexibleSearchQuery(COURSES_QUERY);
        final SearchResult<ChennaRRRCoursesModel> result = getFlexibleSearchService().search(query);
    }
}

```

After Step 8 = Results

ChennaRRRCoursesData				ChennaRRRCoursesModel				
	Code	Name	Duration		Training1	Hybris	90	500
Training1	Hybris	90	500		Training1	Hybris	90	500
Training2	DataHub	55	500		Training2	Datahub	55	500
Training3	Salesfo	60	500		Training3	Salesforce	60	500

Step 10 = Results

Step 5 = Results

It contains All 3 Records

Step 3 = Results

Table = RRRSTrainingCourses

Code	Name	Duration	Amount
Training1	Hybris	90	500
Training2	Data Hub	55	500
Training3	Salesforce	60	500

Step 4 = Results

Database

Step 12 = Results

1011 --- Hybris --- 80 Hrs --- 500
 1012 --- Data Hub --- 50 Hrs --- 500
 1013 --- C Lang --- 60 Hrs --- 500
 Training1 --- Hybris --- 90 --- 500

Project Explor... Package Expl... ChenraRRRCoursesController.java chennaRRRCoursesPage.jsp

```

1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
2 <%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
3
4@:<table style="width:100%" border="2">
5@:<tr>
6    <th>Code</th>
7    <th>Name</th>
8    <th>Duration</th>
9    <th>Amount </th>
10   </tr>
11@:<c:forEach items="${chennaRRRCoursesDetails}" var="chennaRRRCoursesData">
12   <br />
13@:<tr>
14    <td> ${chennaRRRCoursesData.code} </td>
15    <td> ${chennaRRRCoursesData.name} </td>
16    <td>${chennaRRRCoursesData.duration}</td>
17    <td>${chennaRRRCoursesData.amount}</td>
18   </tr>
19 </c:forEach>
20 </table>

```

https://localhost:9002/rrrstraining | +

Not secure | https://localhost:9002/rrrstrainingstorefront/en/myaccount/courses

Code	Name	Duration	Amount
1011	Hybris	80 Hrs	500
1012	Data Hub	50 Hrs	500
1013	C Lang	60 Hrs	500
Training1	Hybris	90	500
Training2	DataHub	55	500
Training3	Salesforce	60	5000

Step 1 = Create DB Itemtype / Table

chenarrstrainingcore-items.xml

```
<itemtype code="ChennaRRRCourses" generate="true" >
<deployment table="RRRSTrainingCourses" >
</deployment>
</itemtype>
```

Step 2 = Do the build [ant clean all]

ChennaRRRCoursesModel.java [Generated]

```
ChennaRRRCoursesModel.java
15 /**
16 @SuppressWarnings("all")
17 public class ChennaRRRCoursesModel
18 {
```

Step 3 = Start Server ... "HAC - Update"

RRRSTrainingCourses [Table]

Code	Name	Duration	Amount	[Cols]
101	Comm1	80 Hrs	500	Obj1
102	Comm2	80 Hrs	500	Obj2
103	SFDC	60 Hrs	500	Obj3

Step 4 = Insert the records

```
insert ChennaRRRCourses;code;name;duration;amount
101;Comm1;80 Hrs ; 500
102;Comm2;80 Hrs ; 500
103;SFDC;60 Hrs ; 500
```

Results =

RRRSTrainingCourses [Table]

Code	Name	Duration	Amount
101	Comm1	80 Hrs	500
102	Comm2	80 Hrs	500
103	SFDC	60 Hrs	500

Step 5 = Create *DAO File [Write FSQL]

ChennaRRRCoursesDao.java

```
public class ChennaRRRCoursesDao extends AbstractItemDao
```

Results =

List<ChennaRRRCoursesModel> Obj

Code	Name	Duration	Amount
101	Comm1	80 Hrs	500
102	Comm2	80 Hrs	500
103	SFDC	60 Hrs	500

Step 6 = Register *DAO File in *-spring.xml

chenarrstrainingcore-spring.xml

```
<bean id="chenarrstrainingcore-spring" class="com.che...
```

Step 7 = Create Service Layer File

ChennaRRRCoursesService.java

```
ChennaRRRCoursesService
public List<ChennaRRRCoursesModel> getAllCourses()
{
    return chennaRRRCoursesDao.getAllCourses();
}
```

Results =

List<ChennaRRRCoursesModel> Obj

Code	Name	Duration	Amount
101	Comm1	80 Hrs	500
102	Comm2	80 Hrs	500
103	SFDC	60 Hrs	500

Step 8 = Register Service Layer File in *-spring.xml

chenarrstrainingcore-spring.xml

```
<bean id="chenarrstrainingcore-service" class="...
```

Step 9 = Create *Data Class (or) *Data Obj

For *Model Obj - We have *.items.xml
For *Data Obj - We have *.facades.xml

chenarrstrainingfacades-beans.xml

```
<bean class="com.chennarrs.che...
```

Step 10 = Do the build [ant clean all]

ChennaRRRCoursesData.java

```
public class ChennaRRRCoursesData
{
    private String code;
    private String name;
    private int duration;
    private int amount;
}
```

Step 11 = Write the Facade Layer File

ChennaRRRCoursesFacade.java

```
public class ChennaRRRCoursesFacade
{
    return getcoursesConverter().convertAll
```

Step 12 = Do below in *-spring.xml

chenarrstrainingfacades-spring.xml

```
<alias name="rrrDefaultSaveCartFacade" />
<bean id="rrrDefaultSaveCartFacade" class="...
```

Step 13 = Create Controller with request mapping.

URL = https://localhost:9002/yyy/en/myaccount/courses

ChennaRRRCoursesController.java

```
@Controller
@RequestMapping("/myaccount")
public class ChennaRRRCoursesController
{
    @RequestMapping(value = "/courses", method = RequestMethod.GET)
    public String getCourses(final Model model)
    {
        model.addAttribute("chennaRRRCoursesDetails");
    }
}
```

Results =

List<ChennaRRRCourses Data> Obj

Code	Comm1	80 Hrs	500	Obj1
102	Comm2	80 Hrs	500	Obj2
103	SFDC	60 Hrs	500	Obj3

Step 14 = Write the UI code

chennaRRRCoursesPage.jsp

```
<c:forEach items="${chennaRRRCoursesDetails}">
<br />
<tr>
<td> ${chennaRRRCoursesData.code} </td>
<td> ${chennaRRRCoursesData.name} </td>
<td> ${chennaRRRCoursesData.duration}</td>
<td> ${chennaRRRCoursesData.amount}</td>
</tr>
```

Results

/chenarrstrainingstorefront/en/myaccount/courses

Code	Name	Duration	Amount
101	Comm1	80 Hrs	500
102	Comm2	80 Hrs	500
103	SFDC	60 Hrs	500

Interceptors – Models in Hybris (Lifecycle)

Q: What is the purpose of Model classes (or) Model Obj?

Model Obj is the True representation of DB – Table record.

Model in Hybris has lifecycle managed by “**ModelService**”.

ModelService – Has the methods to perform CURD Operations.

Create ... Update ... Load ... Remove

Hybris offers Interceptors (Programs) that provides Hook to model lifecycle.

There are 5 types of Interceptors:

a. LoadInterceptor = Invoked whenever a model is loaded from DB

```
void onLoad (Object model, InterceptorContext ctx) throws  
InterceptorException;
```

b. InitDefaultsInterceptor = Invoked during modelService.create() &
modelService.initDefaults()

```
void onInitDefaults(Object model, InterceptorContext ctx) throws  
InterceptorException;
```

c. PrepareInterceptor = Invoked before model is saved to database
& before ValidateInterceptor

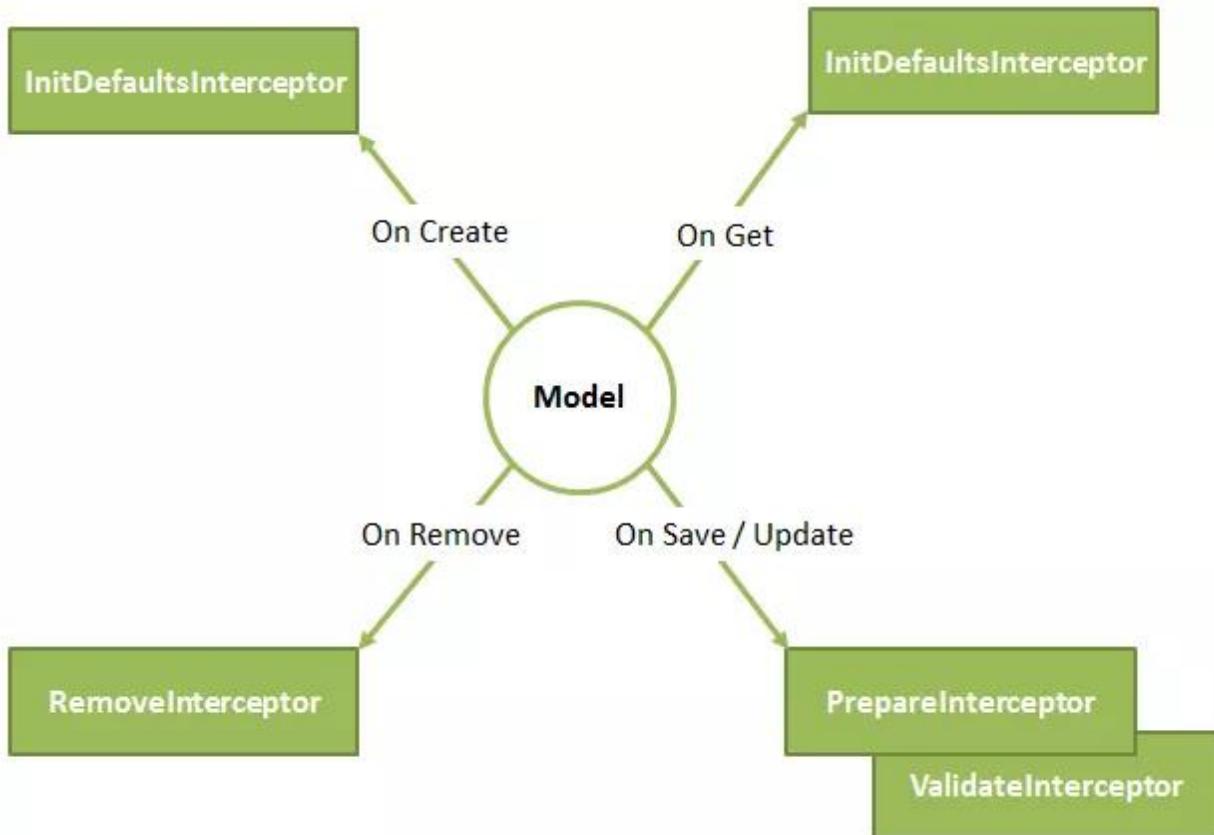
```
void onPrepare(Object model, InterceptorContext ctx) throws  
InterceptorException;
```

d. ValidateInterceptor = Invoked before a model is saved to
database & after PrepareInterceptor

```
void onValidate(Object model, InterceptorContext ctx) throws  
InterceptorException;
```

e. **RemoveInterceptor** = Invoked before a model is removed from the DB.

```
void onRemove(Object model, InterceptorContext ctx) throws  
InterceptorException;
```



These models we use in Hybris has its own lifecycle managed by **ModelService**.

There are several methods provided by **ModelService** to represent its lifecycle.

Example: Create, Save, Update, Load, Remove.

When we call any of these **ModelService** method, it just follows its Lifecycle and completes the execution.

Example: Save method save the model data in Database.

Contact Us = ChennaReddyTraining@RRRS.CO.IN

Q = Why we need interceptors?

The interceptors have the ability to interrupt the **lifecycle of a model** and execute some code and then allow the model to continue its lifecycle.

Q = When we use interceptors?

We need to use interceptors whenever we want to perform some action during the life cycle of a model.

Example: If we need to validate the model data before saving in database

We can use interceptors in some scenarios as mentioned below: -

1. Auditing the details while saving the model data
2. Validating the model data before saving
3. Restrict data from deleting or modifying based on some conditions

We can even modify the model before saving it using interceptors.

Note = Each of above interceptor addresses a particular step of the model life cycle.

When the life cycle of a model reaches a certain step, a corresponding interceptor will be executed automatically.

Note = Remove interceptor is represented by interface called **“RemoveInterceptor”**

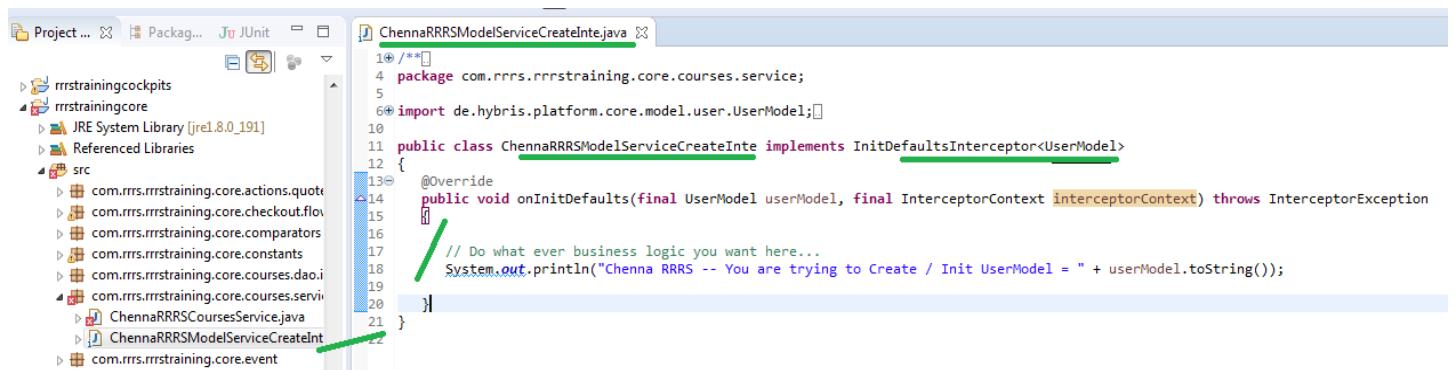
This is called before a model is **removed/deleted** from the database.

We can use remove interceptor whenever we need to remove models that are related to the model but are not in the model context.

Example for “Create – InitDefaultsInterceptor”

Step 1 = Create a class “**ChennaRRRSModelServiceCreateInte**” & implement interface called “**InitDefaultsInterceptor**”.

InitDefaultsInterceptor – This interface having the methods & override method called “**onInitDefaults()**”



```
1 // ...
2 package com.rrrs.rrrstraining.core.courses.service;
3
4 import de.hybris.platform.core.model.user.UserModel;
5
6 public class ChennaRRRSModelServiceCreateInte implements InitDefaultsInterceptor<UserModel>
7 {
8     @Override
9     public void onInitDefaults(final UserModel userModel, final InterceptorContext interceptorContext) throws InterceptorException
10    {
11        // Do what ever business logic you want here...
12        System.out.println("Chenna RRRS -- You are trying to Create / Init UserModel = " + userModel.toString());
13    }
14 }
```

Step 2 = Register “**ChennaRRRSModelServiceCreateInte**” as a bean in ***-spring.xml**



```
555     
```

```
556     
```

```
557     <bean id="chennaRRRCoursesService" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRCoursesService" >
558     </bean>
559
560
561     <bean id="chennaRRRSModelServiceCreateInte" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRSModelServiceCreateInte">
562         <!-- inject beans here if needed -->
563     </bean>
564
565
566
567
568
569
570
571
```

Step 3 = Declare & Configure your interceptor using "InterceptorMapping".

The screenshot shows the Eclipse IDE interface with the project structure on the left and the file 'rrstrainingcore-spring.xml' open in the center. The code defines several beans, including 'chennaRRRCoursesService' and 'chennaRRRSModelServiceCreateInte'. A specific bean definition for 'chennaRRRSModelServiceCreateInte' is highlighted, which includes an 'InterceptorMapping' configuration. A green checkmark is placed next to the 'chennaRRRSModelServiceCreateInte' bean definition.

```
555     </bean>
556
557     <bean id="chennaRRRCoursesService" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRCoursesService" >
558         </bean>
559
560
561     <bean id="chennaRRRSModelServiceCreateInte" class="com.rrrs.rrrstraining.core.courses.service.ChennaRRRSModelServiceCreateInte">
562         <!-- inject beans here if needed -->
563     </bean>
564
565
566     <bean id="chennaRRRSModelServiceCreateInteMapping" class="de.hybris.platform.servicelayer.interceptor.impl.InterceptorMapping">
567         <property name="interceptor" ref="chennaRRRSModelServiceCreateInte"/>
568         <property name="typeCode" value="User"/>
569     </bean>
570
571 <import resource="/rrrstrainingcore/processes/quote/quote-buyer-process-spring.xml"/>
```

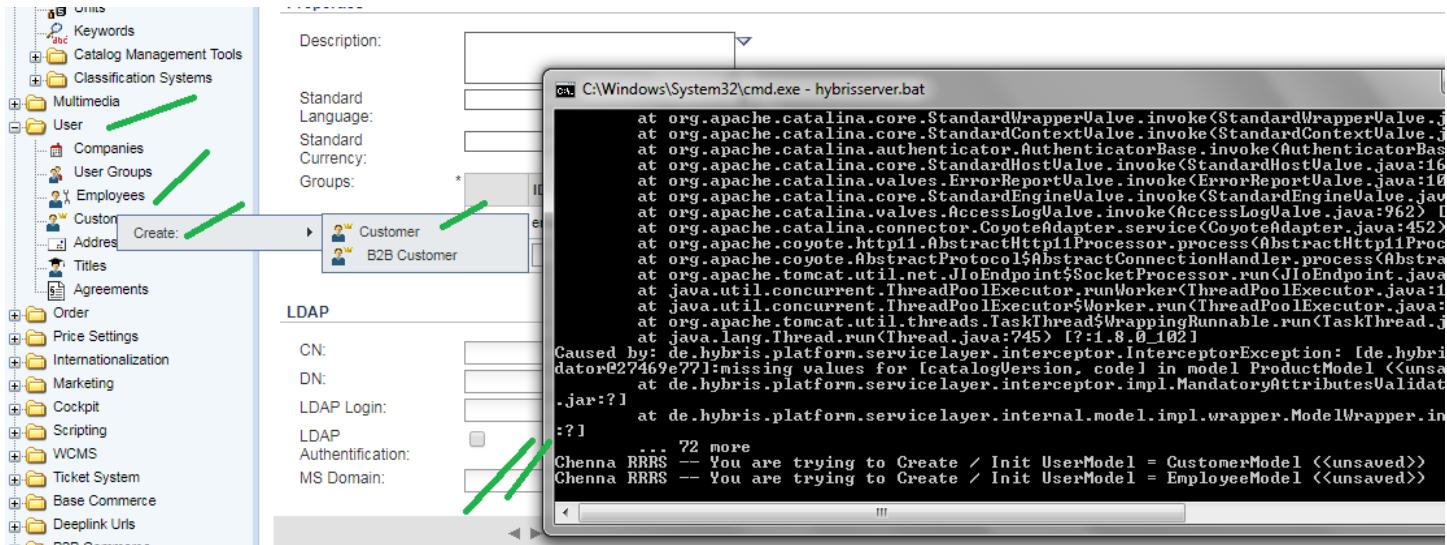
Step 4 = In any class ... Whenever you create "Use Model" instance then you can see "Our Interceptor" will be executed.

The screenshot shows the Eclipse IDE interface with the project structure on the left and two files open: 'ChennaRRRSModelServiceCreateInte.java' and 'ChennaRRRCoursesService.java'. The code in 'ChennaRRRSModelServiceCreateInte.java' contains comments explaining how to invoke an interceptor. The code in 'ChennaRRRCoursesService.java' shows the actual implementation where an interceptor is being used. Two green arrows point from the comments in the first file to the interceptor call in the second file.

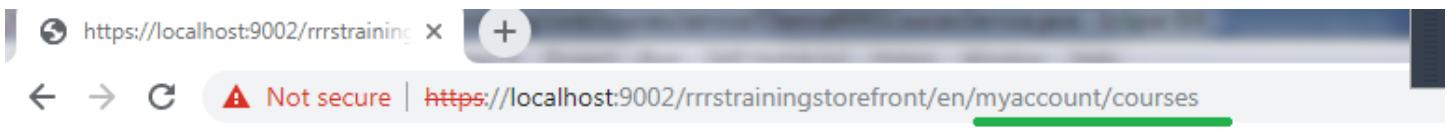
```
13 public class ChennaRRRCoursesService
14 {
15     @Resource
16     private ChennaRRRCoursesDao chennaRRRCoursesDao;
17
18     @Resource
19     private ModelService modelService;
20
21     public List<ChennaRRRCoursesModel> getAllCourses()
22     {
23
24         // 1. This is one way to invoke the InitDefaultInterceptor
25         //final UserModel newUser1 = new UserModel();
26         //modelService.initDefaults(newUser1);
27
28         // 2. This is another way final to invoke final the InitDefaultInterceptor
29         final UserModel newUser2 = modelService.create(UserModel.class);
30
31     }
32 }
```

Step 5 = Test the Results: -

hMC / Backoffice: -



(or)



Code	Name	
1011	Hybris	80 Hrs
1012	Data Hub	50 Hrs
1013		
Training 1		
Training 2		
Training 3		

C:\Windows\System32\cmd.exe - hybrisserver.bat

```
... 72 more
Chenna RRRS -- You are trying to Create / Init UserModel = CustomerModel <<unsaved>>
Chenna RRRS -- You are trying to Create / Init UserModel = EmployeeModel <<unsaved>>
INFO [TaskExecutor-master-263-ProcessTask [8796715680694]] [CalculatePersonalizationFor
mail.com
Wrapper Process has not received any CPU time for 101 seconds. Extending timeouts.
WARN [hybrisHTTP9] [RequireHardLoginEvaluator] missing secure token in session, login
WARN [hybrisHTTP9] [RequireHardLoginBeforeControllerHandler] Redirection required
INFO [hybrisHTTP7] [DefaultGUIDCookieStrategy] Setting guid cookie and session attrib
Begin controller
Chenna RRRS -- You are trying to Create / Init UserModel = UserModel <<unsaved>> ✓
Courses Details Count = 6
Begin controller
Chenna RRRS -- You are trying to Create / Init UserModel = UserModel <<unsaved>> ✓
Courses Details Count = 6
Begin controller
```

Note: - LoadInterceptor this interface is having “onload()” method override it. modelService.get(XXX) calls above override method.