**Project Template Guide Line 2021**

Migration

1. Get copy of Project Template zip file in to project folder.
2. Extract zip in project folder
3. Change folder name <ProjectTemplate> in to <YourDomainName>
4. Same as change bitproject.mwb file name into <YourDomainName>.mwb in <YourDomainName> folder
5. Then double click .mwb file then it will open
6. Change physical schema name into <yourdomainname>
7. Then go menu bar Database 🡪 Forward Engineer 🡪 Check connection 🡪 then click Next button 🡪 then tick Generate Insert statement for table option then click Next 🡪 Next 🡪 Next 🡪 Finish
8. After Forward Engineer Then go to menu bar Database🡪 connect to Database 🡪 check connection detail 🡪 click ok 🡪 verify the database if create or not
9. Open IntelliJ Idea tool then open bitproject project folder using open option.
10. It will take several minutes for download and config dependency file. After complete that configuration it will allow to Run the project.
11. Change application.property file

“spring.datasource.url = jdbc:mysql://localhost:**3308**/**bitproject**?useSSL=true”

**3308**/**bitproject change into your port number and DB name**

“spring.datasource.password = **12345**”

Change password “**12345**” to your server password

1. Change PrivilageRepository File

“select bit\_or(sel) sel, bit\_or(ins) ins, bit\_or(upd) upd, bit\_or(del) del from **bitproject**.privilage where roles\_role\_id in (select role\_id from **bitproject**.user\_role where user\_id=(select user\_id from **bitproject**.users where user\_name = ?1)) and module\_id=(select id from **bitproject**.module where name= ?2);”

**bitproject into your DB name**

1. Then Run the Project
2. Open Browser then change URL into (localhost:8080/login). It will open login page .
3. Login using admin details (username = admin | password = 12345) click login button
4. Then load mainwindow

Phase 1

(Start New Module[M])

1. Get copy of ER 🡪 copy <YourProject>M<N> .mwb 🡪 past into back up folder
2. Get DB Backup -- > Open Workbench. Under Management-->Data Export-->Select Database relevant to your project. Tick it.
3. Check weather "Export to self contained file" is selected.
4. Browse to backup folder then add name as <YourProject>M <n> .sql --> Save.
5. Check Weather "Include Create Schema" checkbox is selected 🡪 Click Button "Start Export".
6. Click "Continue Anyway", if any warning is appeared.
7. Right click the project folder in "Current-Module" in Project Folder and copy. Then Paste back up folder, re name it as <YourProject> M <N>.
8. Open "Current-Module" Folder in Project Folder.
9. Rename ER as <YourProject> M <N>+1. Then Open it.
10. Add the relevant tables of the new sprint.
11. Change Order of the attributes to be used in the form.

Phase 2

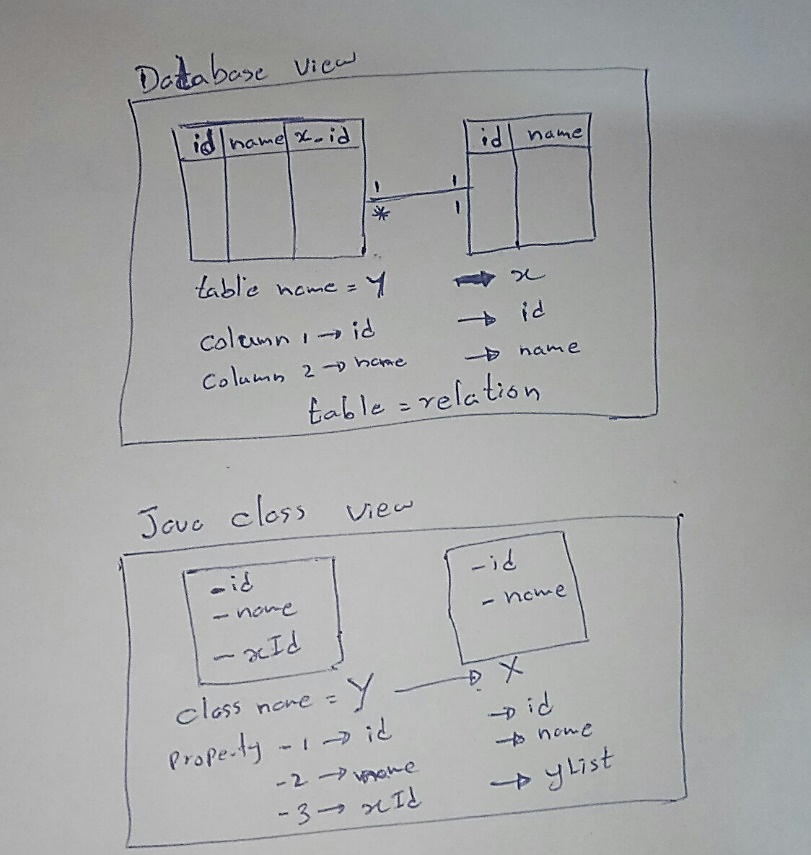
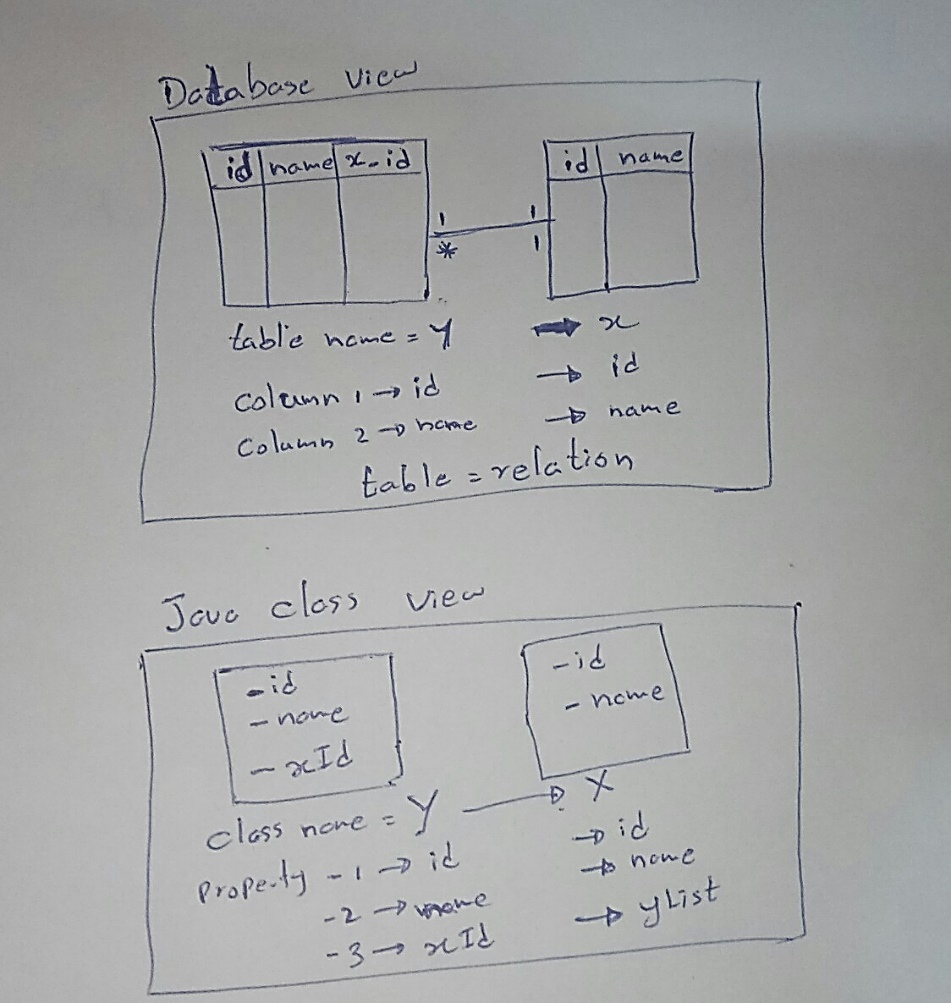
**(DB-Update, View Data, Define Table and Fill-Data)**

(A) DB-Update

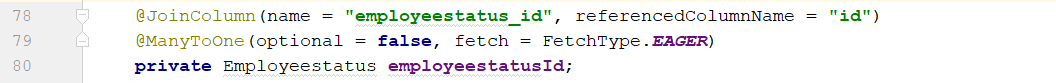
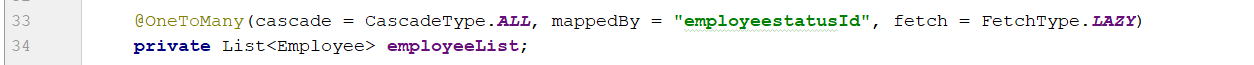
1. Menu Bar-->Database-->Synchronize Model-->Next-->Next-->Next-->Next-->Next-->Execute--> Close
2. Database-->Connect database-->ok (verify new table exist or not)
3. In the Navigator Pane, Under Schemas add data to supportive tables. Click apply Function.
4. Insert data into Master Tables.
5. Observe data in the tables using MySQL Workbench Refresh Function.

(B) View data

Create entity classes (model classes for database table (ORM))



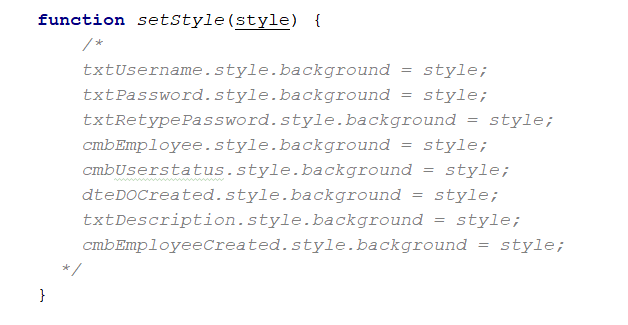
1. Right click model package 🡪 new 🡪 java class (create java class for each mapping for db table)
2. Class map for table (use @Entity @Table(name=”y”) )
3. Property map for column ( @Column(name=”id”))
4. Property for Foreign key map ( @ManytoOne and @OnetoMany)

Ex:

1. Add back end validation (@Pattern or hibernate validation annotation)
2. Right click controller package---->New---> java Class
3. Under Name "<NewModuleName>Controller"------> Click OK
4. Tpye Annotation @RestController over "public class" and add Annotation @RequestMapping(value=”/employee”)
5. Copy FindAll @GetMapping(value=”/findAll”) method in EmployeeController----->past into "<NewModuleName>Controller"
6. Unselect entity.Employee----->OK
7. Replace Employee-----> into "<NewModuleName>"
8. Right click repository package---->New---> java Class
9. Under Name "<NewModuleName>Repository"------> Change type to "Inteface" ----- >Click OK
10. extend JpaRepoitory<"<NewModuleName>", Integer>
11. In side the Controller class Type @Autowired Annotation-----> Press enter ----> type "private <YourModuleName> Repository dao;"
12. Open webSecurityConfiguration class file add permission for Roles
13. Stop and Run the project
14. Login to system
15. type url[http://localhost:8080/<NewModuleName>/FindAll?page=0&size=1](http://localhost:8080/%3cNewModuleName%3e/FindAll?page=0&size=1)

(C) Define Table and Fill-Data

----------------------------------------------------------

1. Create the UI needed for the Module
   1. Copy employee.html, employee.js and past it as your new module
   2. Change the Panel Title to the new Module and rename <title> tag into <your project name>
   3. comment all function inner codes
2. Update “UiController” to deliver the UI for your new model
3. Give permission for roles
4. Run the Project
5. **Test-6** : Request URL for the UI ( http://localhost:8080/newmodel)
6. Define the Table with Pagination (Direct Properties, Object Properties, Derived Data🡨 Function)
7. Comment Initialize() 🡪 loadForm()
8. UnComment loadView function
9. uncomment loadTable
10. Load and Fill data into the Table
11. **Test-7** : Request URL for the UI with the table that would filled with data

Phase-3

**(Define Form, Load-Data, Binding with Validation using Regex and check errors and Add)**

(A) load data service for combobox

1. Define Repository and Get mapping methods (for get list) with Constructors in the Entity Classes for Supportive Tables (Controller🡪 Repository🡪Entity)
2. Update the relevant “Controller” to provide needed data. Use @Autowire annotation for construct to repository for get object
3. Give permission for these services for each Roles
4. **Test-2** : Request URL “http://localhost:8080/<>/list” (userstatus, employee)

(B) Define Form

1. Create the Form add suitable Event Handler
2. Update initialize function by registering Event-Handlers with operational buttons if needed
3. Get Privilages
4. Load the Form with Supportive Data and fill them with Combo Box
5. Combo Must be loaded with Supportive Data
6. Define SetStyle()
7. dissableButton() 🡨 Do not need to adjust

(C) Filter method no binding

1. **Intearctive Componnets must also load (Category🡪Subcategory)**
2. **Test-4** : Validation Color Changes (Custom “Validation and Data-Binding” may be need to defined and register in “initialize()”)

(D) Get error and Add button client side

1. btnAdd🡪 Binding Test (Comment // var errors = getErrors(); // if(errors=="") { if(option==true) { -> To End of Method------------ (error in here)
2. Test 🡪 Confirmation Message 🡪 Adjust the Confirm Method
3. Implement getErrors()
4. Uncomment “btnAdd” test for getErrors() (One by One, Errors will be gone, Finally Confirmation )
5. Update POST-Request in “btnAdd”

(E) add() in server side

1. Define add() with @Post mapping in <NewModel>Controller in Server side (Copy from the Template, Find and Replace Ex: employee with <NewModel>)
2. Do needed validation (@validate will do it by spring-validation-framwork) (Change Status Code for Errors if Possible)
3. Fill the Form add One Item and Observe it both in View and the Database

Phase-4

(**fillForm(), colorchange and getUpdate() with massage, btnUpdate() client side, server side Controler class , btnDelete())**

* + 1. FillForm()

1. Comment content of the getUpdates()
2. Implement FillForm
3. Test fillForm
   * 1. colorchange, getUpdate() with massage
4. Uncomment getUpdates()
5. Find and Replace (employee->item)
6. Comment (// if(option==true))
7. Observe Nothing Updated and then Change One by One and Observe the Updated Message
   * 1. Btnupdate B-6 uncomment with controller and dao
8. Adjust the PUT request
9. Comment loadSearchTable(), add loadView()
10. Define the update in <Newmodel>Controller
11. Test for Updates
    * 1. Delete
12. “Delete function” 🡪 Change the HTTP-Request
13. Clear Correction

Phase-5

(Search)

1. Create search UI in html file
2. Add event handler for search field and bind btnSearchMC
3. Create btnSearchMC function
4. Define Get Mapping for get data to fill table with search value in Controller class
5. Define Query for search with any text

Phase-6

(Table Row print and table print)

1. Create UI for get view of table row data in <module name>.html
2. Create *viewitem* function in <module name>.js file
3. Create btnPrintMC fuction in <module name>.js file
4. Create *btnPrintTableMC* function for get table print

**Start new Sprint, Following Sprint Guide line**