

1.3 Application Developer Area	5
1.1.3.1 About finch	5
1.2.3.2 Standardization	8
1.2.1.3.2.1 UI Standards and Guidelines	8
1.2.1.1 3.2.1.1 Login Screen	11
1.2.1.2 3.2.1.2 Dashboard	12
1.2.1.3 3.2.1.3 Wizard	13
1.2.1.4 3.2.1.4 Query	16
1.3.3.3 Design Phase	19
1.3.1.3.3.1 Software Library	20
1.3.2.3.3.2 Data Model Design	22
1.3.2.1 3.3.2.1 Guidelines	23
1.3.2.2 3.3.2.2 finch Data Model	23
1.3.2.3 3.3.2.3 Infrastructure (INF) Data Model	23
1.3.2.3.1 INF_ACCESS_LOG	23
1.3.2.3.2 INF_ACTION	23
1.3.2.3.3 INF_ACTION_CONSOLIDATION	23
1.3.2.3.4 INF_ACTION_ENTERPRISE_PCPT	23
1.3.2.3.5 INF_ACTION_ROLE_PARTICIPANT	24
1.3.2.3.6 INF_APPLICATION_DATE	24
1.3.2.3.7 INF_APPLICATION_ROLE	24
1.3.2.3.8 INF_BATCH_MASTER	24
1.3.2.3.9 INF_BATCH_ROLE_PCPT	24
1.3.2.3.10 INF_BATCH_RUNNER_CONFIG	25
1.3.2.3.11 INF_BRANCH	25
1.3.2.3.12 INF_BRANCH_NAME_XREF	27
1.3.2.3.13 INF_CALENDAR	27
1.3.2.3.14 INF_CLOSING_STATUS	27
1.3.2.3.15 INF_COMPONENT	27
1.3.2.3.16 INF_CONSTRAINT_LIST	28
1.3.2.3.17 INF_CONSTRAINT_VALUES	28
1.3.2.3.18 INF_EMP_APPLN_ROLE_PARTICIPANT	28
1.3.2.3.19 INF_EMPLOYEE	28
1.3.2.3.20 INF_EMPLOYEE_GROUP_PCPT	29
1.3.2.3.21 INF_EMPLOYEE_NAME_XREF	29
1.3.2.3.22 INF_ENDPOINT	29
1.3.2.3.23 INF_ENTERPRISE	29
1.3.2.3.24 INF_ENTERPRISE_NAME_XREF	32
1.3.2.3.25 INF_FILE_DIRECTORY	32
1.3.2.3.26 INF_GROUP	32
1.3.2.3.27 INF_HOLIDAY	32
1.3.2.3.28 INF_LANGUAGE	33
1.3.2.3.29 INF_NOTIFICATION	33
1.3.2.3.30 INF_NOTIFICATION_ATTACHMENT	33
1.3.2.3.31 INF_NOTIFICATION_DESTINATION	33
1.3.2.3.32 INF_NOTIFICATION_SERVER	34
1.3.2.3.33 INF_PASSWORD_HISTORY	34
1.3.2.3.34 INF_QRTZ_BLOB_TRIGGERS	34
1.3.2.3.35 INF_QRTZ_CALENDARS	34
1.3.2.3.36 INF_QRTZ_CRON_TRIGGERS	35
1.3.2.3.37 INF_QRTZ_FIRED_TRIGGERS	35
1.3.2.3.38 INF_QRTZ_JOB_DETAILS	35
1.3.2.3.39 INF_QRTZ_LOCKS	36
1.3.2.3.40 INF_QRTZ_PAUSED_TRIGGER_GRPS	36
1.3.2.3.41 INF_QRTZ_SCHEDULER_STATE	36
1.3.2.3.42 INF_QRTZ_SIMPLE_TRIGGERS	36
1.3.2.3.43 INF_QRTZ_SIMPROP_TRIGGERS	36
1.3.2.3.44 INF_QRTZ_TRIGGERS	37
1.3.2.3.45 INF_REPORT	37
1.3.2.3.46 INF_REPORT_ENTERPRISE_PCPT	38
1.3.2.3.47 INF_REPORT_GENERATION_LOG	38
1.3.2.3.48 INF_ROUTE	38
1.3.2.3.49 INF_ROUTE_ENDPOINT_PARTICIPANT	38
1.3.2.3.50 INF_SAVED_QUERY	39
1.3.2.3.51 INF_SAVED_TEMPLATE	39
1.3.2.3.52 INF_SERVICE	39
1.3.2.3.53 INF_SERVICE_ROUTE_PARTICIPANT	39
1.3.2.3.54 INF_UI_MENU	39
1.3.2.3.55 INF_UI_PREF_GROUP	40
1.3.2.3.56 INF_UI_PREF_GROUP_NAME_XREF	40
1.3.2.3.57 INF_UI_SCREEN	40
1.3.2.3.58 INF_UPLOAD_TEMPLATE	41

1.3.2.3.59 INF_UPLOAD_TMPL_MAPPING	41
1.3.2.4 3.3.2.4 Dashboard (DBD) Data Model	41
1.3.2.4.1 DBD_CHART	41
1.3.2.4.2 DBD_CHART_APPLN_ROLE_PTCP	41
1.3.2.4.3 DBD_CUSTOM_CHART	42
1.3.2.4.4 DBD_CUSTOM_FEED	42
1.3.2.4.5 DBD_CUSTOM_NOTIFICATION	42
1.3.2.4.6 DBD_CUSTOM_SAVED_QUERY	42
1.3.2.4.7 DBD_CUSTOM_SAVED_TEMPLATE	43
1.3.2.4.8 DBD_EMP_TASK_TYPE_CONFIG	43
1.3.2.4.9 DBD_FEED	43
1.3.2.4.10 DBD_FEED_APPLN_ROLE_PTCP	44
1.3.2.4.11 DBD_FEED_NAME_CROSS_REF	44
1.3.2.4.12 DBD_GROUP	44
1.3.2.4.13 DBD_MENU_SHORTCUT	44
1.3.2.4.14 DBD_NOTIF_DSTN_EMP_PCPT_PK	44
1.3.2.4.15 DBD_ROUTINE_TASK	45
1.3.2.4.16 DBD_TASK_DETAIL	45
1.3.2.4.17 DBD_TASK_STATUS	45
1.3.2.4.18 DBD_TASK_TYPE	45
1.3.2.4.19 DBD_TASK_TYPE_EMP_PTCP	46
1.3.2.4.20 DBD_TEMP_QUERY_PARAM	46
1.3.2.4.21 DBD_WIDGET	46
1.3.2.4.22 DBD_WIDGET_EMP_PTCP	47
1.3.2.4.23 DBD_WIDGET_FEED_PARTCPNT	47
1.3.2.4.24 DBD_WIDGET_TYPE	47
1.3.2.4.25 DBD_WIDGET_TYPE_APP_ROLE_PT	47
1.3.2.4.26 DBD_WIDGET_TYPE_EMP_CONFIG	48
1.3.2.4.27 DBD_WORKSPACE	48
1.3.2.4.28 DBD_WORKSPACE_EMP_PTCP	48
1.3.2.4.29 DBD_WORKSPACE_WIDGET_PTCP	48
1.3.2.5 3.3.2.5 Batch Tables	49
1.3.3 3.3.3 MVC/UI Layer	49
1.3.3.1 3.3.3.1 Menu and view Definition	49
1.3.3.2 3.3.3.2 Menu Shortcut Configuration Guide	50
1.3.3.3 3.3.3.3 Screen Design	51
1.3.3.3.1 3.3.4.1 Single Page Entry	62
1.3.3.3.2 3.3.4.2 Single Page Cancel	105
1.3.3.3.3 3.3.4.3 Single Page Amend	117
1.3.3.3.4 3.3.4.4 Wizard Based Entry	142
1.3.3.3.5 3.3.4.5 Wizard Based Amend	166
1.3.3.3.6 3.3.4.6 Query/Query Result	181
1.3.4 3.3.4 Domain Layer	218
1.3.4.1 3.3.4.1 Service	218
1.3.4.2 3.3.4.2 Repository	221
1.3.5 3.3.5 Concepts	222
1.3.5.1 3.3.5.1 Application Date	223
1.3.6 3.3.6 Batch/Console Service	224
1.3.6.1 3.3.6.1 Console Service Process	226
1.4 3.4 Implementation Phase	226
1.4.1 3.4.1 Getting started	227
1.4.1.1 3.4.1.1 Prerequisite	227
1.4.1.2 3.4.1.2 Set-up Blank Application using finch	228
1.4.1.3 3.4.1.3 finch Deliverables	237
1.4.1.4 3.4.1.4 Initial data set up for finch	238
1.4.1.5 3.4.1.5 CRUD Generation Tool	243
1.4.1.5.1 3.4.1.5.1 CREATE (Entry Template)	248
1.4.1.5.2 3.4.1.5.2 READ (Query Template)	253
1.4.1.5.3 3.4.1.5.3 UPDATE (Amend Template)	254
1.4.1.6 3.4.1.6 Set-up Finch Sample Application	255
1.4.1.7 3.4.1.7 Set-up Sample Application	263
1.4.1.8 Add a new module	267
1.4.2 3.4.2 Setup Development Environment	268
1.4.2.1 3.4.2.1 Environment Setup	268
1.4.2.2 3.4.2.2 Environment Details	270
1.4.2.3 3.4.2.3 GIT Setup	270
1.4.2.3.1 3.4.2.3.1 Command Line Interface (CLI)	276
1.4.2.3.2 3.4.2.3.2 E-GIT	281
1.4.2.4 3.4.2.4 JRebel Configuration	282
1.4.3 3.4.3 Setup Database	288
1.4.3.1 3.4.3.1 finch DB	288
1.4.3.2 3.4.3.2 Application DB	288

1.4.4 3.4.4 Bootstrap Application	288
1.4.4.1 3.4.4.1 Maven Archetypes	289
1.4.4.2 3.4.4.2 Add Module	289
1.4.5 3.4.5 Code Generation Using Tool	293
1.4.6 3.4.6 Feature Wise Implementation	293
1.4.6.1 3.4.6.1 General Feature	293
1.4.6.1.1 3.4.6.1.1 Data Persistence Abstraction	294
1.4.6.1.2 3.4.6.1.2 Rule Engine	295
1.4.6.1.3 3.4.6.1.3 Scheduler	303
1.4.6.1.4 3.4.6.1.4 Application Access Restriction	312
1.4.6.1.5 3.4.6.1.5 Preference	313
1.4.6.1.6 3.4.6.1.6 Notification	314
1.4.6.1.7 3.4.6.1.7 Exception Handling Framework	321
1.4.6.1.8 3.4.6.1.8 Add a new module	323
1.4.6.1.9 3.4.6.1.9 Configure an Online/Batch Report	324
1.4.6.1.10 3.4.6.1.10 Enterprise specific db schema management	328
1.4.6.1.11 3.4.6.1.11 Enterprise specific implementation/customization	330
1.4.6.1.12 3.4.6.1.12 Handle of Concurrent operation	334
1.4.6.1.13 3.4.6.1.13 Manage Transaction	335
1.4.6.1.14 3.4.6.1.14 Multilingual i18n Support	336
1.4.6.1.15 3.4.6.1.15 New Batch Process	343
1.4.6.2 3.4.6.2 Web Features	352
1.4.6.2.1 3.4.6.2.1 Dashboard	352
1.4.6.2.2 3.4.6.2.2 Entry/Amend Wizard Template	366
1.4.6.2.3 3.4.6.2.3 Query/Query Result	376
1.4.6.2.4 3.4.6.2.4 Reference Data Popup	401
1.4.6.2.5 3.4.6.2.5 Message Notification	415
1.4.6.2.6 3.4.6.2.6 i18n support	417
1.4.6.2.7 3.4.6.2.7 Implement Excel/CSV Upload Screen	425
1.4.6.2.8 3.4.6.2.8 Customize User Preference	431
1.4.6.2.9 3.4.6.2.9 UI Online Report	434
1.4.6.2.10 3.4.6.2.10 finch Theme Support	439
1.4.6.2.11 3.4.6.2.11 Bulk Action Wizard	441
1.4.6.2.12 3.4.6.2.12 UI Data Exchange	446
1.4.6.2.13 3.4.6.2.13 Referer Filter	447
1.4.6.2.14 3.4.6.2.14 Office Closing	448
1.4.6.2.15 3.4.6.2.15 Batch with Web UI	449
1.4.6.2.16 3.4.6.2.16 Form field focus on error	451
1.4.6.2.17 3.4.6.2.17 Add new Menu/Action	453
1.4.6.2.18 3.4.6.2.18 Authentication	461
1.4.6.2.19 3.4.6.2.19 Access Control	462
1.4.6.2.20 3.4.6.2.20 Application Role management	463
1.4.6.2.21 3.4.6.2.21 Default Dropdown Values	464
1.4.6.2.22 Entry Screen Customization	467
1.4.6.2.23 3.4.6.2.23 Add a new Theme	509
1.4.6.2.24 3.4.6.2.24 Handle Constraint List & Values	509
1.4.6.2.25 3.4.6.2.25 UI Component	512
1.4.6.2.26 3.4.6.2.27 Entry Screen Customization in finch	572
1.4.6.3 3.4.6.3 Console Features	597
1.4.6.3.1 3.4.6.3.1 Service Process	597
1.4.6.3.2 3.4.6.3.2 Batch Process	606
1.4.6.3.3 3.4.6.3.3 Batch Report Using Jasper	615
1.4.6.3.4 3.4.6.3.4 Password Encryption Batch	632
1.4.7 3.4.7 Unit Testing	634
1.4.7.1 3.4.7.1 Testing Guidelines	650
1.5 3.5 Testing Phase	650
1.5.1 3.5.1 Functional Testing	650
1.5.2 3.5.2 Non Functional Testing	655
1.5.3 3.5.3 Automation	662
1.5.4 3.5.5 Analysis Report From Sonar	678
1.5.4.1 3.5.5.1 Narrowing the Focus	682
1.5.4.2 3.5.5.2 Creating Coding Rules	684
1.5.4.3 3.5.5.3 Sonar Report	684
1.6 3.6 Frequently Asked Questions (FAQ)	691
1.6.1 3.6.1 General Features	692
1.6.2 3.6.2 UI - General	692
1.6.3 3.6.3 Console	692
1.6.4 3.6.4 UI - Dashboard	692
1.6.5 3.6.5 Transaction & Persistence	692
1.6.6 3.6.6 Data Model	692
1.6.7 3.6.7 Unit Testing	692
1.6.8 3.6.8 WAS	692

1.6.9 3.6.9 JMS	693
1.6.10 3.6.10 Logging	693
1.7 3.7 Version Upgrade	693

3. Application Developer Area

Application Developer Area – Introduction

finch provides a framework which can be used by the other applications. While developing an application the application developers uses this finch framework as the base and develops the other features as per the requirement. This section is dedicated for the application developers who all will be developing their application using the finch framework.



Generic Information for Application Developers

The page is titled "Generic Information for Application Developers". It contains a list of expandable sections, each with a plus sign icon:

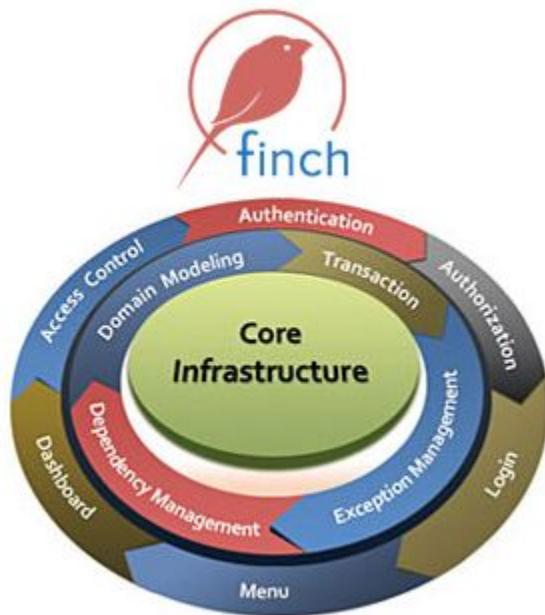
- About finch
- Standardization
- Version Upgrad e
- F A Q
- Design Phase
- Implementation Phase
- Testing Phase

3.1 About finch

Introduction

finch framework can be used in any new enterprise system development projects to develop different features for the application. This framework provides own infrastructural features comprising various common functionality like Dashboard, Service and Batch Framework, Authentication Authorization etc.

- Objective - develop an infrastructural framework for rapid enterprise application bootstrap
- The framework should built using latest industry standard open source technologies



Brief Overview of Features

finch has the following features:

finch Features

Web Features

Dashboard

Query

UI Message
Notification

Excel Upload

UI Online Report

Bulk Action
Wizard

Entry/Amend
Wizard Template

Reference Data
Popup

i18n support

Customize User
Preference

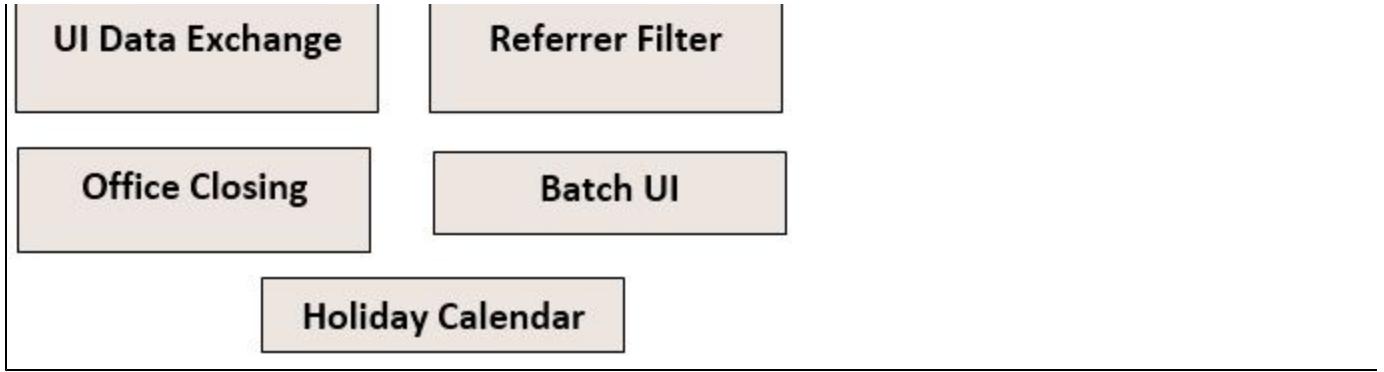
finch Theme
Support

Console Features

Service Process

Console Batch Process

Console Report



Value Addition of finch

Application Stakeholder

- Reduced Application Development Cost & Time
- Follows Agile Process. Roadmap can be aligned with clients' requirements

Developer

- In-house support & faster turn around
- Out-of-the-box functionalities for common uses cases
- Sample application for reference

End User

- Rich User Experience
- Consistent look-n-feel
- Multiple browser support
- Multiple DB support(Oracle,MySQL,DB2)
- Multiple Application Server Support(Apache Tomcat 6.x / 7.x,WebSphere Application Server 8.5.5,jBoss 7.x)

Limitations of finch

- Currently finch does not provide any out of box features to develop mobile application.

3.2 Standardization

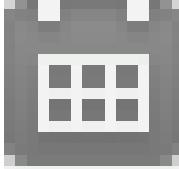
3.2.1 UI Standards and Guidelines

Introduction

finch has flexible and extensible support for UI. It offers many UI components like dashboard, menu, wizard, query etc. out of the box; these can also be customized by the application according to the requirement.

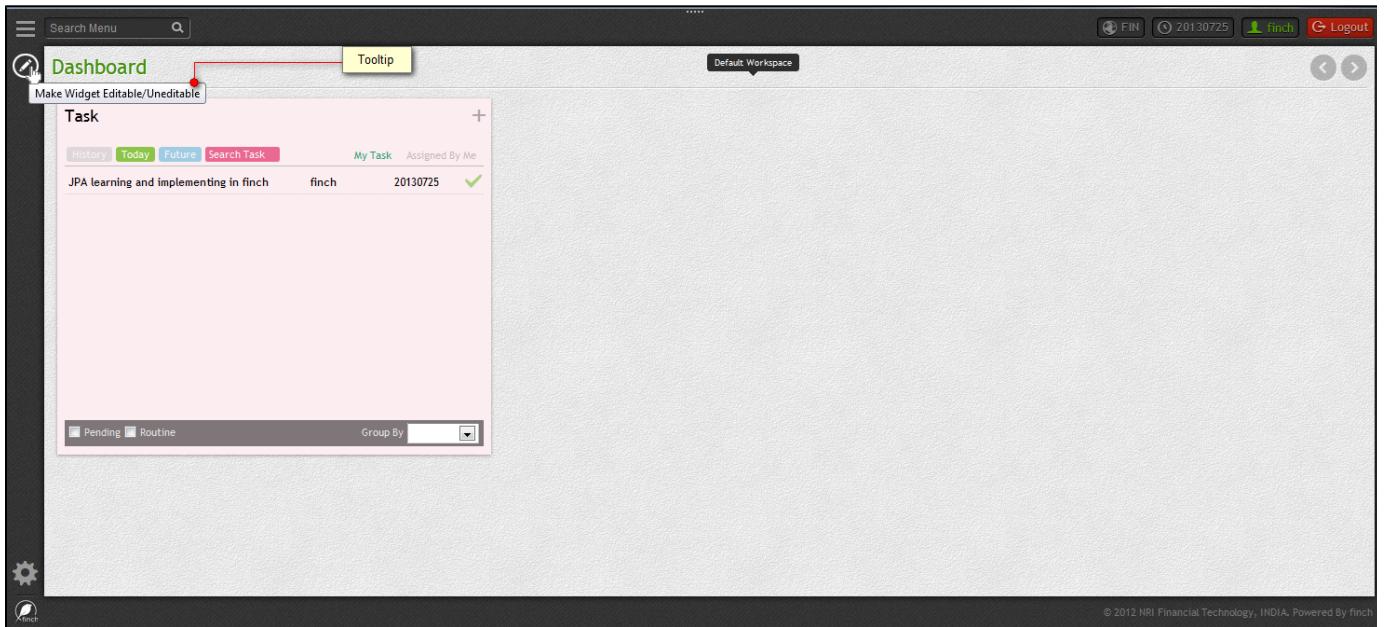
Standard Convention

The following table lists the standard convention.

Element	Colour	Button	CSS Class	Font	Size	Text Align
Mandatory Field label	RED		required	bold 11px,Arial,sans-serif	13px	
Optional Field label	BLUE			bold 11px, Arial,sans-serif	13px	
Popup for Calendar			dateinput			
Popup for Reference data			popupBtn			
Submit button	#83AB3E		btnWrapStyle submitBtn, inputBtnStyle	bold 11px, Arial,sans-serif		
Reset button	#C83A32		btnWrapStyle resetBtn, inputBtnStyle	bold 11px, Arial,sans-serif		
Save Query button	#119BCF		btnWrapStyle ,inputBtnStyle	bold 11px, Arial,sans-serif		
Save button	#83AB3E		btnWrapStyle, saveBtn, savePref,inputBtnStyle	bold 11px, Arial,sans-serif		
Cancel button	#C83A32		btnWrapStyle, cancelBtn, cancelPref,inputBtnStyle	bold 11px, Arial,sans-serif		
Personalize button	#119BCF		btnWrapStyle, persBtn,inputBtnStyle	bold 11px, Arial,sans-serif		
Wizard Next button	#119BCF		btnWrapStyle, wizNext,inputBtnStyle	bold 11px, Arial,sans-serif		
Wizard Previous button	#119BCF		btnWrapStyle, wizPrevious,inputBtnStyle	bold 11px, Arial,sans-serif		
Wizard Back button	#119BCF		btnWrapStyle, wizBack,inputBtnStyle	bold 11px, Arial,sans-serif		
Wizard Confirm button	#83AB3E		btnWrapStyle, wizConfirm,submitBtn,inputBtnStyle	bold 11px, Arial,sans-serif		
Wizard Ok button	#83AB3E		btnWrapStyle, wizOk,submitBtn,inputBtnStyle	bold 11px, Arial,sans-serif		
Textbox	White		textBox	Arial,"Liberation Sans",FreeSans,sans-serif	13px	left
Dropdown	White		dropdowninput	Arial,"Liberation Sans",FreeSans,sans-serif	13px	left
Screen Title	#419300		title, whiteFont, left	"Trebuchet MS","Arial",sans-serif	18px	

Tooltip

Each item in finch has a tooltip which shows the item name on mouse hover. To add tooltip, just add the title attribute.



Screen Title

Each screen's title should be formatted like: <SCREEN ID>: Screen Name [SCREEN DATE]

For example the first screen of Trade Entry page will be like: **TRDEN:Trade Entry [20130629]**

Entry Screen

An entry operation can have one or more screens.

- For one screen entry operation :
 - (i) First screen in edit mode with SUBMIT and RESET button.
 - (ii) On SUBMIT, the User Confirmation Screen follows.
 - (iii) On CONFIRM, the System Confirmation Screen follows.
- For multiple screen entry operation :
 - (i) First screen in edit mode with RESET and NEXT button containing all the mandatory fields.
 - (ii) Next screen in edit mode with RESET, PREVIOUS and SUBMIT button with optional fields and no mandatory fields.
 - (iii) On SUBMIT, the User Confirmation Screen follows.
 - (iv) On CONFIRM, the System Confirmation Screen follows.

The standards for SUBMIT and RESET button is mentioned as the above table.

User Confirmation Screen

User Confirmation screen will have CONFIRM and BACK buttons.

System Confirmation Screen

System Confirmation Screen will have OK Button.

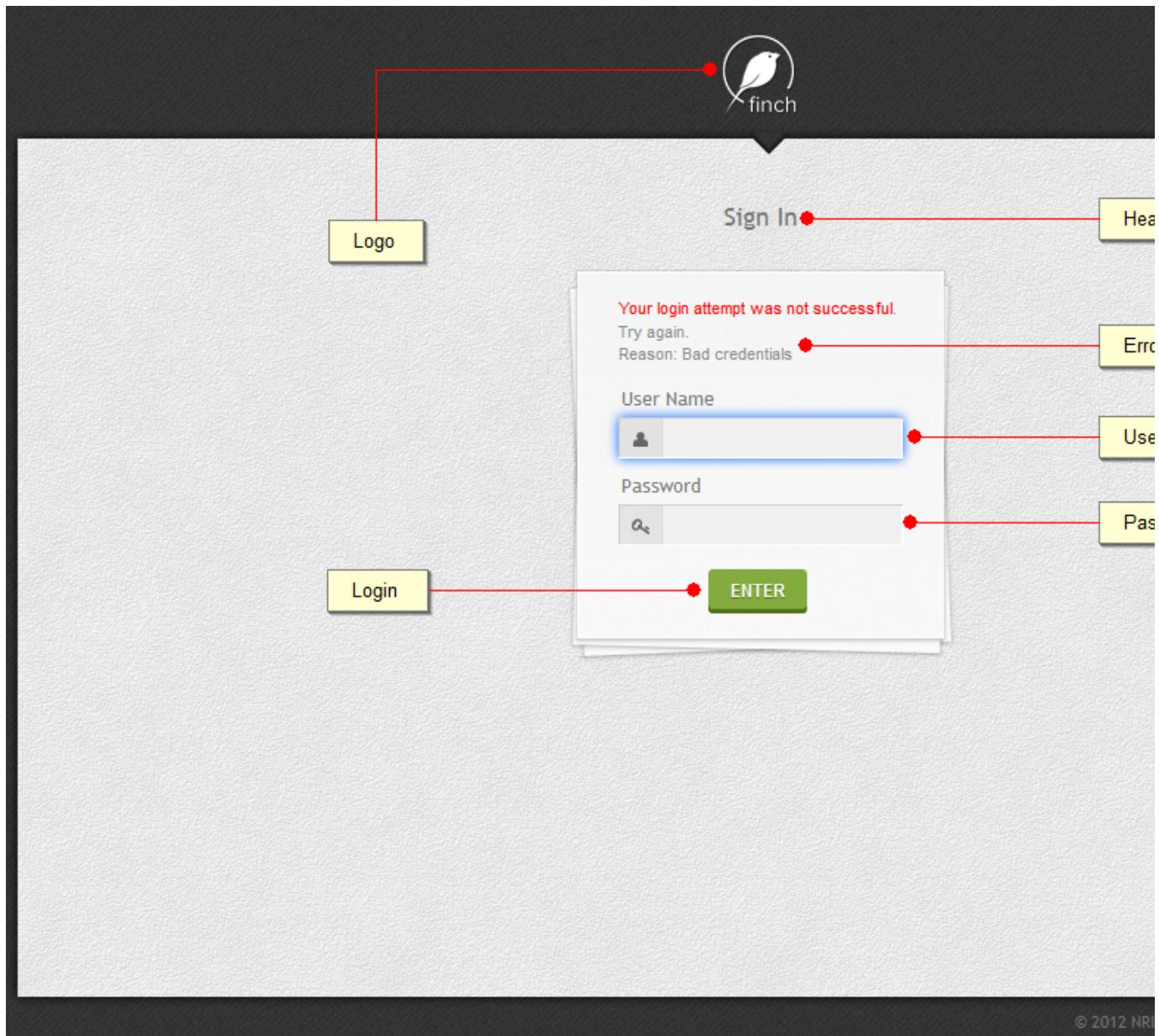
Query Screen

Query Screen can have Expand/Collapse section. Frequently used fields will be shown by default, rest all fields by default will be collapsed.

3.2.1.1 Login Screen

LOGIN SCREEN

finch provides the following login screen out of the box. for without enterprise Various controls and elements of this page have been displayed below:



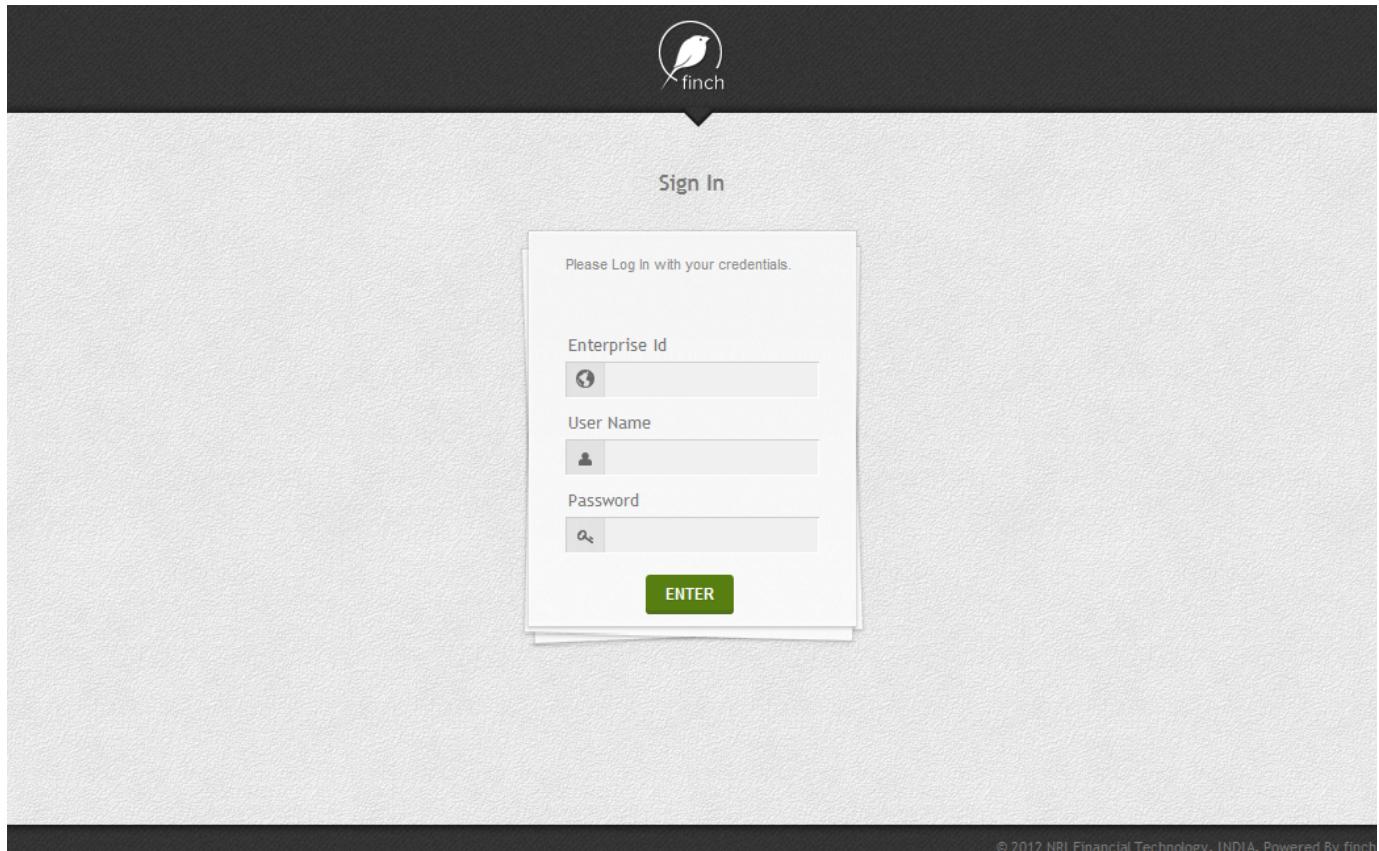
Different elements and controls marked in the above image has been briefly described below:

- Logo: finch logo or application provided logo for the application

- Heading: login heading; can be i18n compatible
- Error message: Error message if login failed
- Username input: The user name input field
- Password input: The password input field
- Login: Login button
- Also all fields are mandatory

LOGIN SCREEN With Enterprise

finch provides the following log~in screen out of the box when login with Enterprise. Various controls and elements of this page have been displayed below:



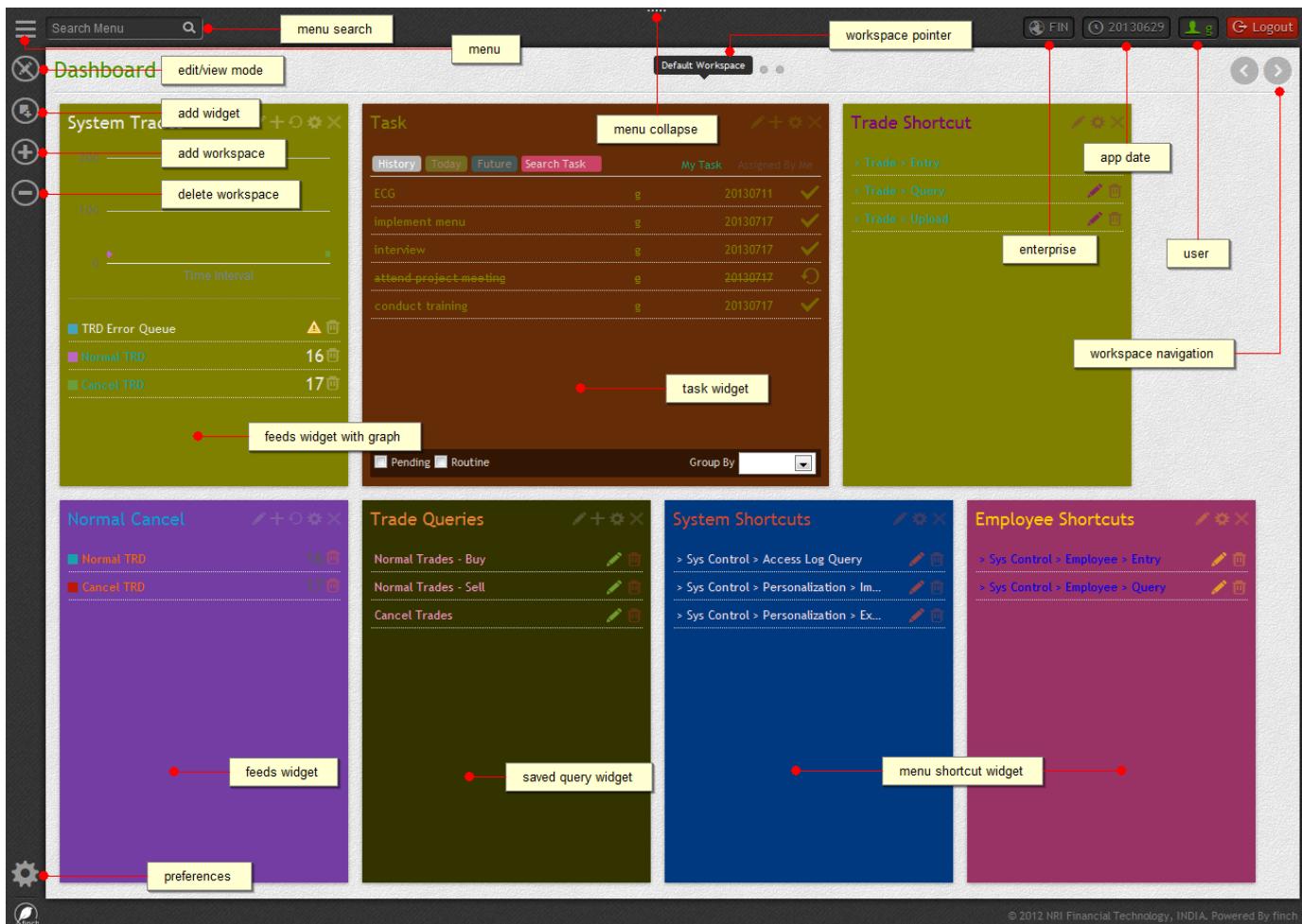
Different elements and controls marked in the above image has been briefly described below:

- Logo: finch logo or application provided logo for the application
- Heading: login heading; can be i18n compatible
- Error message: Error message if login failed
- Enterprise Id: The Enterprise ID input field
- Username input: The user name input field
- Password input: The password input field
- Login: Login button
- Also all fields are mandatory

3.2.1.2 Dashboard

DASHBOARD

After successful login, user landed to a default page, which is displayed below. Please note that this page contains a dashboard defined by a particular user. Dashboard is in the main central part of the page. There are many other UI components other that dashboard; menu, menu search etc.



Dashboard consists of several hierarchical components like workspace, widgets etc i.e. A dashboard is a collection of workspaces in it, subsequently a workspace contains several widgets in it and widget contains various contents. Various UI components are described below:

- menu: User can mouse hover on the menu and various menu items will be open.
- menu search: Autocomplete menu search input. User can navigate to a particular menu from here also.
- menu collapse: Toggle to hide and show the menu options for the application
- enterprise: Enterprise ID for the current user
- app date: app date for the current user
- user: logged in user ID
- edit/view mode: Toggle to edit/view dashboard. In editable mode various widgets, menu shortcuts and feeds are available to the user
- add workspace: For adding new workspace. User will be landed on the newly created workspace
- add widget: For adding new widget in the current workspace
- delete workspace: For deleting the current workspace. Only blank workspace can be deleted
- workspace pointer: Pointer to the current user workspace
- workspace navigation: Navigation button for multiple workspace
- feeds widget with graph: Business feed counter with graph option
- task widget: Task widget that can contain user TODO lists
- saved query: Saved query widget
- menu shortcut widget: Menu shortcut widget
- preferences: User preference menu

3.2.1.3 Wizard

General Layout

Tab marker

Screen Header

1 Trade General 2

Trade Type EQ Trade Date Value Date

Buy/Sell Orientation Account No Inventory Account No

Security Quantity Trade Currency

Execution Market Settlement Currency Price

Account Balance Type

Input fields

Buttons

RESET NEXT

Edit Mode

Page-1

screen details

mandatory field

date field

wizard step

wizard page header

popup

next step

reset

RESET NEXT

FIN 20130629 finch Logout

TRDEN:Trade Entry [20130629]

1 Trade General 2

Trade Type EQ Trade Date Value Date

Buy/Sell Orientation Account No Inventory Account No

Security Quantity Trade Currency

Execution Market Settlement Currency Price

Account Balance Type

Page- 2

Search Menu FIN 20130629 Logout

TRDEN:Trade Entry [20130629]

1 Trade General **2 Trade Details**

Trade Type	EQ	Trade Date	20130723	Value Date	20130725
Buy/Sell Orientation	AB	Account No	121212007	Inventory Account No	121212006
Security Code	CISCO	Quantity	100	Price	100.50
Execution Market	BR001	Trade Currency	INR	Settlement Currency	INR
Account Balance Type	1				

Tax Fee

Tax Fee ID(*)	<input type="text"/>			
Rate	<input type="text"/>			
Rate Type	<input type="text"/>			
Amount	<input type="text"/>			
ADD				
Action	ID	Rate	Rate Type	Amount

RR

RR Role	RR Number	Commission Amount
Executing RR	<input type="text"/>	<input type="text"/>
RR	<input type="text"/>	<input type="text"/>
Split RR1	<input type="text"/>	<input type="text"/>
Split RR2	<input type="text"/>	<input type="text"/>
Trader	<input type="text"/>	<input type="text"/>

Buttons: previous, submit, RESET, PREVIOUS, SUBMIT

© 2012 NRI Financial Technology, INDIA. Powered By finch.

User Confirmation Mode

Search Menu FIN 20130629 Logout

TENUC:Trade Entry User Confirmation [20130629]

1 Trade General **2 Trade Details**

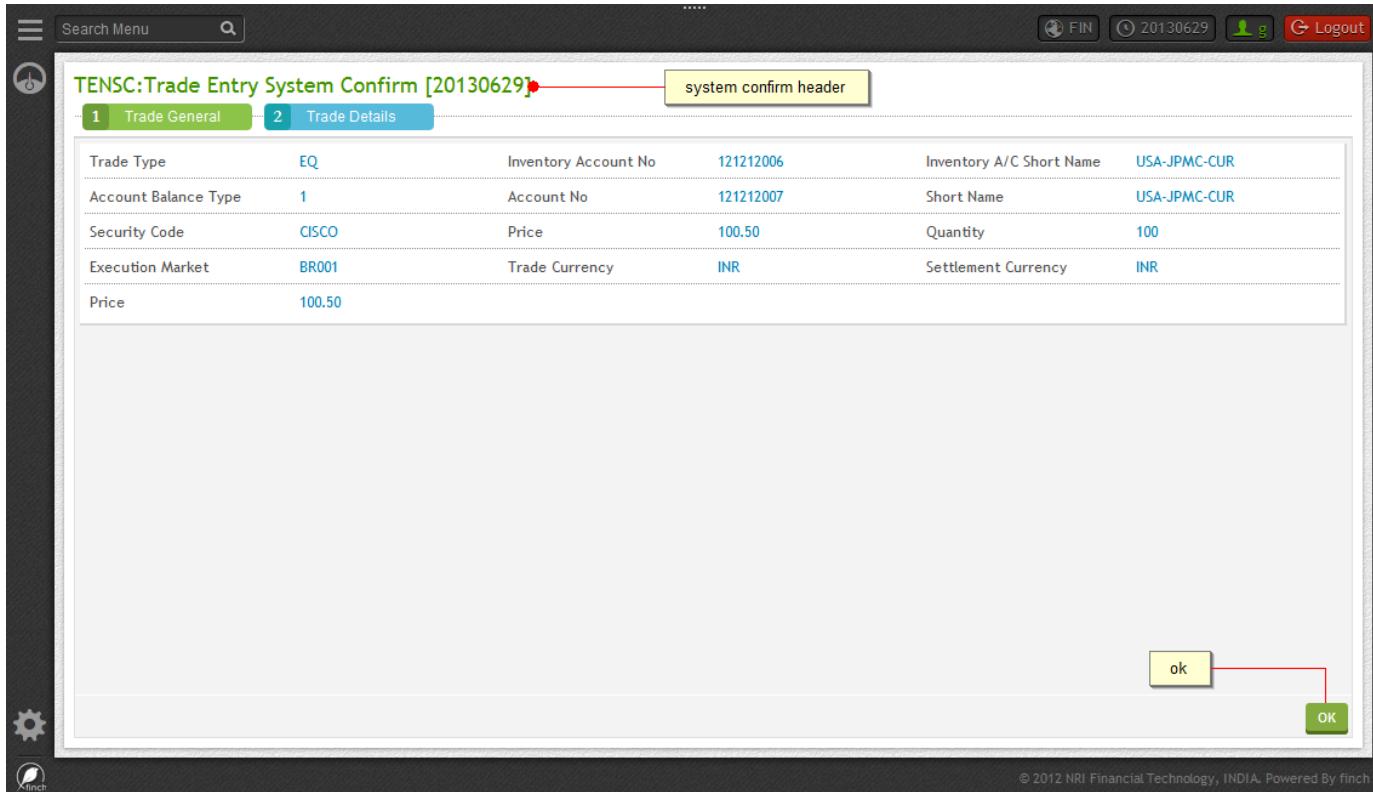
Trade Type	EQ	Inventory Account No	121212006	Inventory A/C Short Name	USA-JPMC-CUR
Account Balance Type	1	Account No	121212007	Short Name	USA-JPMC-CUR
Security Code	CISCO	Price	100.50	Quantity	100
Execution Market	BR001	Trade Currency	INR	Settlement Currency	INR
Price	100.50				

Buttons: back, confirm, BACK, CONFIRM

Labels: user confirmation header, read only info

© 2012 NRI Financial Technology, INDIA. Powered By finch.

System Confirmation Mode



3.2.1.4 Query

QUERY Tab

The Query Tab screen and the components have been described below.

General Layout:

The screenshot shows a query interface with several sections highlighted:

- Criteria items**: A red box highlights the top section containing fields like Trade Type, Inventory Account No, Settlement Currency, and Instrument Type.
- Query Header**: A red dot points to the header area which includes the screen name "TRDQR:Trade Query [20130629]", a search menu, and navigation tabs for "Query" and "Result".
- More criteria items**: A red box highlights the bottom section containing fields like Buy/Sell Flag, Last Entry Date From - To, Account Balance Type, and Data Source.
- Sort Items**: A red box highlights the "Sort Criteria" section at the bottom, which contains three dropdown menus labeled 1, 2, and 3, each with a "DESC" checkbox.
- Buttons**: A red box highlights the bottom right corner where there are "PERSONALIZE", "SAVE QUERY", "RESET", and "SUBMIT" buttons.

- **Query Header** : It contains the following items-
 - a) Screen Definition : The Screen name and Id.
 - b) Navigation tab : The tab for navigating between query and results screen.
- **Criteria Items** : It contains the criteria fields that are frequently used in building the query.
- **More Criteria Items** : It contains the criteria fields that are not frequently used in building the query.
- **Sort Items** : It contains the items that can be used as sort criteria.
- **Buttons** : It contains all the buttons required during the query criteria.

Detail Layout

The screenshot shows the 'TRDQR:Trade Query [20130629]' screen. At the top, there's a 'query header' containing search fields for Trade Type, Inventory Account No, Settlement Currency, Instrument Type, Trade Reference No, Security, Trade Date From-To, Execution Market, Account No, Trade Currency, and Value Date From - To. Below these are 'Sort Criteria' fields labeled 1, 2, and 3, each with a dropdown and a 'DESC' checkbox. A 'more' button is located next to the sort criteria. On the right side of the header is a 'query tab' with buttons for 'Query', 'Result', and 'Logout'. At the bottom left are 'PERSONALIZE' and 'SAVE QUERY' buttons, and at the bottom right are 'reset', 'RESET', and 'SUBMIT' buttons. A 'popup' button is also present. The footer contains a copyright notice: '© 2012 NRI Financial Technology, INDIA. Powered By finis'.

Result Tab

The Query Tab screen and the components have been described below.

General Layout

TRDRS:Trade Query Result [17-Aug-2013]						
	Trade Date	Reference No	Trade Curren...	Trade Type	Value Date	Account Bal...
1	17-Aug-2013	T20130817001012	USD	EQ	20-Aug-2013	5
2	19-Aug-2013	T20130817001082	USD	EQ	21-Aug-2013	1
3	02-Sep-2013	T20130817001169	USD	EQ	05-Sep-2013	-1
4	20-Sep-2013	T20130817001361	INR	EQ	23-Sep-2013	1
5	28-Aug-2013	T20130817001096	INR	EQ	30-Aug-2013	1
6	11-Dec-2012	T20130817001020	INR	EQ	03-Dec-2012	1
7	26-Aug-2013	T20130817001072	INR	EQ	28-Aug-2013	-1
8	22-Aug-2013	T20130817001070	USD	EQ	26-Aug-2013	-1
9	02-Sep-2013	T20130817001142	INR	EQ	04-Sep-2013	5
10	01-Aug-2013	T20130817001125	JPY	EQ	05-Aug-2013	5
11	29-Aug-2013	T20130817001115	USD	EQ	02-Sep-2013	5
12	26-Aug-2013	T20130817001076	USD	EQ	28-Aug-2013	1
13	10-Sep-2013	T20130817001321	JPY	FI	18-Sep-2013	8
14	12-Sep-2013	T20130817001261	INR	EQ	16-Sep-2013	1

- Result Screen Header : It contains the following items-
 - a) Screen Definition : The Screen name and Id.
 - b) Navigation tab : The tab for navigating between query and results screen.
- Result Grid Header : It contains the following items:
 - a) Pagination tool : Used for navigating between records.
 - b) Report Download : Used for downloading report in various formats.
 - c) Columns : Used for filtering columns during the display of result along with column settings.

- d) Save Preference : Used for storing the user based modification.
- Column Headers : It contains the column heading of the query result.
- Result Values : It contains the query result values under each specified head along with consolidated action for further operation.

Detail Layout

TRDRS:Trade Query Result [17-Aug-2013]						
	Trade Date	Reference No	Trade Curren...	Trade Type	Value Date	Account Bal...
↗	17-Aug-2013	T20130817001012	USD	EQ	20-Aug-2013	5
↗	19-Aug-2013	T20130817001082	USD	EQ	21-Aug-2013	1
↗	02-Sep-2013	T20130817001169	USD	EQ	05-Sep-2013	-1
↗	20-Sep-2013	T20130817001361	INR	EQ	23-Sep-2013	1
↗	28-Aug-2013	T20130817001096	INR	EQ	30-Aug-2013	1
↗	11-Dec-2012	T20130817001020	INR	EQ	03-Dec-2012	1
↗	26-Aug-2013	T20130817001072	INR	EQ	28-Aug-2013	-1
↗	22-Aug-2013	T20130817001070	USD	EQ	26-Aug-2013	-1
↗	02-Sep-2013	T20130817001142	INR	EQ	04-Sep-2013	5
↗	01-Aug-2013	T20130817001125	JPY	EQ	05-Aug-2013	5
↗	29-Aug-2013	T20130817001115	USD	EQ	02-Sep-2013	5
↗	26-Aug-2013	T20130817001076	USD	EQ	28-Aug-2013	1
↗	10-Sep-2013	T20130817001321	JPY	FI	18-Sep-2013	8
↗	12-Sep-2013	T20130817001261	INR	EQ	16-Sep-2013	1

Consolidated Actions:

TRDRS:Trade Query Result [17-Aug-2013]						
	Trade Date	Reference No	Trade Curren...	Trade Type	Value Date	Account Bal...
↗	17-Aug-2013	T20130817001012	USD	EQ	20-Aug-2013	5
↗	19-Aug-2013	T20130817001082	USD	EQ	21-Aug-2013	1
↗	02-Sep-2013	T20130817001169	USD	EQ	05-Sep-2013	-1
↗	20-Sep-2013	T20130817001361	INR	EQ	23-Sep-2013	1
↗	28-Aug-2013	T20130817001096	INR	EQ	30-Aug-2013	1
↗	11-Dec-2012	T20130817001020	INR	EQ	03-Dec-2012	1
↗	26-Aug-2013	T20130817001072	INR	EQ	28-Aug-2013	-1
↗	22-Aug-2013	T20130817001070	USD	EQ	26-Aug-2013	-1
↗	02-Sep-2013	T20130817001142	INR	EQ	04-Sep-2013	5
↗	01-Aug-2013	T20130817001125	JPY	EQ	05-Aug-2013	5
↗	29-Aug-2013	T20130817001115	USD	EQ	02-Sep-2013	5
↗	26-Aug-2013	T20130817001076	USD	EQ	28-Aug-2013	1
↗	10-Sep-2013	T20130817001321	JPY	FI	18-Sep-2013	8
↗	12-Sep-2013	T20130817001261	INR	EQ	16-Sep-2013	1

3.3 Design Phase

3.3.1 Software Library

Introduction

finch application has certain software requirements or requires other software resources to be present on a computer. These prerequisites are known as software requirements and required to be full filled for getting optimal performance. finch defines different set of software requirements for Client and Server side respectively. The subsequent sections details the software requirement for finch application.

Client Machine Software Requirement

finch web based user interface runs on standard desktop or laptop machine with standard web browser. The client machine requires certain software to install and run finch, which have been discussed in the subsequent sections.

Web Browser

finch supports the following Web Browsers:

- Microsoft Internet Explorer - version 8+
- Mozilla Firefox - version 16+
- Google Chrome - version 23+

Screen Resolution

finch supports the minimum resolution of 1024 x 768.

Font Size

100% defines as the standard.

Other Software

In the Client machine the following software need to be installed:

- PDF Reader (like Adobe Acrobat Reader) – to view reports in PDF format.
- Microsoft Excel – to view reports in Excel (XLS and XLSX) format.

Server Machine Software Requirement

The Server machine has certain software requirements to install and finch, which have been discussed in the subsequent sections.

Operating System

finch has been developed on Java platform and is capable of running on most of the popular operating system. finch has been tested on following suite of operating systems:

- Windows - Windows 7 professional
- Linux - RHEL 6.x

Database

finch uses open standard API (Java Persistence API) for database persistence and is capable of running on any standard RDBMS. finch has been tested on following databases:

- Oracle - version 10g, 11g
- MySQL - version 5.x

Application Server

finch is capable of running on any application server that has standard Java Servlet container. finch has been tested on following application Application Servers:

- Apache Tomcat - version 6.x, version 7.x
- Redhat JBoss AS - version 7.x

Library Used

Server Side

No	Name	Version	Description
1	Java SE	1.7.x	Java Standard Library
2	Spring Framework	3.2.2	Dependency injection, transaction handling and Web MVC(model-view-controller) framework
3	Spring Web Flow	2.3.2	Spring Web Flow. A library over Spring MVC to facilitate navigation flow.
4	Spring Security	3.1.3	Spring Security for authentication and authorization.
5	Spring Batch	2.2.0	Spring batch framework
6	JPA	2.0	JPA(Java Persistence API). Java standard interface for data persistence.
7	Hibernate	4.1.12	JPA(Java Persistence API) implementation. Used for data persistence.
8	Google Guava	13.0.1	Open source Utility library
9	Altassian Fugue	1.1	Open source Utility library
10	Simple Logging Facade for Java (SLF4J)	1.7.2	Facade abstraction for logging framework
11	Logback	1.0.9	Application logging API.
12	Apache Camel	2.12.1	Free open-source implementation of enterprise service bus(ESB)
13	Apache Zookeeper	3.3.5	open-source server which enables highly reliable distributed coordination. Used for controlling console processes.
14	DBUnit	2.4.9	For data driven unit testing.
15	Jackson	2.1.4	Java library for processing JSON data format
16	Apache Tiles	2.2.1	Open source java template library. Used for screen layout.
17	JSTL	1.2	Standard JSP tag library
18	Apache common API		List of common utility API from Apache: <ul style="list-style-type: none"> • beanutils • codec 1.7 • collections 3.2.1 • digester 2.1 • discovery 0.2 • fileupload 1.2.2 • io 1.3.2 • lang3 3.1
19	Jasper Report	5.0.1	Report generation
20	Apache POI	3.7	API to read and create Microsoft Excel file
21	iText	2.1.7	API to read and create PDF file
22	Oracle JDBC Thin driver	11.2.0.1.0	Oracle JDBC Thin driver
23	JMS	1.1	Java standard API to interact with messaging provider.
24	XLS Reader	1.0.1	API to read XLS, XLSX file format.

25	XML		for XML followings are used: <ul style="list-style-type: none"> • xml-apis 1.4.01 • xmlsec 1.5.3 • jaxb-imp I-2.2.5 • xstream-1.3.1 • jdom-b10
26	joda-time	2.1	Java library for Date and Time handling
27	Security Assertion Markup Language (SAML)	2.5.1-1	Open source SAML API for single-sign-on(SSO)
28	jOOQ	3.0.1	DSL based abstraction over JDBC API.
29	Open CSV	2.0	Open source API to read CSV file.
30	DropWizard	0.6.2	Java framework for RESTful web services

Client Side

No	Name	Version	Description
1	jQuery	1.7.1	jQuery 1.7.1 is used
2	Handlebars	1.0 rc 2	Handlebars 1.0.rc.2
3	High Charts	2.1.9	High Charts 2.1.9
4	Hover Intent		Hover Intent
5	Json 2	2.0	Json 2
6	Objx	2.3.6	Objx 2.3.6
7	Slick Grid	2.1	Slick Grid 2.1
8	Super Fish	1.4.8	Super Fish 1.4.8
9	Date	1.0 alpha 1	Date 1.0 alpha 1
10	jQuery Alerts	1.1	jQuery Alerts 1.1
11	jQuery Color Box	3.19	jQuery Color Box 3.19
12	jQuery Dyna Tree	1.2.4	jQuery Dyna Tree 1.2.4
13	jQuery Elastislide		jQuery Elastislide
14	jQuery Growl	1.0.2	jQuery Growl 1.0.2
15	jQuery Dock	2.0.0	jQuery Dock 2.0.0
16	jQuery UI	1.8.16	jQuery UI 1.8.16
17	jQuery Tmpl	1.0.0 pre	jQuery Tmpl 1.0.0 pre

3.3.2 Data Model Design

This section contains all the data models used during construction of developer's guide and user guide.

All the tables described in this section contain few audit fields which are mandatory and have been removed from the table structure. The audit fields are described below:

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
APP_REGI_DATE	DATE	NO	The application registration date
APP_UPD_DATE	DATE	NO	The updated application date
CREATED_BY	VARCHAR2(20 BYTE)	NO	The user who has inserted the record
CREATION_DATE	DATE	NO	The date on which the record was inserted
UPDATED_BY	VARCHAR2(20 BYTE)	NO	The user who has updated the record

UPDATE_DATE	DATE	NO	The date on which the record has been updated
CC_CHECK	NUMBER(10,0)	NO	Concurrency Check Field

3.3.2.1 Guidelines

3.3.2.2 finch Data Model

3.3.2.3 Infrastructure (INF) Data Model

INF_ACCESS_LOG

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ACCESS_LOG_PK	NUMBER	No	The primary key of the table
URL	VARCHAR2(255 BYTE)	No	URL of the resource which is accessed by the user
SCREEN_ID	VARCHAR2(20 BYTE)	Yes	Screen ID of the resource which is accessed by the user
USER_ID	VARCHAR2(20 BYTE)	No	User ID of the user who accessed the resource
SESSION_ID	VARCHAR2(128 BYTE)	No	HTTP Session ID of the user
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	The foreign key references of INF_ENTERPRISE

INF_ACTION

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ACTION_ID	VARCHAR2(8 BYTE)	No	The primary key
ACTION_URL	VARCHAR2(255 BYTE)	No	The URL to be executed
ACTION_NAME	VARCHAR2(50 BYTE)	Yes	The name of the action
ACTION_KEY	VARCHAR2(8 BYTE)	Yes	The action key associated
ACTION_URL_PATTERN	VARCHAR2(255 BYTE)	Yes	The URL pattern

INF_ACTION_CONSOLIDATION

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ACTION_ID	VARCHAR2(8 BYTE)	No	Composite Primary Key and The Foreign key references of INF_ACTION
SUB_ACTION_ID	VARCHAR2(8 BYTE)	No	Composite Primary Key and Foreign key references of INF_ACTION
DISPLAY_SEQ	NUMBER(2,0)	Yes	Display Sequence of the actions

INF_ACTION_ENTERPRISE_PCPT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ACTION_ENTERPRISE_PCPT_PK	NUMBER(10,0)	No	The primary key
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	The foreign key of the INF_ENTERPRISE
ACTION_ID	VARCHAR2(8 BYTE)	No	The foreign key of the INF_ACTION

STATUS	VARCHAR2(6 BYTE)	Yes	The status field
--------	------------------	-----	------------------

INF_ACTION_ROLE_PARTICIPANT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ACTION_ROLE_PCPT_PK	NUMBER(10,0)	No	The primary key of the application
APPL_ROLE_PK	NUMBER(10,0)	No	The foreign key of the INF_APPLICATION_ROLE
ACTION_ID	VARCHAR2(8 BYTE)	No	The foreign key of the INF_ACTION
STATUS	VARCHAR2(6 BYTE)	Yes	Status of the role

INF_APPLICATION_DATE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	The foreign key reference to INF_ENTERPRISE
COMPONENT_ID	VARCHAR2(6 BYTE)	No	The foreign key reference to INF_COMPONENT
APP_MODE	VARCHAR2(2 BYTE)	No	Application Mode
DESCRIPTION	VARCHAR2(50 BYTE)	Yes	The foreign key of the INF_ACTION
APPLICATION_DATE	DATE	No	Status of the role
BRANCH_PK	NUMBER(10,0)	Yes	The foreign key reference to INF_BRANCH
APPLICATION_DATE_PK	NUMBER(4,0)	No	The primary key

INF_APPLICATION_ROLE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
APPL_ROLE_PK	NUMBER(5,0)	No	The primary key
APPLICATION_ROLE_NAME	VARCHAR2(30 BYTE)	No	The name of the role
BRANCH_PK	NUMBER(4,0)	No	The foreign key references to INF_BRANCH
REMARKS	VARCHAR2(40 BYTE)	Yes	The remarks of the application role
STATUS	VARCHAR2(6 BYTE)	Yes	The status of the role

INF_BATCH_MASTER

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
BATCH_NAME	VARCHAR2(50 BYTE)	No	The primary key
STATUS	VARCHAR2(6 BYTE)	No	Status of batch
COMPONENT_NAME	VARCHAR2(10 BYTE)	Yes	Name of the component

INF_BATCH_ROLE_PCPT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
BATCH_ROLE_PCPT_PK	VARCHAR2(8 BYTE)	No	The primary key of the table
BATCH_NAME	VARCHAR2(8 BYTE)	No	Foreign key references of INF_BATCH_MASTER
APPL_ROLE_PK	NUMBER(5,0)	No	Foreign key references of INF_APPLICATION_ROLE

INF_BATCH_RUNNER_CONFIG

General Table Structure

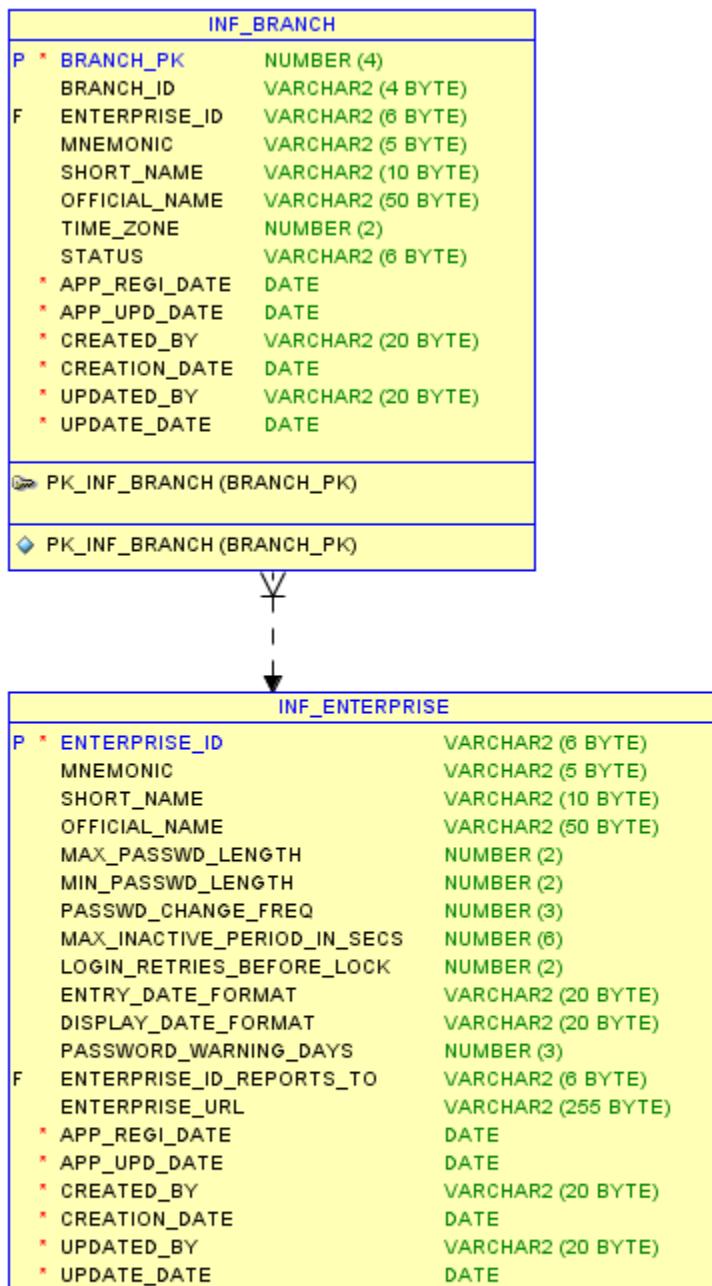
COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
CONFIG_PK	NUMBER(10,0)	No	The primary key
SERVER_HOST	VARCHAR2(512 BYTE)	No	Server Host Value
SERVER_PORT	VARCHAR2(512 BYTE)	No	Server Port Value
SERVER_STATUS	VARCHAR2(512 BYTE)	No	Server Status whether Started or Stopped etc.

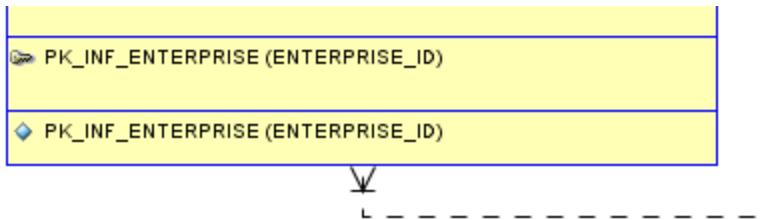
INF_BRANCH

Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
BRANCH_PK	NUMBER(4,0)	No	The primary key
BRANCH_ID	VARCHAR2(4 BYTE)	Yes	The branch id
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	The foreign key references of INF_ENTERPRISE
MNEMONIC	VARCHAR2(5 BYTE)	Yes	The mnemonic name of the enterprise
SHORT_NAME	VARCHAR2(10 BYTE)	Yes	The short name of the branch
OFFICIAL_NAME	VARCHAR2(50 BYTE)	Yes	The official name of the branch
TIME_ZONE	NUMBER(2,0)	Yes	The timezone of the branch
STATUS	VARCHAR2(6 BYTE)	Yes	The status of the branch
CALENDAR_PK	NUMBER(3,0)	No	The foreign key references of INF_CALENDAR

Diagram With Relations





INF_BRANCH_NAME_XREF

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
BRANCH_NAME_XREF_PK	NUMBER(6,0)	No	The primary key
BRANCH_PK	NUMBER(4,0)	Yes	Foreign key references of INF_BRANCH
BRANCH_ID	VARCHAR2(4 BYTE)	No	The Branch ID
LANGUAGE_CODE	VARCHAR2(15 BYTE)	Yes	Foreign key references of INF_LANGUAGE
SHORT_NAME	VARCHAR2(20 BYTE)	Yes	Branch Short Name
OFFICIAL_NAME	VARCHAR2(100 BYTE)	Yes	Official Name
INF_BRANCH_NAME_XREF_PK	NUMBER(19,0)	No	

INF_CALENDER

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
CALENDAR_PK	NUMBER(4,0)	No	The primary key
CALENDAR_ID	VARCHAR2(40 BYTE)	Yes	Calender ID
CALENDAR_NAME	VARCHAR2(100 BYTE)	Yes	Name of the Calender
WEEKEND_PATTERN	VARCHAR2(7 BYTE)	No	Pattern of weekend
STATUS	VARCHAR2(6 BYTE)	No	Status of the calender

INF_CLOSING_STATUS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
CLOSING_STATUS_PK	NUMBER(10,0)	No	The primary key
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	The Foreign key references of INF_ENTERPRISE
COMPONENT_ID	VARCHAR2(6 BYTE)	No	The Foreign key references of INF_COMPONENT
APP_MODE	VARCHAR2(2 BYTE)	No	Application Mode
STATUS	CHAR(1 BYTE)	No	Any Status
APPLICATION_DATE	DATE	No	Date of the Application
SCREEN_ID	VARCHAR2(5 BYTE)	Yes	Id of the Screen

INF_COMPONENT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
COMPONENT_ID	VARCHAR2(6 BYTE)	No	The primary key of the table
COMPONENT_NAME	VARCHAR2(100 BYTE)	Yes	Component Name
COMPONENT_DESCRIPTION	VARCHAR2(100 BYTE)	Yes	Component Description
PARENT_COMPONENT_ID	VARCHAR2(6 BYTE)	Yes	Foreign key references of INF_COMPONENT
COMPONENT_PACKAGE	VARCHAR2(50 BYTE)	Yes	Base java package for this component

INF_CONSTRAINT_LIST

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
CONSTRAINT_PK	NUMBER(6,0)	No	The primary key of the table
CONSTRAINT_NAME	VARCHAR2(100 BYTE)	No	Name of the Constraint

INF_CONSTRAINT_VALUES

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
CONSTRAINT_PK	NUMBER(4,0)	No	Composite Primary Key and The Foreign key references of INF_CONSTRAINT_LIST
CONSTRAINT_VALUE	VARCHAR2(100 BYTE)	No	Composite Primary Key
LANGUAGE_CODE	VARCHAR2(15 BYTE)	No	Foreign key references of INF_LANGUAGE
DEFAULT_MSG	CHAR(1 BYTE)	Yes	Default Message
DEFAULT_UI	CHAR(1 BYTE)	Yes	Whether Default UI or Not
UI_DISPLAY_VALUE	VARCHAR2(100 BYTE)	Yes	Display value of UI
DISPLAY_SEQ	NUMBER(2,0)	Yes	Display Sequence of the actions
ENTERPRISE_ID	NUMBER(10,0)	No	Foreign key references of INF_ENTERPRISE

INF_EMP_APPLN_ROLE_PARTICIPANT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
APPL_ROLE_PK	NUMBER(5,0)	No	Composite Primary Key and The Foreign key references of INF_APPLLICATION_ROLE
EMPLOYEE_PK	NUMBER(10,0)	No	Composite Primary Key and The Foreign key references of INF_EMPLOYEE

INF_EMPLOYEE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
EMPLOYEE_PK	NUMBER(10,0)	No	The primary key
LAST_NAME	VARCHAR2(35 BYTE)	Yes	The last name of the employee
FIRST_NAME	VARCHAR2(35 BYTE)	Yes	The first name of the employee
MIDDLE_INITIAL	VARCHAR2(4 BYTE)	Yes	The middle initial of the employee
SUFFIX	VARCHAR2(5 BYTE)	Yes	The suffix of the employee
TITLE	VARCHAR2(5 BYTE)	Yes	The title of the employee

USER_ID	VARCHAR2(20 BYTE)	Yes	The user id of the employee
APPL_PASSWD	VARCHAR2(50 BYTE)	Yes	The application password
START_DATE	DATE	Yes	The start date of the login
END_DATE	DATE	Yes	The end date of the login
LAST_PASSWORD_CHANGE_DATE	DATE	Yes	The last time the password was changed
WRONG_PASSWORD_COUNT	NUMBER(10,0)	Yes	The number of invalid attempts
STATUS	VARCHAR2(6 BYTE)	Yes	The status of the employee
LOCKED	CHAR(1 BYTE)	Yes	Whether the employee is locked
MAIL	VARCHAR2(40 BYTE)	Yes	The mail address of the employee
EMPLOYEE_ID	VARCHAR2(15 BYTE)	Yes	The employee unique id
EMPLOYEE_OPEN_DATE	DATE	Yes	The employee open date
EMPLOYEE_CLOSED_BY	VARCHAR2(20 BYTE)	Yes	The employee was closed by
EMPLOYEE_CLOSE_DATE	DATE	Yes	The closing date of the employee
EMPLOYEE_OPENED_BY	VARCHAR2(20 BYTE)	Yes	The employee was opened by
DEFAULT_BRANCH_PK	NUMBER(4,0)	Yes	The default branch to which the employee belongs

INF_EMPLOYEE_GROUP_PCPT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
EMPLOYEE_GROUP_PK	NUMBER(10,0)	No	The primary key of the table
GROUP_PK	NUMBER(10,0)	No	Foreign key references of INF_GROUP
EMPLOYEE_PK	NUMBER(10,0)	Yes	Foreign key references of INF_EMPLOYEE

INF_EMPLOYEE_NAME_XREF

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
EMPLOYEE_NAME_XREF_PK	NUMBER(6,0)	No	The primary key of the table
EMPLOYEE_PK	NUMBER(10,0)	Yes	Foreign key references of INF_EMPLOYEE
LANGUAGE_CODE	VARCHAR2(15 BYTE)	Yes	Foreign key references of INF_LANGUAGE
LAST_NAME	VARCHAR2(35 BYTE)	Yes	Last Name of the Employee
FIRST_NAME	VARCHAR2(35 BYTE)	Yes	First Name of the Employee

INF_ENDPOINT

General Table Structure

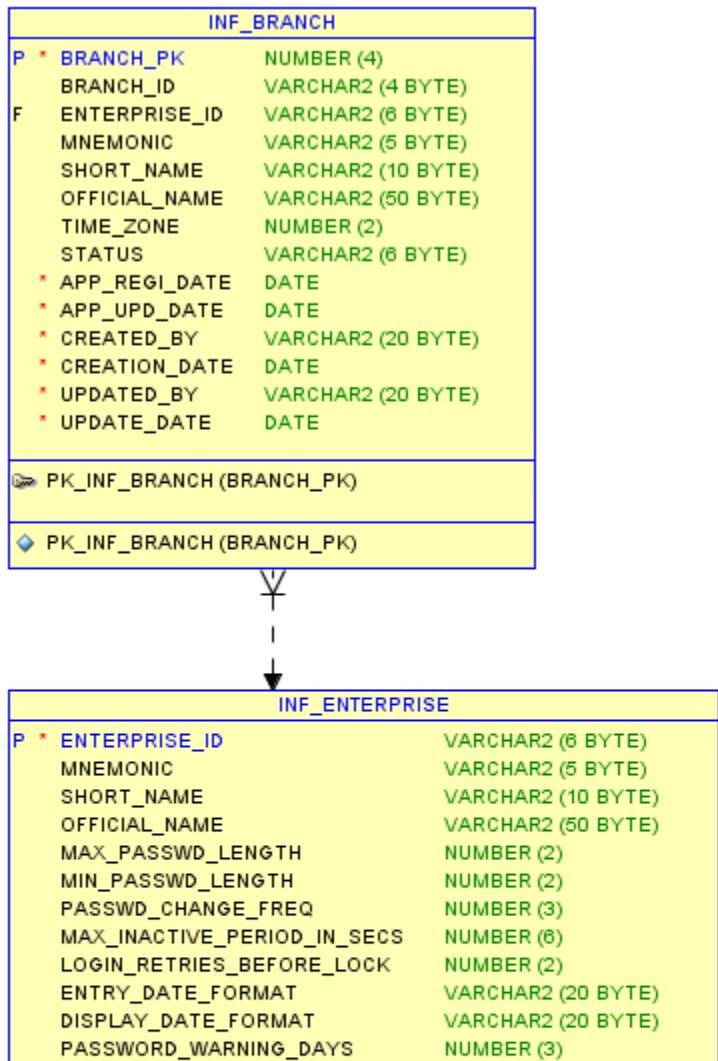
COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ENDPOINT_PK	NUMBER(10,0)	No	The primary key of the table
ENDPOINT_URI	VARCHAR2(512 BYTE)	No	Apache Camel Complained URI for Endpoint
ENTERPRISE_ID	NUMBER(10,0)	No	Foreign key references of INF_ENTERPRISE

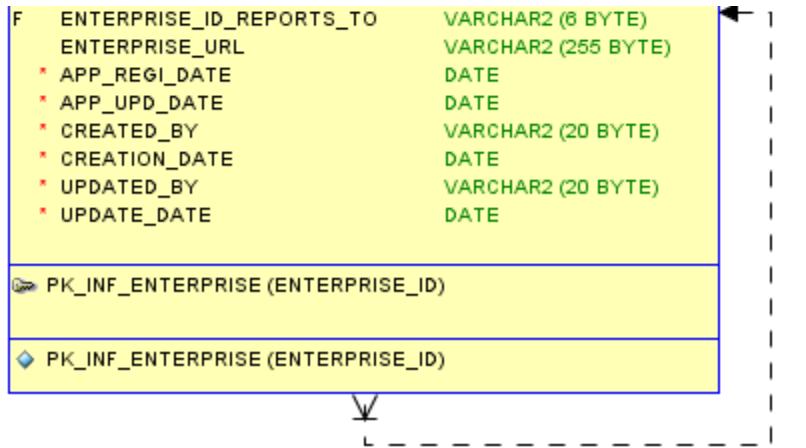
INF_ENTERPRISE

Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	The primary key
MNEMONIC	VARCHAR2(5 BYTE)	Yes	Mnemonic for the enterprise
SHORT_NAME	VARCHAR2(10 BYTE)	Yes	The short name of the enterprise
OFFICIAL_NAME	VARCHAR2(50 BYTE)	Yes	The official name of the enterprise
MAX_PASSWD_LENGTH	NUMBER(2,0)	Yes	The maximum password length
MIN_PASSWD_LENGTH	NUMBER(2,0)	Yes	The minimum password length
PASSWD_CHANGE_FREQ	NUMBER(3,0)	Yes	the number of days after INF_EMPLOYEE.LAST_PASSWORD_CHANGE_DATE when the password will expire.
MAX_INACTIVE_PERIOD_IN_SECS	NUMBER(6,0)	Yes	Maximum inactive period in seconds
LOGIN_RETRIES_BEFORE_LOCK	NUMBER(2,0)	Yes	Number of retries before lock
ENTRY_DATE_FORMAT	VARCHAR2(20 BYTE)	Yes	The entry date format
DISPLAY_DATE_FORMAT	VARCHAR2(20 BYTE)	Yes	The display date format
PASSWORD_WARNING_DAYS	NUMBER(3,0)	Yes	The number of days to show the warning
ENTERPRISE_ID_REPORTS_TO	VARCHAR2(6 BYTE)	Yes	Parent enterprise whom current one reports to
ENTERPRISE_URL	VARCHAR2(255 BYTE)	Yes	The url of the enterprise

Diagram with Relations





INF_ENTERPRISE_NAME_XREF

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ENTERPRISE_NAME_XREF_PK	NUMBER(6,0)	No	The primary key of the table
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	Foreign key references of INF_ENTERPRISE
LANGUAGE_CODE	VARCHAR2(15 BYTE)	Yes	Foreign key references of INF_LANGUAGE
SHORT_NAME	VARCHAR2(20 BYTE)	Yes	Short Name
OFFICIAL_NAME	VARCHAR2(100 BYTE)	Yes	Official Name
ENTERPRISE_ID	NUMBER(10,0)	No	Foreign key references of INF_ENTERPRISE

INF_FILE_DIRECTORY

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
FILE_DIRECTORY_PK	NUMBER(6,0)	No	The primary key of the table
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	Foreign key references of INF_ENTERPRISE
DIRECTORY_ID	VARCHAR2(50 BYTE)	Yes	Id of the Directory
PATH	VARCHAR2(200 BYTE)	No	User ID of the user who accessed the resource

INF_GROUP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
GROUP_PK	NUMBER(10,0)	No	The primary key of the table
GROUP_ID	VARCHAR2(20 BYTE)	No	ID of the group
GROUP_DESC	VARCHAR2(100 BYTE)	Yes	Description of the group
GROUP_EMAIL	VARCHAR2(50 BYTE)	Yes	Email of the group

INF_HOLIDAY

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
HOLIDAY_PK	NUMBER(10,0)	No	The primary key of the table
CALENDAR_PK	NUMBER(4,0)	No	Foreign key references of INF_CALENDAR
HOLIDAY_DATE	DATE	No	Date of Holiday
HOLIDAY_NAME	VARCHAR2(100 BYTE)	No	Name of Holiday
HOLIDAY_TYPE	VARCHAR2(100 BYTE)	Yes	Type of Holiday
STATUS	VARCHAR2(6BYTE)	Yes	Status of holiday

INF_LANGUAGE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
LANGUAGE_CODE	VARCHAR2(15 BYTE)	No	Foreign key references of INF_LANGUAGE
DESCRIPTION	VARCHAR2(50 BYTE)	Yes	Language Description
LOCALE	VARCHAR2(20 BYTE)	No	Local name

INF_NOTIFICATION

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
NOTIFICATION_PK	NUMBER(10,0)	No	The primary key of the table
EXT_REF_NUMBER	NUMBER(10,0)	No	Unique Key
SOURCE_TYPE	VARCHAR2(15 BYTE)	Yes	Type of Source
SUMMARY	VARCHAR2(256 BYTE)	Yes	Summary of notification
DETAIL	VARCHAR2(3072 BYTE)	Yes	Details of the notification
NOTIFICATION_TYPE	VARCHAR2(30 BYTE)	Yes	Type of Notification

INF_NOTIFICATION_ATTACHMENT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
NOTIFICATION_PK	NUMBER(10,0)	No	Foreign key reference to INF_NOTIFICATION
NOTIFICATION_ATTACHMENT_PK	NUMBER(10,0)	No	The primary key of the table
ATTACHMENT_TYPE	VARCHAR2(200 BYTE)	Yes	Type of Attachment
ATTACHMENT_CONTENT	BLOB	Yes	Content of the attachment
ATTACHMENT_NAME	VARCHAR2(200 BYTE)	Yes	Name of attachment

INF_NOTIFICATION_DESTINATION

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
NOTIFICATION_DESTINATION_PK	NUMBER(10,0)	No	The primary key of the table

NOTIFICATION_PK	NUMBER(10,0)	No	Foreign key reference to INF_NOTIFICATION
CHANNEL_TYPE	VARCHAR2(20 BYTE)	Yes	Type of channel
CHANNEL_ADDRESS	VARCHAR2(200 BYTE)	Yes	Address of Channel
DELIVERY_STATUS	CHAR(1 BYTE)	Yes	Delivery Status
GROUP_PK	NUMBER(10,0)	Yes	Foreign key reference to INF_GROUP
EMPLOYEE_PK	NUMBER(10,0)	Yes	Foreign key reference to INF_EMPLOYEE

INF_NOTIFICATION_SERVER

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
NOTIFICATION_SERVER_PK	NUMBER(10,0)	No	The primary key of the table
NOTIFICATION_HOST	VARCHAR2(20 BYTE)	No	Contains the host name for the notification server
NOTIFICATION_PORT	VARCHAR2(20 BYTE)	No	Contains the port number for the notification server

INF_PASSWORD_HISTORY

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
PASSWORD_HISTORY_PK	NUMBER(10,0)	No	The primary key of the table
EMPLOYEE_PK	NUMBER(10,0)	Yes	Foreign key references of INF_EMPLOYEE
APPL_PASSWORD	VARCHAR2(50 BYTE)	Yes	Old Application Password
START_DATE	VARCHAR2(20 BYTE)	Yes	Start Date
END_DATE	VARCHAR2(100 BYTE)	Yes	End Date

INF_QRTZ_BLOB_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
TRIGGER_NAME	VARCHAR2(200)	No	Trigger name
TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group
BLOB_DATA	BLOB	Yes	Blob data

INF_QRTZ_CALENDARS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
CALENDAR_NAME	VARCHAR2(200)	No	Calendar name

CALENDAR	BLOB	No	Calendar BLOB data
----------	------	----	--------------------

INF_QRTZ_CRON_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
TRIGGER_NAME	VARCHAR2(200)	No	Trigger name
TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group
CRON_EXPRESSION	VARCHAR2(120)	No	Cron expression
TIME_ZONE_ID	VARCHAR2(80)	Yes	Time zone id

INF_QRTZ_FIRED_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
ENTRY_ID	VARCHAR2(95)	No	Entry ID
TRIGGER_NAME	VARCHAR2(200)	No	Trigger name
TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group
INSTANCE_NAME	VARCHAR2(200)	No	Instance name
FIRED_TIME	NUMBER(13)	No	Fired time
SCHED_TIME	NUMBER(13)	No	Scheduled time
PRIORITY	NUMBER(13)	No	Job priority
STATE	VARCHAR2(16)	No	Trigger state
JOB_NAME	VARCHAR2(200)	Yes	Job name
JOB_GROUP	VARCHAR2(200)	Yes	Job group
IS_NONCONCURRENT	VARCHAR2(1)	Yes	Is nonconcurrent flag
REQUESTS_RECOVERY	VARCHAR2(1)	Yes	Request recovery flag

INF_QRTZ_JOB_DETAILS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
JOB_NAME	VARCHAR2(200)	No	Job name
JOB_GROUP	VARCHAR2(200)	No	Job group
DESCRIPTION	VARCHAR2(250)	Yes	Description
JOB_CLASS_NAME	VARCHAR2(250)	No	Job class name
IS_DURABLE	VARCHAR2(1)	No	Is durable flag
IS_NONCONCURRENT	VARCHAR2(1)	No	Is nonconcurrent flag
IS_UPDATE_DATA	VARCHAR2(1)	No	Is update data flag
REQUESTS_RECOVERY	VARCHAR2(1)	No	Request recovery flag

JOB_DATA	BLOB	Yes	Job data
----------	------	-----	----------

INF_QRTZ_LOCKS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
LOCK_NAME	VARCHAR2(40)	No	Lock name

INF_QRTZ_PAUSED_TRIGGER_GRPS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group

INF_QRTZ_SCHEDULER_STATE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
INSTANCE_NAME	VARCHAR2(200)	No	Instance name
LAST_CHECKIN_TIME	NUMBER(13)	No	Last check in time
CHECKIN_INTERVAL	NUMBER(13)	No	Check in interval

INF_QRTZ_SIMPLE_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
trigger_name	VARCHAR2(200)	No	Trigger name
trigger_group	VARCHAR2(200)	No	Trigger group
repeat_count	NUMBER(7)	No	Repeat count
repeat_interval	NUMBER(12)	No	Repeat interval
times_triggered	NUMBER(10)	No	Times triggered

INF_QRTZ_SIMPROP_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
trigger_name	VARCHAR2(200)	No	Trigger name

TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group
STR_PROP_1	VARCHAR2(512)	Yes	STR property 1
STR_PROP_2	VARCHAR2(512)	Yes	STR property 2
STR_PROP_3	VARCHAR2(512)	Yes	STR property 3
INT_PROP_1	NUMBER(10)	Yes	INT property 1
INT_PROP_2	NUMBER(10)	Yes	INT property 2
LONG_PROP_1	NUMBER(13)	Yes	Long property 1
LONG_PROP_2	NUMBER(13)	Yes	Long property 2
DEC_PROP_1	NUMERIC(13,4)	Yes	Decimal property 1
DEC_PROP_2	NUMERIC(13,4)	Yes	Decimal property 2
BOOL_PROP_1	VARCHAR2(1)	Yes	Boolean property 1
BOOL_PROP_2	VARCHAR2(1)	Yes	Boolean property 2

INF_QRTZ_TRIGGERS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCHED_NAME	VARCHAR2(120)	No	Name of the scheduler
TRIGGER_NAME	VARCHAR2(200)	No	Trigger name
TRIGGER_GROUP	VARCHAR2(200)	No	Trigger group
JOB_NAME	VARCHAR2(200)	No	Job name
JOB_GROUP	VARCHAR2(200)	No	Job group
DESCRIPTION	VARCHAR2(250)	Yes	Description
NEXT_FIRE_TIME	NUMBER(13)	Yes	Next fire time
PREV_FIRE_TIME	NUMBER(13)	Yes	Previous fire time
PRIORITY	NUMBER(13)	Yes	Job priority
TRIGGER_STATE	VARCHAR2(16)	No	Trigger state
TRIGGER_TYPE	VARCHAR2(8)	No	Trigger type
START_TIME	NUMBER(13)	No	Start time
END_TIME	NUMBER(13)	Yes	End time
CALENDAR_NAME	VARCHAR2(200)	Yes	Calendar name
MISFIRE_INSTR	NUMBER(2)	Yes	Missfire instruction
JOB_DATA	BLOB	Yes	Job data

INF_REPORT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
REPORT_ID	VARCHAR2(8 BYTE)	No	The primary key of the table
SHORT_NAME	VARCHAR2(15 BYTE)	No	Short Name
REPORT_NAME	VARCHAR2(127 BYTE)	Yes	Report Name
DESCRIPTION	VARCHAR2(255 BYTE)	Yes	Description
FORMAT	VARCHAR2(4 BYTE)	Yes	Format

FILE_NAME_PATTERN	VARCHAR2(100 BYTE)	Yes	File Name Pattern
APP_MODE	VARCHAR2(2 BYTE)	Yes	Application Mode

INF_REPORT_ENTERPRISE_PCPT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
REPORT_ID	VARCHAR2(8 BYTE)	No	Composite Primary Key and The Foreign key references of INF_Report
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	Composite Primary Key and The Foreign key references of INF_Enterprise
APP_REGI_DATE	DATE	No	Application registration date
APP_UPD_DATE	DATE	No	Application update date
CREATED_BY	VARCHAR2(20 BYTE)	No	User ID of the person who has created the record
CREATION_DATE	DATE	No	Creation date of the record
UPDATED_BY	VARCHAR2(20 BYTE)	No	User ID of the person who has updated the record
UPDATE_DATE	DATE	No	Updation date of the record

INF_REPORT_GENERATION_LOG

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
REPORT_GENERATION_LOG_PK	NUMBER(10,0)	No	The primary key of the table
REPORT_ID	VARCHAR2(8 BYTE)	Yes	Foreign key references of INF_REPORT
BRANCH_PK	NUMBER(4,0)	Yes	Foreign key references of INF_BRANCH
GENERATED_FILE_PATH	VARCHAR2(255 BYTE)	Yes	Path where console report generated
STATUS	VARCHAR2(15 BYTE)	Yes	Status of the report
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	Foreign key references of INF_ENTERPRISE

INF_ROUTE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ROUTE_PK	NUMBER(10,0)	No	The primary key of the table
ROUTE_ID	VARCHAR2(25 BYTE)	No	Specify the unique Route Id
ROUTE_DESCRIPTION	VARCHAR2(128 BYTE)	Yes	Description of Route

INF_ROUTE_ENDPOINT_PARTICIPANT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ROUTE_ENDPOINT_PARTICIPANT_PK	NUMBER(10,0)	No	The primary key of the table
ROUTE_PK	NUMBER(10,0)	No	Foreign key references of INF_ROUTE
ENDPOINT_PK	NUMBER(10,0)	No	Foreign key references of INF_ENDPOINT
ENDPOINT_TYPE	VARCHAR2(3 BYTE)	No	Endpoint Type whether IN,OUT or ERROR

INF_SAVED_QUERY

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SAVED_QUERY_PK	NUMBER(10,0)	No	The primary key of the table
SCREEN_ID	VARCHAR2(5 BYTE)	No	Screen Id of the corresponding screens
VERSION_NO	NUMBER(10,0)	No	Version No
CRITERIA_NAME	VARCHAR2(100 BYTE)	No	Criteria Name
CRITERIA	CLOB	No	Criteria Json
URL	VARCHAR2(200 BYTE)	No	Corresponding URL of Saved Query
STATUS	VARCHAR2(6 BYTE)	Yes	Status of the saved query
EMPLOYEE_PK	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE

INF_SAVED_TEMPLATE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SAVED_TEMPLATE_PK	NUMBER(10)	No	Primary key of this table
SCREEN_ID	VARCHAR2(5)	No	Screen ID of the entry screen whose template is saved
VERSION_NO	NUMBER(10)	No	Version no of the entry screen whose template is saved
CRITERIA_NAME	VARCHAR2(100)	No	Name of the saved template
CRITERIA	CLOB	No	JSON data of the command form of the entry screen
URL	VARCHAR2(200)	No	URL for invoking the entry screen
STATUS	VARCHAR2(6)	Yes	Status field
EMPLOYEE_PK	NUMBER(10)	No	Employee PK
SHARE_TEMPLATE	VARCHAR2(3)	No	Share template flag. Y=Share N=Not Shared

INF_SERVICE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SERVICE_PK	NUMBER(10,0)	No	The primary key of the table
SERVICE_ID	VARCHAR2(10 BYTE)	No	Name of the Service
SERVICE_DESCRIPTION	VARCHAR2(30 BYTE)	Yes	Description of Service

INF_SERVICE_ROUTE_PARTICIPANT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SERVICE_ROUTE_PK	NUMBER(10,0)	No	The primary key of the table
SERVICE_PK	NUMBER(10,0)	Yes	Foreign key references of INF_SERVICE
ROUTE_PK	NUMBER(10,0)	Yes	Foreign key references of INF_ROUTE

INF_UI_MENU

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
MENU_PK	NUMBER(5,0)	No	The primary key of menu
SCREEN_KEY	NUMBER(5,0)	Yes	The foreign key of INF_SCREEN
MENU_NAME	VARCHAR2(20 BYTE)	Yes	The name of the menu
MENU_DESCRIPTION	VARCHAR2(100 BYTE)	Yes	The description of the menu
MENU_PARENT_PK	NUMBER(5,0)	Yes	The parent menu pk for a child
DISPLAY_SEQ	NUMBER(5,0)	No	The order of display in UI
COMPONENT_ID	VARCHAR2(3 BYTE)	Yes	The component ID from INF_COMPONENT
ACTION_ID	VARCHAR2(8 BYTE)	Yes	The foreign key of INF_ACTION
MENU_ID	VARCHAR2(30 BYTE)	No	The short abbreviation of an menu

INF_UI_PREF_GROUP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
PREF_GROUP_PK	NUMBER(10,0)	No	The primary key of the table
GROUP_NAME	VARCHAR2(100 BYTE)	No	Group Name
DISPLAY_SEQ	NUMBER(5,0)	No	Display Sequence

INF_UI_PREF_GROUP_NAME_XREF

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
PREF_GROUP_NAME_XREF_PK	NUMBER(10,0)	No	The primary key of the table
PREF_GROUP_PK	NUMBER(10,0)	No	Foreign key references of INF_UI_PREF_GROUP
LANGUAGE_CODE	VARCHAR2(15 BYTE)	No	Foreign key references of INF_LANGUAGE
GROUP_NAME	VARCHAR2(100 BYTE)	No	Last Name of the Employee

INF_UI_SCREEN

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
SCREEN_KEY	NUMBER(5,0)	No	The primary key
SCREEN_ID	VARCHAR2(5 BYTE)	No	The ID to be used across UI
SCREEN_NAME	VARCHAR2(30 BYTE)	No	The name of the screen
SHORT_NAME	VARCHAR2(20 BYTE)	Yes	The short name of a screen
DESCRIPTION	VARCHAR2(255 BYTE)	Yes	The description of the screen
VERSION_NUMBER	NUMBER(10,2)	No	The version no of the screen
NEXT_SCREEN_PK	NUMBER(10,0)	Yes	NEXT SCREEN PK (BASICALLY THE RESULT SCREEN OF CURRENT SCREEN ID)

COMPONENT_ID	VARCHAR2(6 BYTE)	No	The component id
--------------	------------------	----	------------------

INF_UPLOAD_TEMPLATE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
TEMPLATE_PK	NUMBER(10,0)	No	The primary key of the table
ACTION_ID	VARCHAR2(20 BYTE)	No	Action ID
TEMPLATE_ID	VARCHAR2(20 BYTE)	Yes	Template ID
TEMPLATE_DESCRIPTION	VARCHAR2(200 BYTE)	Yes	Template Description
TEMPLATE_DATA	CLOB	No	Template data
STATUS	VARCHAR2(15 BYTE)	No	Template Status
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	Foreign key references of INF_ENTERPRISE

INF_UPLOAD_TMPL_MAPPING

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
MAPPING_PK	NUMBER(10,0)	No	The primary key of the table
TEMPLATE_PK	NUMBER(10,0)	No	Foreign key references of INF_UPLOAD_TEMPLATE
MAPPING_DATA	CLOB	No	Mapping Json Data
MAPPING_ID	VARCHAR2(20 BYTE)	Yes	Template Mapping Name
MAPPING_DESCRIPTION	VARCHAR2(200 BYTE)	Yes	Template Mapping Description
STATUS	VARCHAR2(15 BYTE)	No	Status
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	Foreign key references of INF_ENTERPRISE
DOCUMENT_FORMAT	VARCHAR2(6 BYTE)	No	Document Format whether CVS,XLS,XLSX
HEADER_ROW_NO	NUMBER(5,0)	Yes	Header Row Number
DATA_START_ROW_NO	NUMBER(5,0)	Yes	Data Start Row Number
SHEET_NAME	VARCHAR2(30 BYTE)	Yes	Sheet Name
HEADERS_PRESENT	CHAR(1 BYTE)	No	(Y/N) Headers are present or not

3.3.2.4 Dashboard (DBD) Data Model

DBD_CHART

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CHART_PK	NUMBER(10,0)	No	The primary key of the table
DBD_CHART_NAME	VARCHAR2(100 BYTE)	No	Chart Name
DBD_CHART_URL	VARCHAR2(200 BYTE)	No	Url of the Chart
REQUEST_LAG_TIME_MSEC	NUMBER(10,0)	Yes	Request Lag Time
STATUS	VARCHAR2(6 BYTE)	No	Status of the chart
DATA_URL	VARCHAR2(200 BYTE)	No	Url of Data

DBD_CHART_APPLN_ROLE_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
APPL_ROLE_PK	NUMBER(10,0)	No	Foreign key reference to INF_APPLICATION_ROLE
DBD_CHART_PK	NUMBER(10,0)	No	Foreign key reference to DBD_CHART

DBD_CUSTOM_CHART

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CUSTOM_CHART_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	Foreign key reference to DBD_WORKSPACE_WIDGET_PTCP
DBD_CHART_PK	NUMBER(10,0)	No	Foreign key reference to DBD_CHART
DBD_CHART_NAME	VARCHAR2(100 BYTE)	Yes	Chart Name
STATUS	VARCHAR2(6 BYTE)	No	Status of the chart
DBD_CHART_TYPE	VARCHAR2(10 BYTE)	Yes	Type of chart
DBD_CHART_BG_COLOR	VARCHAR2(10 BYTE)	Yes	Background color of chart

DBD_CUSTOM_FEED

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CUSTOM_FEED_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	Yes	PK of DBD WORKSPACE WIDGET
DBD_FEED_PK	NUMBER(10,0)	Yes	Foreign key references of DBD_FEED
DBD_FEED_NAME	VARCHAR2(100 BYTE)	Yes	Dashboard Feed Name
STATUS	VARCHAR2(6 BYTE)	Yes	Feed Status
FEED_ORDER	NUMBER(5,0)	Yes	Feed Order

DBD_CUSTOM_NOTIFICATION

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CUSTOM_NOTIFICATION_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	Foreign key references of DBD_WORKSPACE_WIDGET_PK
DBD_NOTIF_INFO_COLOR	VARCHAR2(10 BYTE)	Yes	Color of notification information
DBD_NOTIF_WARN_COLOR	VARCHAR2(10 BYTE)	Yes	Color of notification warning
DBD_NOTIF_ERROR_COLOR	VARCHAR2(10 BYTE)	Yes	Color of notification error
STATUS	VARCHAR2(6 BYTE)	No	Status of custom notification

DBD_CUSTOM_SAVED_QUERY

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CUSTOM_SAVED_QUERY_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	Foreign key references of DBD_WORKSPACE_WIDGET_PTCP
SAVED_QUERY_PK	NUMBER(10,0)	No	Foreign key references of INF_SAVED_QUERY
CRITERIA_NAME	VARCHAR2(400 BYTE)	No	Criteria Name
STATUS	VARCHAR2(6 BYTE)	No	Status
FEED_ORDER	NUMBER(2,0)	No	Feed Order

DBD_CUSTOM_SAVED_TEMPLATE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_CUSTOM_SAVED_TEMPLATE_PK	NUMBER(10,0)	No	PK for the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	PK of the DBD_WORKSPACE_WIDGET_PTCP table
SAVED_TEMPLATE_PK	NUMBER(10,0)	No	PK of the INF_SAVED_TEMPLATE table
CRITERIA_NAME	VARCHAR2(400 BYTE)	No	Save Template criteria name
STATUS	VARCHAR2(6 BYTE)	No	Status
FEED_ORDER	NUMBER(2,0)	No	Feed order

DBD_EMP_TASK_TYPE_CONFIG

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
EMP_TASK_TYPE_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET	NUMBER(10,0)	No	Foreign key references of DBD_WORKSPACE_WIDGET_PTCP
DBD_TASK_TYPE	NUMBER(10,0)	No	Foreign key references of DBD_TASK_TYPE
USER_LIST	VARCHAR2(200 BYTE)	Yes	List of User
DATE_OFFSET	NUMBER(10,0)	No	Number of records to be displayed within offset

DBD_FEED

General Table Structure

COLUMN_NAME	DATA_TYPE	NULLABLE	DESCRIPTION
DBD_FEED_PK	NUMBER(10,0)	No	The primary key
DBD_FEED_NAME	VARCHAR2(100 BYTE)	No	The feed name to be displayed on the dashboard
DBD_FEED_URL	VARCHAR2(200 BYTE)	No	The URL to be fired upon the click on the feed name
VIEW_RESULT_COLUMN	VARCHAR2(30 BYTE)	No	Defines the count
REQUEST_LAG_TIME_MSEC	NUMBER(10,0)	Yes	Request Lag Time Interval
STATUS	VARCHAR2(6 BYTE)	No	The status of the feed
FEED_VIEW	VARCHAR2(30 BYTE)	No	The view of the feed

DATA_URL	VARCHAR2(200 BYTE)	No	The URL to get the counters
----------	--------------------	----	-----------------------------

DBD_FEED_APPLN_ROLE_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
APPL_ROLE_PK	NUMBER(10,0)	No	The foreign key for INF_APPLICATION_ROLE
DBD_FEED_PK	NUMBER(10,0)	No	The foreign key for DBD_FEED

DBD_FEED_NAME_CROSS_REF

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_FEED_NAME_CROSS_REF_PK	NUMBER(10,0)	No	PK for the table
DBD_FEED_PK	NUMBER(10,0)	No	FK for the DBD_FEED.DBD_FEED_PK
DBD_FEED_NAME	VARCHAR2(100 BYTE)	No	Display name of the feed
LANGUAGE_CODE	VARCHAR2(15 BYTE)	No	Language code

DBD_GROUP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_GROUP_PK	NUMBER(10,0)	No	The primary key of the table
DBD_GROUP_NAME	VARCHAR2(100 BYTE)	No	Dashboard Group Name
STATUS	VARCHAR2(6 BYTE)	No	Status
ENABLE_FLAG	CHAR(1 BYTE)	Yes	Enable the flag
DBD_GROUP_ORDER	NUMBER(2,0)	No	Dashboard Group Order
EMPLOYEE_PK	NUMBER(10,0)	No	Employee Primary Key

DBD_MENU_SHORTCUT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_MENU_SHORTCUT_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	Dashboard Workspace Widget Primary Key
MENU_PK	NUMBER(5,0)	Yes	Foreign key references of INF_UI_MENU
SHORTCUT_NAME	VARCHAR2(400 BYTE)	Yes	Last Name of the Employee
STATUS	VARCHAR2(6 BYTE)	No	First Name of the Employee
FEED_ORDER	NUMBER(2,0)	No	Order of Feed

DBD_NOTIF_DSTN_EMP_PCPT_PK

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
NOTIFICATION_DESTINATION_PK	NUMBER(10,0)	No	Foreign key references of INF_NOTIFICATION_DESTINATION
EMPLOYEE_PK	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE
DBD_NOTIF_DSTN_EMP_PCPT_PK	NUMBER(10,0)	No	The primary key
STATUS	VARCHAR2(6 BYTE)	No	Status field

DBD_ROUTINE_TASK

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ROUTINE_TASK_PK	NUMBER(10,0)	No	The primary key of the table
TASK_INTERVAL	NUMBER(10,0)	No	Task Interval value
TASK_DURATION	NUMBER(10,0)	No	Task Duration Value
STATUS	VARCHAR2(6 BYTE)	No	Status
ASSIGNED_BY	NUMBER(10,0)	No	Assigned by value
ASSIGNED_TO	NUMBER(10,0)	No	Assigned To Value

DBD_TASK_DETAIL

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
TASK_DETAIL_PK	NUMBER(10,0)	No	The primary key of the table
ASSIGNED_TO	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE
TASK_NAME	VARCHAR2(50 BYTE)	No	Name of the Task
TASK_ID	VARCHAR2(20 BYTE)	No	ID of the Task
DUE_DATE	DATE	No	Due Date
ASSIGNED_BY	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE
TASK_STATUS_PK	NUMBER(10,0)	No	Foreign key references of DBD_TASK_STATUS
IS_ROUTINE	CHAR(1 BYTE)	Yes	Whether Routing task or not
ROUTINE_TASK	NUMBER(10,0)	Yes	Foreign key references of DBD_ROUTINE_TASK

DBD_TASK_STATUS

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
TASK_STATUS_PK	NUMBER(10,0)	No	The primary key of the table
TASK_STATUS_ID	VARCHAR2(10 BYTE)	No	Task Status Id
TASK_STATUS_NAME	VARCHAR2(30 BYTE)	No	Task Status Name

DBD_TASK_TYPE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_TASK_TYPE_PK	NUMBER(10,0)	No	The primary key
DBD_WIDGET_TYPE_PK	NUMBER(10,0)	No	The foreign key of DBD_WIDGET_TYPE
DBD_TASK_TYPE_ID	VARCHAR2(10 BYTE)	No	The unique name ID of task type
DBD_TASK_TYPE_DESC	VARCHAR2(200 BYTE)	No	The description of task type
DATA_URL	VARCHAR2(200 BYTE)	No	The url to be fetched for data population
TASK_ORDER	NUMBER(2,0)	Yes	The sequence of task to be displayed in UI
USER_LIST	VARCHAR2(200 BYTE)	Yes	The list of user permitted to
REQUEST_LAG_TIME_MSEC	NUMBER(10,0)	Yes	Request Lag Time Interval
DATE_OFFSET	NUMBER(10,0)	No	Number of records to be displayed within offset

DBD_TASK_TYPE_EMP_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_TASK_TYPE_EMP_PTCP_PK	NUMBER(10,0)	No	The primary key of the table
DBD_TASK_TYPE_PK	NUMBER(10,0)	No	Foreign key references of DBD_TASK_TYPE
EMPLOYEE_PK	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE
DATE_OFFSET	NUMBER(10,0)	No	Number of records to be displayed within offset
USER_LIST	VARCHAR2(200 BYTE)	Yes	List of users

DBD_TEMP_QUERY_PARAM

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	The Composite primary key of the table and Enterprise ID
EMPLOYEE_PK	NUMBER(10,0)	No	The Composite primary key of the table and Enterprise ID
QUERY_DATE	DATE	No	Date of query
EXM_GROUP_ID	VARCHAR2(20 BYTE)	Yes	
DEFAULT_INST_CODE_TYPE	VARCHAR2(20 BYTE)	Yes	Default institute code type
ALL_INST_CODE_TYPE	VARCHAR2(20 BYTE)	Yes	Alternate institute code type

DBD_WIDGET

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_PK	NUMBER(10,0)	No	The primary key
DBD_WIDGET_NAME	VARCHAR2(400 BYTE)	No	The name of the widget
DBD_WIDGET_TYPE	NUMBER(10,0)	No	The foreign key of DBD_WIDGET_TYPE
STATUS	VARCHAR2(6 BYTE)	No	The status of the widget
ENABLE_FLAG	CHAR(1 BYTE)	Yes	Whether enable

REQUEST_LAG_TIME_MSEC	NUMBER(10,0)	Yes	Time interval for request lag
SYSTEM_DEFINED	VARCHAR2(10 BYTE)	No	Whether system defined
AVAILABLE_TO_USER	VARCHAR2(10 BYTE)	No	Whether available to user

DBD_WIDGET_EMP_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_PK	NUMBER(10,0)	No	The first part of the composite primary key along with EMPLOYEE_PK & Foreign key reference to DBD_WIDGET Table
EMPLOYEE_PK	NUMBER(10,0)	No	The second part of the composite primary key along with DBD_WIDGET_PK & Foreign key reference to INF_EMPLOYEE Table

DBD_WIDGET_FEED_PARTCPNT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_PK	NUMBER(10,0)	No	Composite primary key of the table and Foreign key references of DBD_WIDGET
DBD_FEED_PK	NUMBER(10,0)	No	Composite primary key of the table and Foreign key references of DBD_FEED
FEED_ORDER	NUMBER(2,0)	No	Order of the feed

DBD_WIDGET_TYPE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_TYPE_PK	NUMBER(10,0)	No	The primary key
DBD_WIDGET_TYPE_NAME	VARCHAR2(50 BYTE)	No	The widget type name
DBD_WIDGET_TYPE_DESC	VARCHAR2(200 BYTE)	No	The widget type description
IS_FEED_DRAGABLE	VARCHAR2(10 BYTE)	No	Whether the feed is dragable
IS_FEED_DROPABLE	VARCHAR2(10 BYTE)	No	Whether the feed is droppable
ALL_FEEDS_URL	VARCHAR2(200 BYTE)	Yes	URL for feeds
MULTI_WIDGET_MODE	VARCHAR2(10 BYTE)	No	Whether multi widget mode
DATA_URL	VARCHAR2(200 BYTE)	No	Data to be fetched from
IS_FEED_SELECTABLE	VARCHAR2(10 BYTE)	No	Whether feed is selectable
IS_FEED_UNSELECTABLE	VARCHAR2(10 BYTE)	No	Whether feed is unselectable
BACKGROUND_COLOR	VARCHAR2(20 BYTE)	No	Background color of feed
foreground_color	VARCHAR2(100 BYTE)	Yes	Foreground color of feed
LINK_COLOR	VARCHAR2(100 BYTE)	Yes	Link color
IS_FEED_REFRESHABLE	VARCHAR2(10 BYTE)	No	Whether feeds can be refreshed
REFRESH_INTERVAL_MIN	NUMBER(10,0)	No	Interval for feed refreshing
WIDTH	NUMBER(5,0)	No	Width of the widget
HEIGHT	NUMBER(5,0)	No	Height of the widget
CUSTOM_WIDGET	VARCHAR(10 BYTE)	No	Custom Widget

DBD_WIDGET_TYPE_APP_ROLE_PT

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_TYPE_PK	NUMBER(10,0)	No	The first part of the composite primary key along with WIDGET_TYPE_PK
APPLICATION_ROLE_PK	NUMBER(5,0)	No	The second part of the composite primary key along with APPLICATION_ROLE_PK
STATUS	VARCHAR2(6 BYTE)	No	Status field

DBD_WIDGET_TYPE_EMP_CONFIG

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WIDGET_TYPE_PK	NUMBER(10,0)	No	Foreign key of DBD_WIDGET_TYPE
EMPLOYEE_PK	NUMBER(10,0)	No	Foreign key of INF_EMPLOYEE
STATUS	VARCHAR2(6 BYTE)	No	The status of the record

DBD_WORKSPACE

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WORKSPACE_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_NAME	VARCHAR2(100 BYTE)	No	Workspace Name

DBD_WORKSPACE_EMP_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WORKSPACE_PK	NUMBER(10,0)	No	Composite primary key of the table and Foreign key references of DBD_WORKSPACE
EMPLOYEE_PK	NUMBER(10,0)	No	Composite primary key of the table and Foreign key references of INF_EMPLOYEE
STATUS	VARCHAR2(6 BYTE)	No	Foreign key references of DBD_TASK_TYPE
DBD_WORKSPACE_ORDER	NUMBER(10,0)	No	List of User

DBD_WORKSPACE_WIDGET_PTCP

General Table Structure

COLUMN NAME	DATA TYPE	NULLABLE	DESCRIPTION
DBD_WORKSPACE_WIDGET_PK	NUMBER(10,0)	No	The primary key of the table
DBD_WORKSPACE_PK	NUMBER(10,0)	No	Foreign key references of DBD_WORKSPACE
DBD_WIDGET_PK	NUMBER(10,0)	No	Foreign key references of DBD_WIDGET
STATUS	VARCHAR2(6 BYTE)	Yes	Status
DBD_WIDGET_ORDER	NUMBER(2,0)	No	Number of records to be displayed within offset

EMPLOYEE_PK	NUMBER(10,0)	No	Foreign key references of INF_EMPLOYEE
DBD_WIDGET_NAME	VARCHAR2(400 BYTE)	Yes	Dashboard Widget Name
BACKGROUND_COLOR	VARCHAR2(20 BYTE)	Yes	Background Color
FOREGROUND_COLOR	VARCHAR2(20 BYTE)	Yes	Foreground Color
SHOW_GRAPH	VARCHAR2(10 BYTE)	Yes	data to populate graph
LINK_COLOR	VARCHAR2(20 BYTE)	Yes	any link color
WIDGET_WIDTH	NUMBER(5,0)	Yes	Width of Widget
WIDGET_HEIGHT	NUMBER(5,0)	Yes	Height of Widget

3.3.2.5 Batch Tables

We are using typical Spring provided tables for batch framework.

For documentation please refer to the following link

<http://docs.spring.io/spring-batch/reference/html/metaDataSchema.html>

3.3.3 MVC/UI Layer

3.3.3.1 Menu and view Definition

Introduction

Menu configuration and details available in [this page](#).

Linked page display activity needs to be done for menu.

Description

Main action for design for menu:

- Menu Definition
- Screen Key Definition
- Basic Action Definition
- Sub Action Definition
 - This definition is defined in INF_ACTION_CONSOLIDATION table : Below table provides one definition for Employee A

ACTION_ID	SUB_ACTION_ID	DISPLAY_SEQ	Note
EMPLQR	EMPLAM	1	For Amend Sub action for Employee
EMPLQR	EMPLCX	2	For Cancel Sub action for Employee

Access control get defined in INF_ACTION_ROLE_PARTICIPANT table.

- Application Role and Action Mapping: Access control (INF_ACTION_ROLE_PARTICIPANT).

View Definition

View definition get defined in views.xml available for jspx pages.

One example provided below:

```

views.xml

...
<definition name="userEntry" extends="wizard-page">
  <put-attribute name="wizard-page"
    value="/WEB-INF/views/inf/user/userEntry.jspx" />
</definition>
<!-- User amend page -->
<definition name="userAmend" extends="wizard-page">
  <put-attribute name="wizard-page"
    value="/WEB-INF/views/inf/user/userAmend.jspx" />
</definition>
<!-- User General detail page -->
<definition name="userGeneralDetail" extends="wizard-page">
  <put-attribute name="wizard-page"
    value="/WEB-INF/views/inf/user/userDetail.jspx" />
</definition>
...

```

3.3.3.2 Menu Shortcut Configuration Guide

Introduction

This document describes how to configure menu to have keyboard shortcuts. Finch implements the menu shortcut based on application configuration for menu.

How to Configure Menu Shortcut

Currently finch provides menu shortcut only for the first level menu. Shortcut character will be underlined in the menu name. Application has to provide the character to be underlined for the menu.

finch gets the menu name from the menu.properties file. So application has to give menu name in menu.properties or menu_ja.properties file.

If an entry for a menu name is not present in menu.properties file then finch gets the menu name from MENU_NAME column of INF_UI_MENU table. So application need to enter menu name in INF_UI_MENU table.

Following changes are required for menu shortcut

- application has to give a '&' before the character of the menu and finch will assign shortcut for that menu.

For example if menu name is "&Sys Control" then menu will be displayed as "Sys Control"

- If application wants to display '&' in menu name then provide '¥¥&' in the menu.

For example if menu name is "&Sys ¥¥& Control" then menu will be displayed as "Sys & Control"

- If application did not provide '&' character in menu name then finch will not create any menu shortcut

How to Configure Menu Shortcut Key

Menu can be accessed from keyboard by pressing a key+<underline_character_of_menu> like alt+s or ctrl+shift+s

Application can configure the key for the menu short. Shortcut key for a menu will be key+<underline_character_of_menu> .

By default finch bind menu shortcut in alt key. If application wants to change the key then application has to make following entry in app-init-config.js file

FINCH.menuShortcutKey = <key>

For example - `FINCH.menuShortcutKey = "shift"`

In that case shortcut for a menu will be shift+<underline_character>

Examples

1) TRD.name=Tr&ade

menu will be displayed as Trade and menu shortcut key is alt+a

2) SYS.name=Sys ¥¥& &Control

menu will be displayed as Sys & Control and shortcut will be alt+c

3) TRD.name=Trade&

An WARN type error will be logged and menu shortcut will not be assigned and menu will be displayed as Trade

4) TRD.name=Tr& ade

An WARN type error will be logged and menu shortcut will not be assigned and menu will be displayed as Tr ade

5) TRD.name=Tr&&ade

An WARN type error will be logged and menu shortcut will not be assigned and menu will be displayed as Trade

6) TRD.name=Tr&¥¥&ade

An WARN type error will be logged and menu shortcut will not be assigned and menu will be displayed as Tr&ade

Recommendation

Browser also have menu shortcut for accessing browser's menu and **finch does not restrict browser keyboard shortcut**.

So finch recommends not to use menu shortcut for their application menu that is also a browser keyboard shortcut or window's standard keyboard shortcuts like ctrl+c , ctrl+a , alt+f etc.

If application use those key combination for their menu shortcut then both actions(browser action or window's action and application's action) will be perform against that combination.

Menu labels will appear in the next line if entered menu label is too long and with space. But it will show ellipsis if the entered menu label is without any space and too long

3.3.3.3 Screen Design

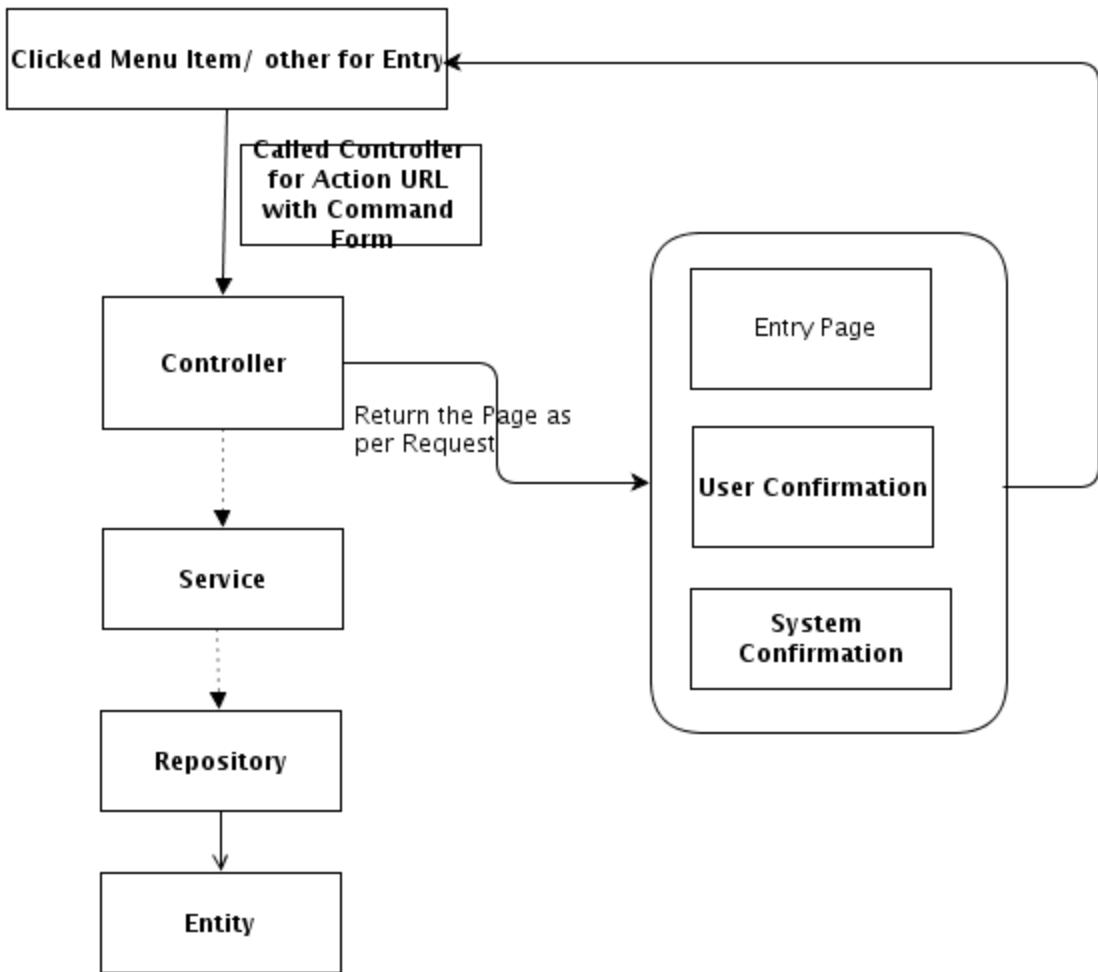
The root page 3.3.4.3 Screen Design could not be found in space finch 2.

Define following in database

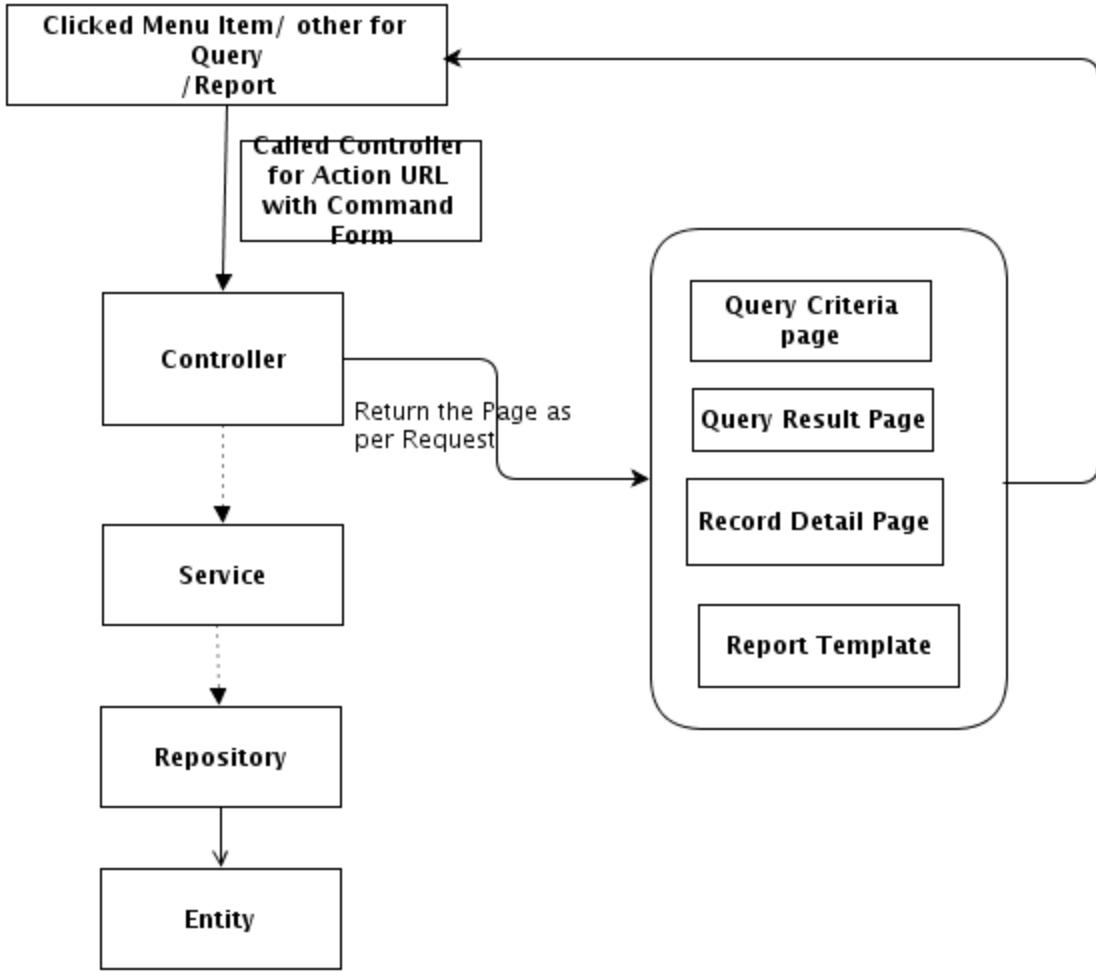
1. menu item with Access control.
2. Define the INF Action with URL, URL pattern, and other properties.
3. Sub Action

Flow for Complete UI Event:

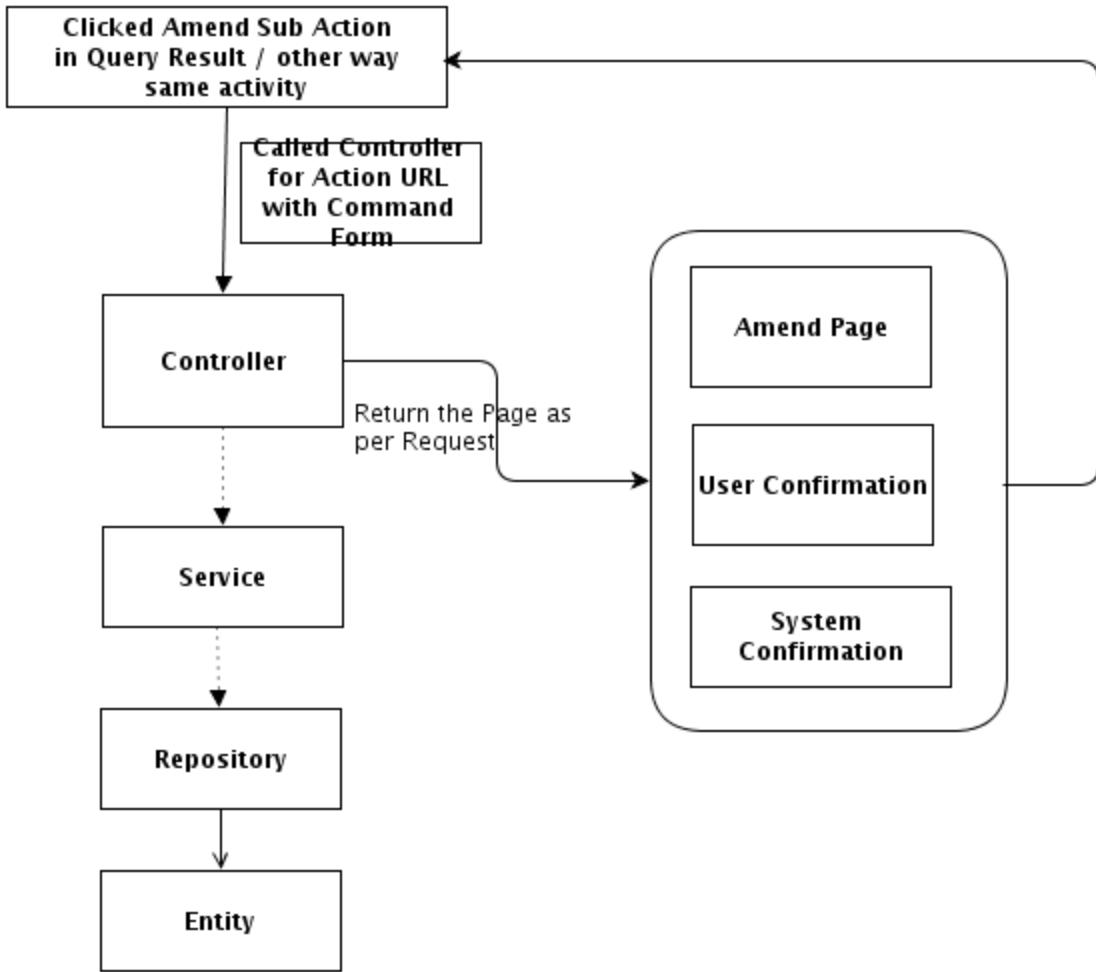
Entry:



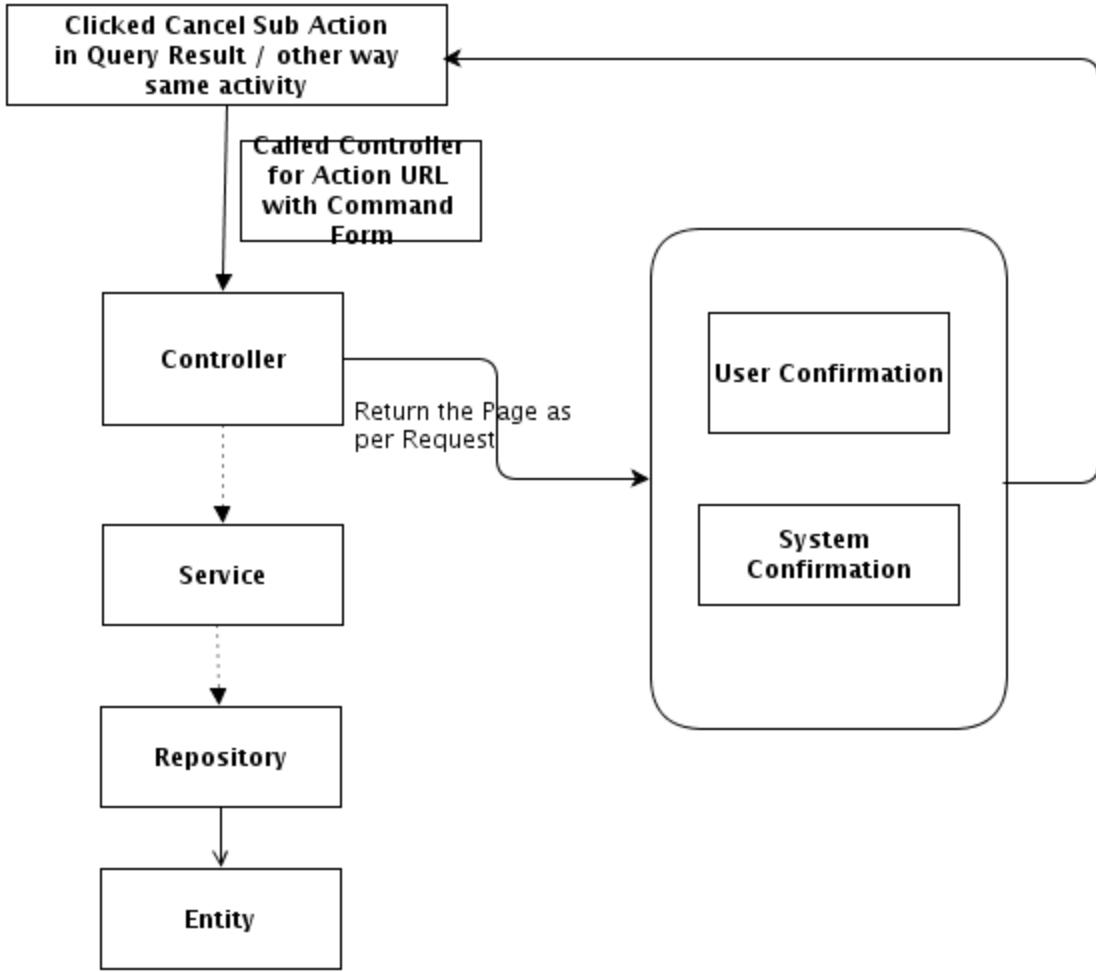
Query:



Amend:



Cancel:



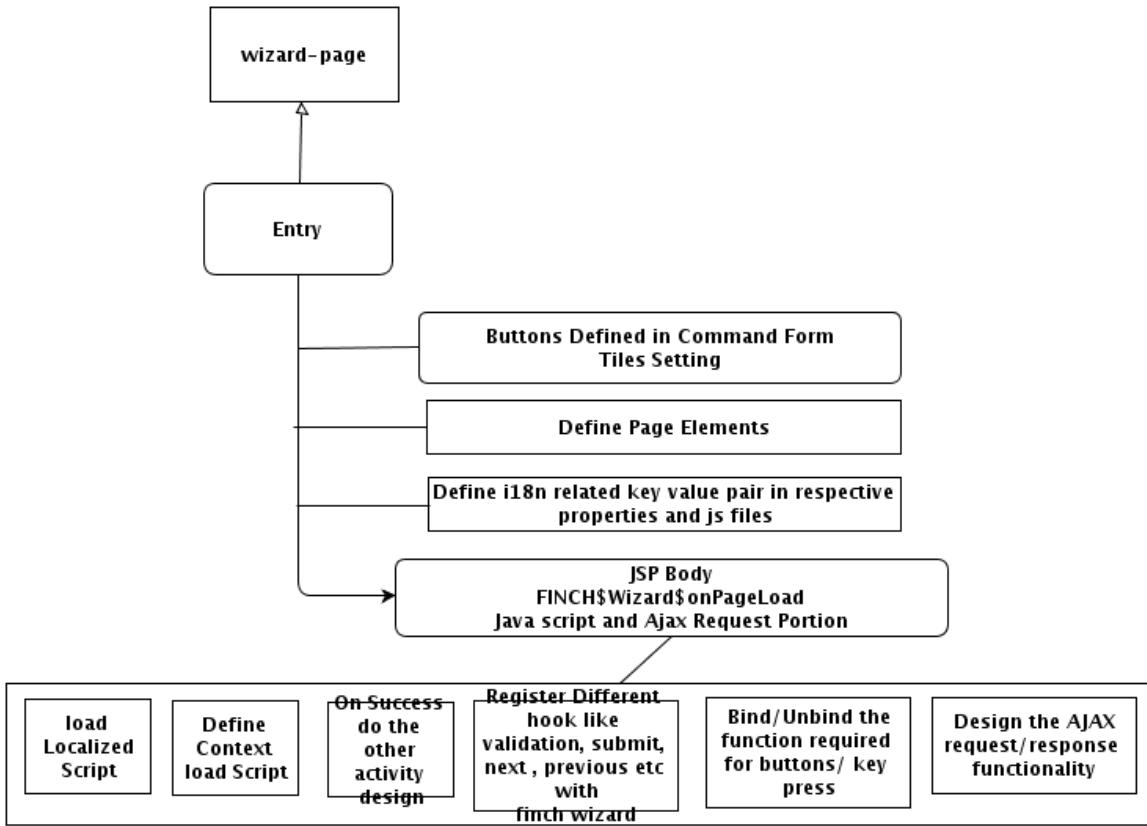
MVC UI

Entry:

Define the Pages and respective tiles definition in views.xml.

In Command Form default constructor set the wizard with each applicable page flow and buttons:

- Define the page tiles for each Entry/User Confirmation/ System Confirmation with name defined in tool
- Buttons for each tiles page
- Define Finch Wizard Mode
- Create Finch Wizard with Title, Tile Name, modes, Pages, buttons.

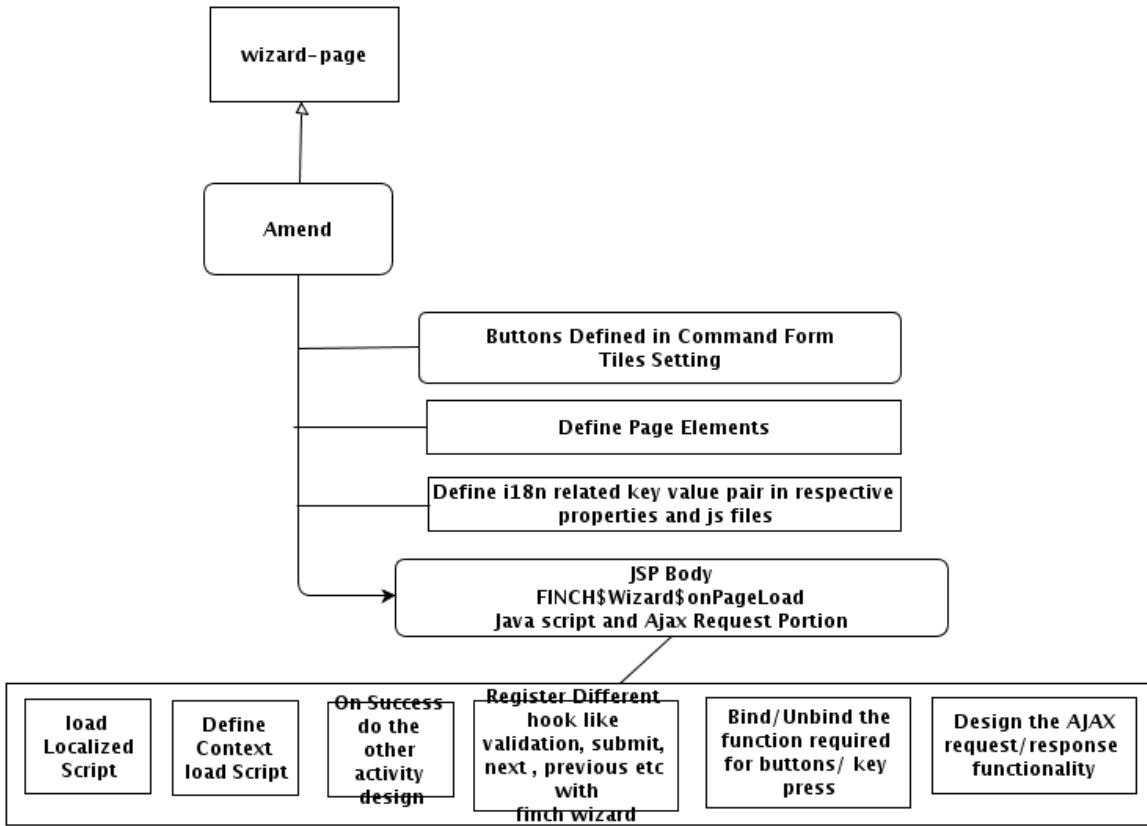


Amend:

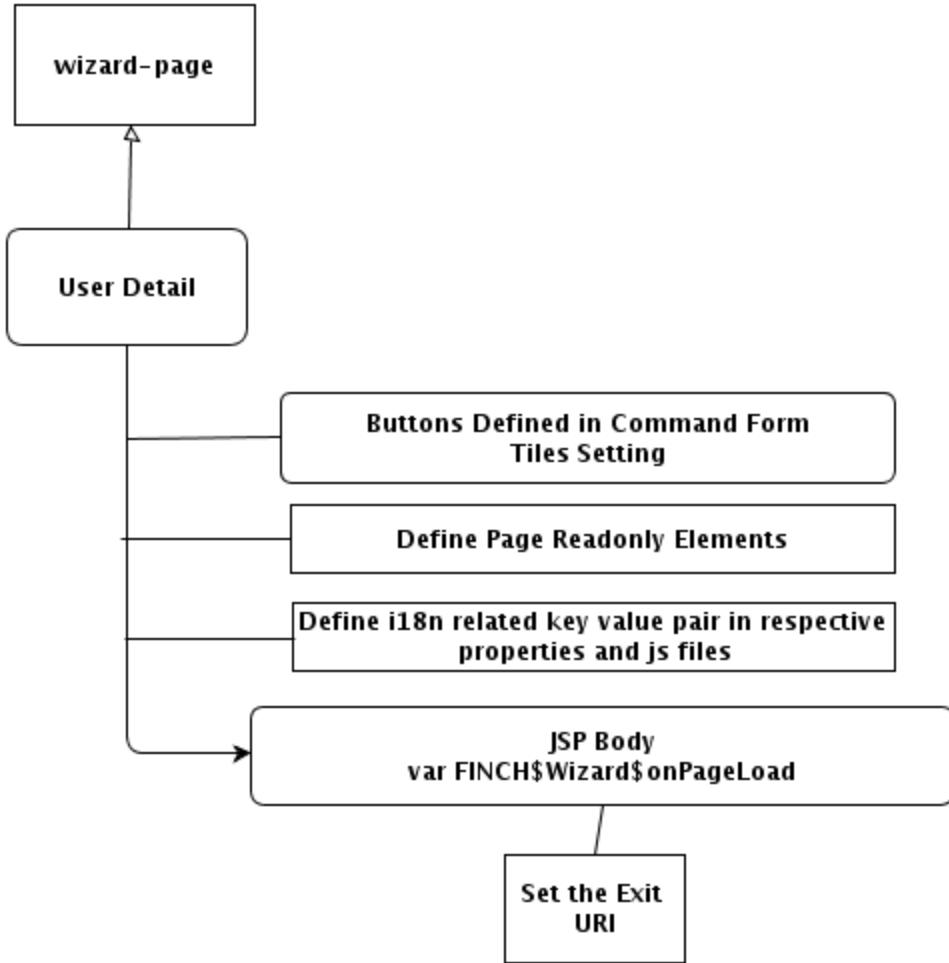
Define the Pages and respective tiles definition in views.xml.

In Command Form default constructor set the wizard with each applicable page flow and buttons:

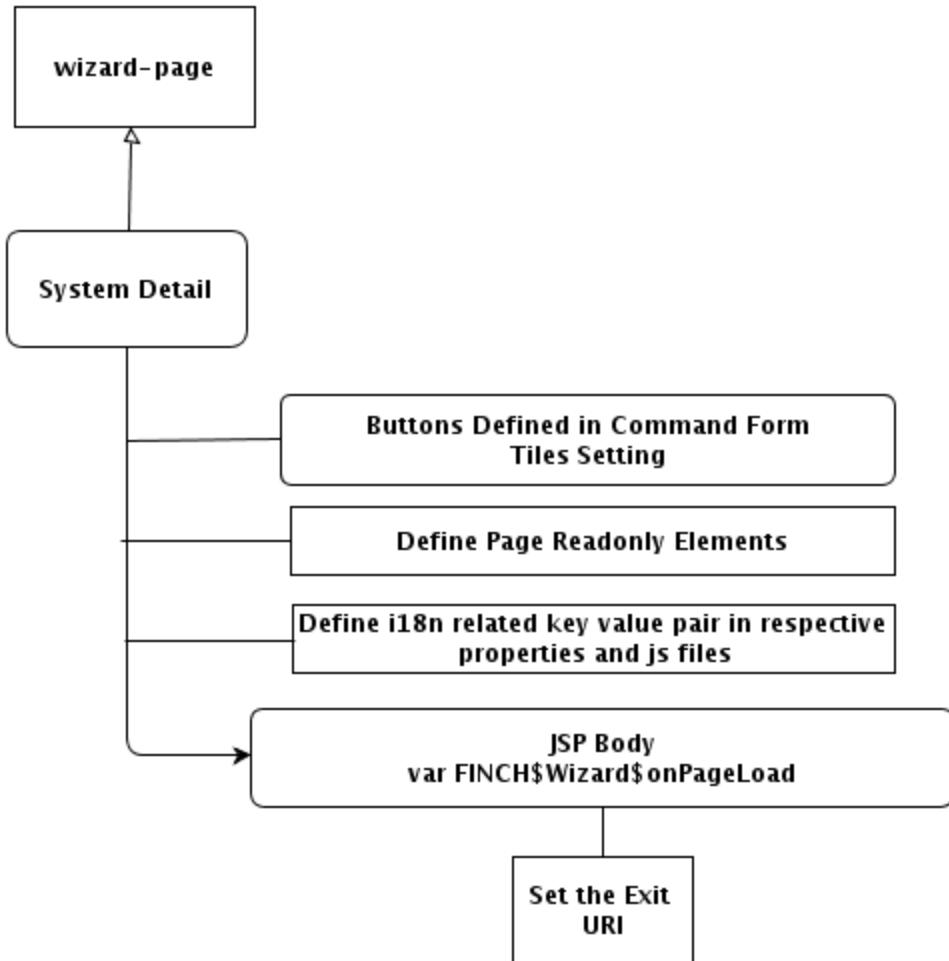
- Define the page tiles for each Entry/User Confirmation/ System Confirmation with name defined in tool
- Buttons for each tiles page
- Define Finch Wizard Mode
- Create Finch Wizard with Title, Tile Name, modes, Pages, buttons.



User Confirmation:

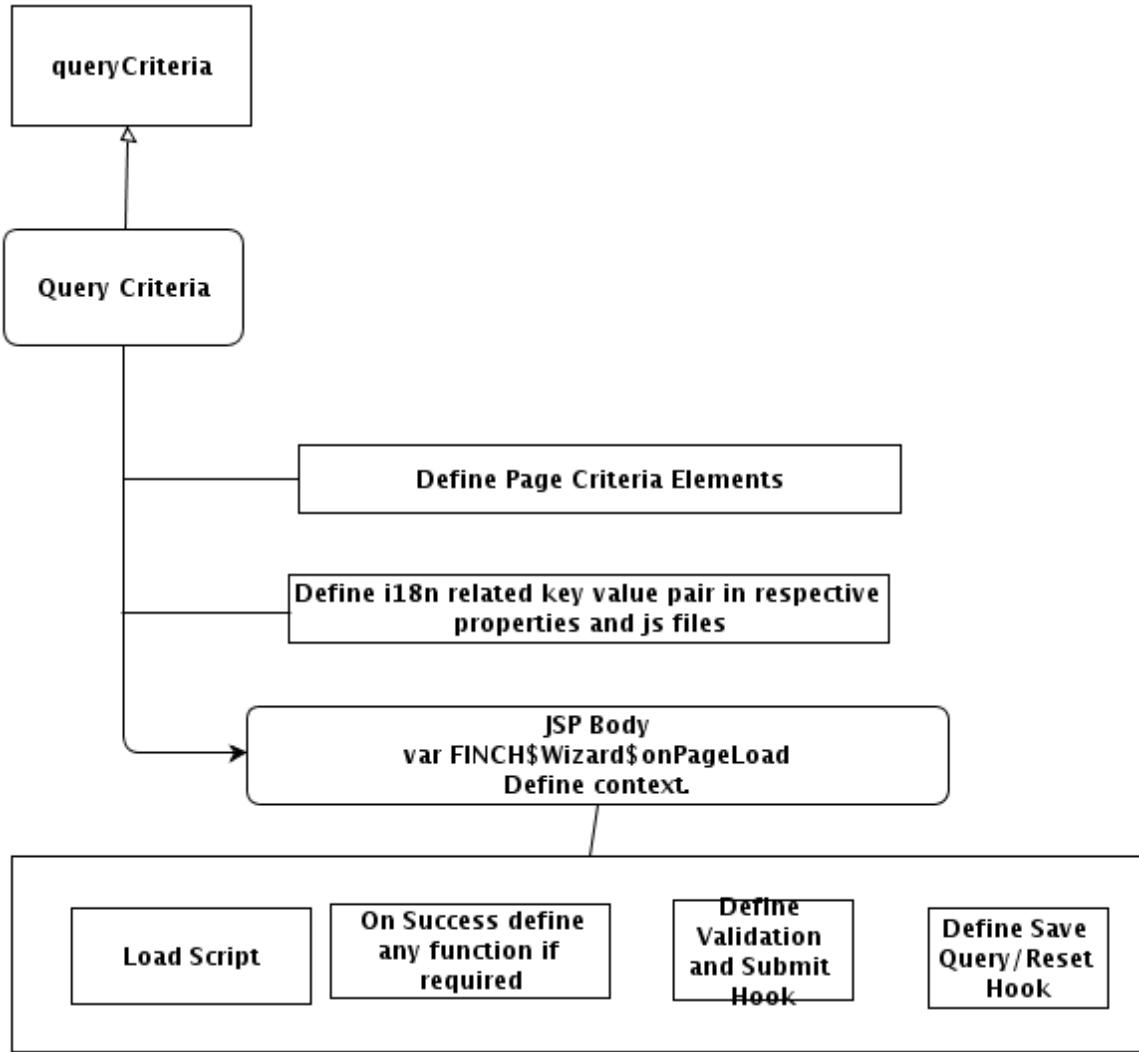


System Confirmation:

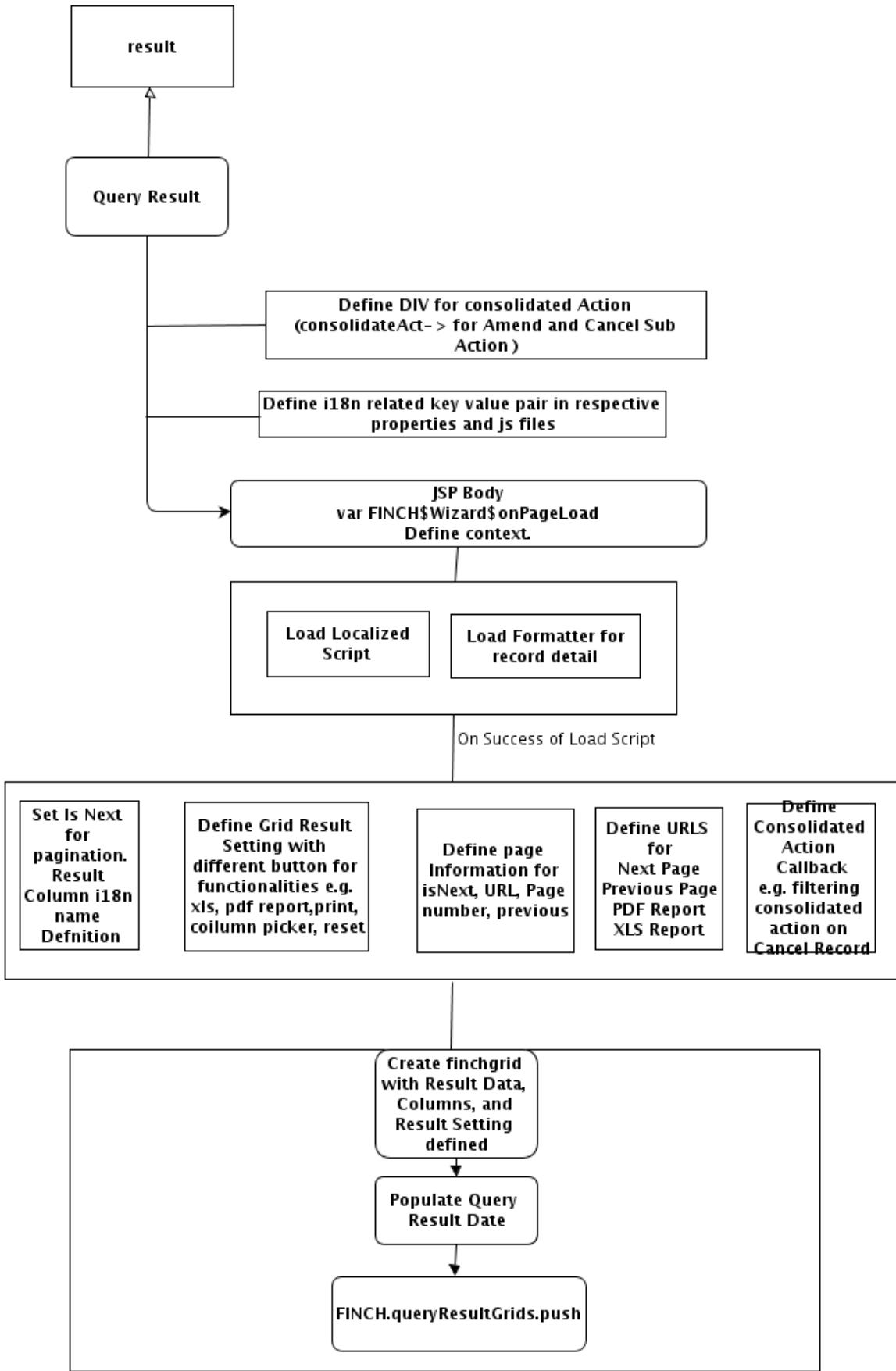


Query:

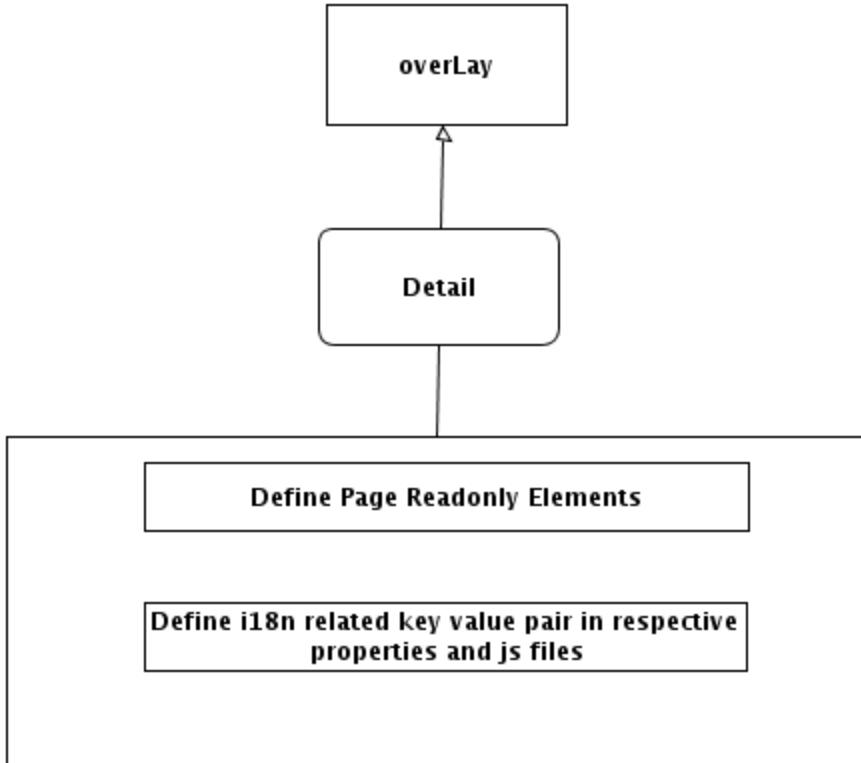
Define query criteria view name and query result view name as defined in views.xml tiles definition file.



Result:



Detail



3.3.4.1 Single Page Entry

1. Scope

In finch application, some entities are created in a single step. The entities like Employee, Holidays etc., which have lesser number of fields associated with them, are ideal candidates for single page entry . Generally only a single table or a table with very few linked child tables are used to store the data for such entities. In order to gather all information related to these entities, the input fields are included in a single page. To create such an entity like, Employee and capture all their required information through UI, the single page based entry model is used. The single page entry screen is followed by the user confirmation and system confirmation screens. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation details for single page entry. It outlines all the necessary steps, both on the client side and server side, to create a new entity using single page based entry model.

2. Prerequisites

Must See

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception](#) for proper handling of exceptions.

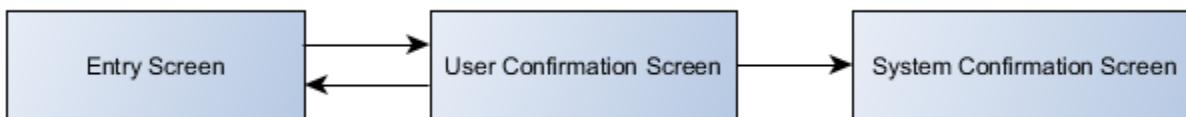
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The screen flow for single page entry is shown below :



Screen	Purpose
Entry	It is used to provide input in the entry form fields for creating a particular entity. It also allows user to submit the entry form for user confirmation.
User Confirmation	It is used to preview and confirm the input data before proceeding further for system confirmation. If the input needs modification then user can go back to the entry screen.
System Confirmation	It confirms that the entry data has been successfully saved in the database. It shows the system confirmation message along with the reference number, wherever applicable, for the completed transaction.

2.3 UI components

The entry page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing entry views, should be retained to get the basic layout, look & feel and functionality of finch entry forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side.

The form input data validations are handled on the client side and the business validations are handled on the server side.

See the [Exception Management](#) document for more details on error handling.

2.7 Menu creation & Action URL patterns

A predefined and uniform URL pattern is essential in a web application.

In finch , all the controller actions are mapped to a unique URL.This URL follows a strict pattern to ensure cleaner code and secure access.

See the [Add new Menu/Action](#) document for more details on URL pattern conventions and menu creation.

Please note that we have used the Employee Entry use case as a reference throughout this document, for all the explanations.

3.New and Modified File List

Layer	Package/Location(sample)	File type/purpose	File name (sample)	New/Modified
Entity	/finch-core/src/main/java/com/nrift/finch/inf/domain/entity	The persistent class for database table	Employee.java	New
Repository	/finch-core/src/main/java/com/nrift/finch/inf/domain/repository	Contains methods for retrieving domain objects	EmployeeRepository.java	New
	/finch-web-infra/src/main/java/com/nrift/finch/inf/domain/repository	Contains all the necessary methods to perform CRUD operation in database	EmployeeUIRepository.java	New
Service	/finch-core/src/main/java/com/nrift/finch/inf/domain/service/	Interface that provides methods related to Employee	EmployeeService.java	New
Service Implementation	/finch-core/src/main/java/com/nrift/finch/inf/domain/service/	Service implementation for Employee Service	EmployeeServiceImpl.java	New
Model	/finch-web-infra/src/main/java/com/nrift/finch/inf/web/employee	Command Form (i.e. POJO) class	EmployeeEntryCommandForm.java	New
View	/finch-web-infra/src/main/web/WEB-INF/views/inf	Module specific xml file to provide view tile definition	views.xml	Modified
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for en locale	finch-i18n_en.js	Modified
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for ja locale	finch-i18n_ja.js	Modified
	/finch-web-infra/src/main/web/WEB-INF/views/inf/user/	jsp for editable entry view	userEntry.jspx	New
	/finch-web-infra/src/main/web/WEB-INF/views/inf/user/	jsp for non-editable user & system confirmation view	userDetail.jspx	New
Controller	/finch-web-infra/src/main/java/com/nrift/finch/inf/web/controller/	Controller class, contains all the action APIs for Entry	EmployeeEntryController.java	New

Some other Helper and Utility classes can be used, but they differ from functionality to functionality.

4. Sequence of Steps

4.1 Configuration

4.1.2 If required add configuration for log in logback.xml file: e.g.

```
Logger                                › Expand
.
.
<configuration>

<property name="finch.log.dir" value="${finch.log.dir:-${java.io.tmpdir}}" />
<property name="finch.log.file.name.INF" value="${finch.log.dir}/finch-inf.log"

.
.

<!-- Standard Rollover Frequencies (for DailyRollingFileAppender) -->
<property name="finch.log.file.MonthlyRolloverFrequency" value="-'yyyy-MM" />

.
.

<logger name="com.nrft.finch.inf" level="INFO">
  <appender-ref ref="finchDlyFileINF"/>
</logger>

.
.

.

```

4.1.3 Add Menu Text in menu.properties for default language English or menu_ja.properties for Japanese language:

```
menu.properties

EMP.name=Employee
EMPEN.name=Entry
EMPQR.name=Query
.
```

```
menu_ja.properties

EMP.name=¥u5F93¥u696D¥u54E1
EMPEN.name=¥u30A8¥u30F3¥u30C8¥u30EA¥u30FC
EMPQR.name=¥u30AF¥u30A8¥u30EA¥u30FC
.
```

4.1.4 Populate Action in INF_ACTION table to display in Menu for Action. e.g.:

INF ACTION

› [Expand](#)

```
Insert into INF_ACTION
(ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,ACTION_URL_PATTERN,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,
CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK)
values ('EMPLEN','/inf/employee/entry/init','Employee
Entry','4','/inf/employee/entry.*',to_date('09-OCT-13','DD-MON-RR'),
to_date('09-OCT-13','DD-MON-RR'),'FINCH-SAMPLE',to_date('09-OCT-13','DD-MON-RR'),'FDBA-71',to_date('10
-OCT-13','DD-MON-RR'),1);
commit;
```

[source](#)

4.2 Create the model

4.2.1 Create Tables and Sequences for the new entity

One needs to create the main and the associate tables for saving the information of the new entity. Information of most of the entities in finch have been stored through multiple tables. The following tables have been used to store information of Employee in finch:

- INF_EMPLOYEE
- INF_EMPLOYEE_GROUP_PCPT
- INF_EMPLOYEE_NAME_XREF
- INF_EMP_APPLN_ROLE_PARTICIPANT

For each and every newly added table, a new Sequence should be added in database. Basically, the Sequence holds the last value of the Primary Key of a table. Now during inserting a new record, it increments the value of the Primary Key and sets it to the new record. A Sequence contains the following information:

- Minimum Value : The minimum value of the PK, generally 1.
- Maximum Value : The maximum value, generally it should be a very large number
- Increment Value : The PK is being incremented by this value. Normally, PK is incremented by 1.
- Last Number : The last value of the PK. The new PK will be generated by adding the Increment Value to this.

4.2.1 CRUD Generation Tool

The CRUD generation tool of finch provides a convenient way to generate several components related to the CRUD (Create, Read, Update & Delete) operations. Components like JPA Entities, Service, Repositories, Controllers, JSPIX can be generated by the user with minimal effort. The tool reads the database schema and generates the classes accordingly. User needs to provide a file in JSON (JavaScript Object Notation) format, containing several information, which the tool reads in order to generate the classes.

Reference

For details of this please see [CRUD Generation Tool](#)

4.2.2 Create Command Form

For sending and receiving the Form/Request values between client and server side, finch uses POJO classes i.e. CommandForms. A CommandForm contains all the field names (JSP field names used in the screen) along with theirgetter() and setter() method. One should be very careful about the data types of the member variables of a CommandForm. If there exists any mismatch between the data types of the variables of CommandForm and the variables used in the screen, it'll lead to an exception.

In finch , a CommandForm for a single page entry should always extend the base class com.nrft.fin.ch.inf.web.FinchAbstractWizardCommandForm. For Employee entry, the EmployeeEntryCommandForm.java has been defined in the package "com.nrft.fin.ch.inf.web.employee".

EmployeeEntryCommandForm

› [Expand](#)

```
public class EmployeeEntryCommandForm extends FinchAbstractWizardCommandForm {
    private static final long serialVersionUID = 1L;
    public static final int GENERAL_TAB = 0;
    public static final int DETAILS_TAB = 1;
    private Employee employee;
    private String applPasswd;
```

[source](#)

```
private String employeeId;
private Date startDate;
private Date endDate;
private String firstName;
private String middleInitial;
private String lastName;
private String title;
private String mail;
private String userId;
private String status;
private String suffix;
private Date employeeOpenDate;
private String employeeOpenedBy;
private Date employeeCloseDate;
private String employeeClosedBy;
private Date lastPasswordChangeDate;
private String locked;
private Branch defaultBranch;
private String defaultBranchName;
private String defaultBranchPk;
private EmployeeDTO employeeDTO;
private List<EmployeeBranchRoleDTO> userAppRoleWithBranchList = new
LinkedList<EmployeeBranchRoleDTO>();
private Map<Long, String> branches;
private Map<Long, String> applicationRoleValues;
private int rowNoForBranchRole = 0;
private String branch;
private String branchPk;
private String role;
private String rolePk;
/**
 * @return the branch
 */
public String getBranch() {
    return this.branch;
}
/**
 * @param branch
 *      the branch to set
 */
public void setBranch(String branch) {
    this.branch = branch;
}
/**
 * @return the branchPk
 */
public String getBranchPk() {
    return this.branchPk;
}
/**
 * @param branchPk
 *      the branchPk to set
 */
public void setBranchPk(String branchPk) {
    this.branchPk = branchPk;
}
/**
 * @return the role
 */
```

```
public String getRole() {
    return this.role;
}
/**
 * @param role
 *      the role to set
 */
public void setRole(String role) {
    this.role = role;
}
/**
 * @return the rolePk
 */
public String getRolePk() {
    return this.rolePk;
}
/**
 * @param rolePk
 *      the rolePk to set
 */
public void setRolePk(String rolePk) {
    this.rolePk = rolePk;
}
/**
 * @return the message
 */
public String getMessage() {
    return this.message;
}
/**
 * @param message
 *      the message to set
 */
public void setMessage(String message) {
    this.message = message;
}
/**
 * @return the rowNoForBranchRole
 */
public int getRowNoForBranchRole() {
    return this.rowNoForBranchRole;
}
/**
 * @param rowNoForOfficeRole
 *      the rowNoForOfficeRole to set
 */
public void setRowNoForBranchRole(int rowNoForOfficeRole) {
    this.rowNoForBranchRole = rowNoForOfficeRole;
}
/**
 * @return the userApIRIWithOfficeList
 */
public List<EmployeeBranchRoleDTO> getUserApIRoleWithBranchList() {
    return this.userApIRoleWithBranchList;
}
/**
 * @param userApIRIWithOfficeList
 *      the userApIRIWithOfficeList to set
 */
```

```

public void setUserApplRoleWithBranchList(
    List<EmployeeBranchRoleDTO> userApplRIWithBranchList) {
    this.userApplRoleWithBranchList = userApplRIWithBranchList;
}
private boolean isInFirstScreen = Boolean.TRUE;
// hidden fields
private String mode = null;
private String employeeMode="ENTRY";

/**
 *
 * @return EmployeeMode
 */
public String getEmployeeMode() {
    return employeeMode;
}
/**
 * Set EmployeeMode
 * @param employeeMode
 */
public void setEmployeeMode(String employeeMode) {
    // this.employeeMode = employeeMode;
}
private String message = "";
public EmployeeEntryCommandForm() {
    Map<FinchWizardMode, String> page0tiles = new HashMap<FinchWizardMode, String>();
    page0tiles.put(EDIT, "userEntry");
    page0tiles.put(USER_CONFIRMATION, "userGeneralDetail");
    page0tiles.put(SYSTEM_CONFIRMATION, "userGeneralDetail");
    FinchWizardPage[] pages = new FinchWizardPage[1];
    Map<FinchWizardMode, FinchWizardButton[]> page0ExBttns = ImmutableMap
        .of(EDIT, new FinchWizardButton[] { NEXT, PREVIOUS });
    pages[0] = new FinchWizardPage("inf.employee.entryaction.label.empgen",
        page0tiles, null, page0ExBttns);
    Map<FinchWizardMode, FinchWizardButton[]> buttons = ImmutableMap.of(
        EDIT,
        new FinchWizardButton[] { PREVIOUS, NEXT, RESET, SUBMIT, TEMPLATESAVE },
        USER_CONFIRMATION, new FinchWizardButton[] { BACK, CONFIRM },
        SYSTEM_CONFIRMATION, new FinchWizardButton[] { OK });
    FinchWizardMode[] modes = new FinchWizardMode[] { EDIT,
        USER_CONFIRMATION, SYSTEM_CONFIRMATION };
    FinchWizard finchWizard = new FinchWizard(
        "inf.employee.entryaction.label.employeeentry", "wizard",
        "inf/employee/entry", modes, pages, buttons);
    setWizard(finchWizard);
}
/**
 * @return the mode
 */
public String getMode() {
    return this.mode;
}
/**
 * @param mode
 *      the mode to set
 */
public void setMode(String mode) {
    this.mode = mode;
}

```

```
/*
 * @return the message
 */
public String getSuccessMessage() {
    return this.message;
}
/**
 * @param message
 *      the message to set
 */
public void setSuccessMessage(String message) {
    this.message = message;
}
@Override
public String getModule() {
    return "INF";
}
/**
 * @return
 */
public boolean isInFirstScreen() {
    return isInFirstScreen;
}
/**
 * @param isInFirstScreen
 */
public void setInFirstScreen(boolean isInFirstScreen) {
    this.isInFirstScreen = isInFirstScreen;
}
/**
 * @return the applPasswd
 */
public String getApplPasswd() {
    return this.applPasswd;
}
/**
 * @return the employeeDTO
 */
public EmployeeDTO getEmployeeDTO() {
    return this.employeeDTO;
}
/**
 * @return the employee
 */
public Employee getEmployee() {
    return this.employee;
}
/**
 * @param employee
 *      the employee to set
 */
public void setEmployee(Employee employee) {
    this.employee = employee;
}
/**
 * @param employeeDTO
 *      the employeeDTO to set
 */
public void setEmployeeDTO(EmployeeDTO employeeDTO) {
```

```
        this.employeeDTO = employeeDTO;
    }
    /**
     * @param applPasswd
     *      the applPasswd to set
     */
    public void setApplPasswd(String applPasswd) {
        this.applPasswd = applPasswd;
    }
    /**
     * @return the employeeCloseDate
     */
    public Date getEmployeeCloseDate() {
        return this.employeeCloseDate;
    }
    /**
     * @param employeeCloseDate
     *      the employeeCloseDate to set
     */
    public void setEmployeeCloseDate(Date employeeCloseDate) {
        this.employeeCloseDate = employeeCloseDate;
    }
    /**
     * @return the employeeClosedBy
     */
    public String getEmployeeClosedBy() {
        return this.employeeClosedBy;
    }
    /**
     * @param employeeClosedBy
     *      the employeeClosedBy to set
     */
    public void setEmployeeClosedBy(String employeeClosedBy) {
        this.employeeClosedBy = employeeClosedBy;
    }
    /**
     * @return the employeedId
     */
    public String getEmployeedId() {
        return this.employeedId;
    }
    /**
     * @param employeedId
     *      the employeedId to set
     */
    public void setEmployeedId(String employeedId) {
        this.employeedId = employeedId;
    }
    /**
     * @return the employeeOpenDate
     */
    public Date getEmployeeOpenDate() {
        return this.employeeOpenDate;
    }
    /**
     * @param employeeOpenDate
     *      the employeeOpenDate to set
     */
    public void setEmployeeOpenDate(Date employeeOpenDate) {
```

```
        this.employeeOpenDate = employeeOpenDate;
    }
    /**
     * @return the employeeOpenedBy
     */
    public String getEmployeeOpenedBy() {
        return this.employeeOpenedBy;
    }
    /**
     * @param employeeOpenedBy
     *      the employeeOpenedBy to set
     */
    public void setEmployeeOpenedBy(String employeeOpenedBy) {
        this.employeeOpenedBy = employeeOpenedBy;
    }
    /**
     * @return the endDate
     */
    public Date getEndDate() {
        return this.endDate;
    }
    /**
     * @param endDate
     *      the endDate to set
     */
    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }
    /**
     * @return the firstName
     */
    public String getFirstName() {
        return this.firstName;
    }
    /**
     * @param firstName
     *      the firstName to set
     */
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    /**
     * @return the lastName
     */
    public String getLastName() {
        return this.lastName;
    }
    /**
     * @param lastName
     *      the lastName to set
     */
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    /**
     * @return the lastPasswordChangeDate
     */
    public Date getLastPasswordChangeDate() {
        return this.lastPasswordChangeDate;
    }
```

```
}

/**
 * @param lastPasswordChangeDate
 *      the lastPasswordChangeDate to set
 */
public void setLastPasswordChangeDate(Date lastPasswordChangeDate) {
    this.lastPasswordChangeDate = lastPasswordChangeDate;
}
/**
 * @return the locked
 */
public String getLocked() {
    return this.locked;
}
/**
 * @param locked
 *      the locked to set
 */
public void setLocked(String locked) {
    this.locked = locked;
}
/**
 * @return the mail
 */
public String getMail() {
    return this.mail;
}
/**
 * @param mail
 *      the mail to set
 */
public void setMail(String mail) {
    this.mail = mail;
}
/**
 * @return the middleInitial
 */
public String getMiddleInitial() {
    return this.middleInitial;
}
/**
 * @param middleInitial
 *      the middleInitial to set
 */
public void setMiddleInitial(String middleInitial) {
    this.middleInitial = middleInitial;
}
/**
 * @return the startDate
 */
public Date getStartDate() {
    return this.startDate;
}
/**
 * @param startDate
 *      the startDate to set
 */
public void setStartDate(Date startDate) {
    this.startDate = startDate;
}
```

```
}

/**
 * @return the status
 */
public String getStatus() {
    return this.status;
}

/**
 * @param status
 *      the status to set
 */
public void setStatus(String status) {
    this.status = status;
}

/**
 * @return the suffix
 */
public String getSuffix() {
    return this.suffix;
}

/**
 * @param suffix
 *      the suffix to set
 */
public void setSuffix(String suffix) {
    this.suffix = suffix;
}

/**
 * @return the title
 */
public String getTitle() {
    return this.title;
}

/**
 * @param title
 *      the title to set
 */
public void setTitle(String title) {
    this.title = title;
}

/**
 * @return the userId
 */
public String getUserId() {
    return this.userId;
}

/**
 * @param userId
 *      the userId to set
 */
public void setUserId(String userId) {
    this.userId = userId;
}

/**
 * @return the defaultBranch
 */
public Branch getDefaultBranch() {
    return this.defaultBranch;
}
```

```

/**
 * @param defaultBranch
 *      the defaultBranch to set
 */
public void setDefaultBranch(Branch defaultBranch) {
    this.defaultBranch = defaultBranch;
}
/**
 * Reset the command Form
 */
public void reset() {
    this.setApplPasswd(StringUtils.EMPTY);
    this.setDefaultBranch(null);
    this.setEmployee(null);
    this.setEmployeeCloseDate(null);
    this.setEmployeeClosedBy(StringUtils.EMPTY);
    this.setEmployeeId(StringUtils.EMPTY);
    this.setEmployeeOpenDate(null);
    this.setEmployeeOpenedBy(StringUtils.EMPTY);
    this.setEndDate(null);
    this.setFirstName(StringUtils.EMPTY);
    this.setLastName(StringUtils.EMPTY);
    this.setLastPasswordChangeDate(null);
    this.setLocked(StringUtils.EMPTY);
    this.setMail(StringUtils.EMPTY);
    this.setEmployeeId(StringUtils.EMPTY);
    this.setMiddleInitial(StringUtils.EMPTY);
    this.setStartDate(null);
    this.setStatus(StringUtils.EMPTY);
    this.setSuffix(StringUtils.EMPTY);
    this.setTitle(StringUtils.EMPTY);
    this.setUserId(StringUtils.EMPTY);
    if (this.getEmployeeDTO() != null) {
        this.getEmployeeDTO().reset();
    }
    this.setMode(StringUtils.EMPTY);
    this.setSuccessMessage(StringUtils.EMPTY);
    this.userApplRoleWithBranchList.clear();
}
/**
 * @return the defaultBranchName
 */
public String getDefaultBranchName() {
    return defaultBranchName;
}
/**
 * @param defaultBranchName
 *      the defaultBranchName to set
 */
public void setDefaultBranchName(String defaultBranchName) {
    this.defaultBranchName = defaultBranchName;
}
/**
 * @return the defaultBranchPk
 */
public String getDefaultBranchPk() {
    return defaultBranchPk;
}
/**

```

```
* @param defaultBranchPk
*      the defaultBranchPk to set
*/
public void setDefaultBranchPk(String defaultBranchPk) {
    this.defaultBranchPk = defaultBranchPk;
}
/**
* @return the branches
*/
public Map<Long, String> getBranches() {
    return branches;
}
/**
* @param branches
*      the branches to set
*/
public void setBranches(Map<Long, String> branches) {
    this.branches = branches;
}
/**
* @return the applicationRoleValues
*/
public Map<Long, String> getApplicationRoleValues() {
    return applicationRoleValues;
}
/**
* @param applicationRoleValues
*      the applicationRoleValues to set
*/
public void setApplicationRoleValues(Map<Long, String> applicationRoleValues) {
```

```
        this.applicationRoleValues = applicationRoleValues;  
    }  
}
```

4.3 Create the view

4.3.1 Usage of Tiles

In order to create the layout for any single page entry, developer needs to provide a tile definition of its corresponding view in views.xml.

For employee entry, user needs two views :

- Editable employee entry page
- Non-editable user confirmation & system confirmation pages

The view corresponding to employee entry say "userEntry.jspx" needs to be defined in views.xml as shown below :

View Definition for Employee Entry

```
<!-- User entry page -->  
<definition name="userEntry" extends="wizard-page">  
    <put-attribute name="wizard-page" value="/WEB-INF/views/inf/user/userEntry.jspx"/>  
</definition>
```

The new view "userEntry.jspx" needs to extend "wizard-page". By doing so, the new view inherits all the common functionality and markup from "wizardPage.jspx" & "wizardContainer.jspx".

Employee entry page inherits the following features/elements from its wizard template :

Form tag and all its attributes

Header content

Form action area containing all the buttons

The view corresponding to employee details say "userDetail.jspx" needs to be defined in views.xml as shown below :

View definition for Employee Details

```
<!-- User General detail page -->  
<definition name="userGeneralDetail" extends="wizard-page">  
    <put-attribute name="wizard-page"  
        value="/WEB-INF/views/inf/user/userDetail.jspx" />  
</definition>
```

See the [View templates - Screen Composition](#) section to get further details.

4.3.2 Basic components of single page entry screen

Entry page components
Entry User Confirmation Page
Entry System Confirmation Page

EMPEN:Employee Entry [20140303]

Action	Branch Id	Application Role
	FBR	ROLE1

Actions

Branch Id

Application Role

User ID: 1

First Name: 2

Default Branch: 3

Title: 4

Start Date (UTC): 20150201

End Date (I):

Suffix: 5

E-Mail: 6

Middle Initial:

Last Name:

Password: 7

Confirm Pa:

SAVE TEMPLATE: 5

1. Label - See [Form Elements](#) - Label indicating mandatory fields section for further details.
2. Text input control - See [Form Elements](#) - Text input control section for further details.
3. Drop-down input control - See [Form Elements](#) - Drop-down input control section for further details.
4. Date Picker control See [Form Elements](#) Date Picker input control further details.
5. Grid section - See [Grid Component](#) for further details.
6. Action area of entry form - See the Initialize new entry page section below for details.
7. Save template - See [Saved Template](#) and [Saved Template](#) section for more details

Search Menu 

EMPUC:Employee Entry User Confirmation [20140303]

User ID	dummyemp	First Name	dummyEmp	Middle Initial
Last Name		Default Branch	FBR	Title
Start Date (UTC)	20150201	End Date (UTC)	20150228	Suffix
E-Mail		Employee Opened By	finch	

Application Roles

Branch Id	Application Role
FBR	ROLE1

1

2

1. Employee detail block - See [Detail Page Layout](#) for further details.
2. Action area of user confirmation page - See [Create user confirmation page](#) for new entry section for further details.

EMPSC:Employee Entry System Confirmation [20140303]

User ID	dummyemp	First Name	dummyEmp	Middle Initial
Last Name		Default Branch	FBR	Title
Start Date (UTC)	20150201	End Date (UTC)	20150228	Suffix
E-Mail		Employee Opened By	finch	

Application Roles

Branch Id	Application Role
FBR	ROLE1

1. System confirmation message - See Create system confirmation page for new entry for details.
2. Employee detail block - See [Detail Page Layout](#) for further details.
3. Action area of system confirmation page - See Create system confirmation page for new entry for details.

4.3.3 Create new entry page

After providing the tile definition, the developer needs to create the new entry page "userEntry.jspx".

The "userEntry.jspx" needs to be placed in a specific folder. Its location will depend on the type of data it collects.

In general, each entity in finch application is maintained in a different finch module depending on its type. Whatever be the type of data, all jspx files are kept within views folder. Within each views folder, a folder with the same name as the <module-name> is created.

So the general path for the new jspx should be :

```
devel/finch-<project-name>/src/main/web/WEB-INF/views/inf/<module-name>/new.jspx
```

The path for "userEntry.jspx" will be :

```
devel/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx
```

In the "userEntry.jspx" developer needs to create the form elements using specific classes and ids.

See the [Entry/Amend Form](#) section to get the details about the markup needed to create an entry page.

Now that the "userEntry.jspx" markup is ready, developer needs to map the form attributes and element values to a data model. In finch application commandForms have been used for sending the form values from client to the server.

The variable names used in the commandForms should be identical to the names used in the form elements and attributes.

For example :For branch, the variable defined in EmployeeEntryCommandForm class is "branch".

In "userEntry.jspx", the branch select element options should be mapped to this variable using "\${commandForm.branches}" as shown below :

```
/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx > Expand  
source  
<div class="formItem">  
    <form:label path="commandForm.employeeDTO.defaultBranch" class="required">  
        <spring:message code="inf.employeeentryaction.label.defaultbranch" htmlEscape="false" />  
    </form:label>  
    <form:select id="defaultBranch" path="commandForm.employeeDTO.defaultBranchPk"  
    class="dropdowninput" defaultValue="defaultBranch">  
        <form:option value="" /></form:option>  
        <form:options items="${commandForm.branches}" />  
    </form:select>  
    <div class="clear">  
        <spring:message text="" htmlEscape="false" />  
    </div>  
</div>
```

See the section on Create Command Form and Create the Controller for further details.

commandForm.uniqueId
commandForm.uniqueId is an unique id assigned to each instance of an entry page. This unique id is required for multi-tab control

4.3.4 Initialize new entry page

The new entry page has a life cycle with the following phases:

- data
- create
- init
- destroy

See [UI Core](#) for more details on finch plugin structure and related methods.

The new entry page is editable. It allows users to

- Provide inputs
- Submit the form using Submit button
- Reset the form using Reset button

The list of hooks, defined in finch-wizard.js, which is inherited from the wizard template to support the above mentioned actions are :

- reset - It resets the form element values to the original state
- submit - It submits the form and on successful submission navigates to user confirmation page

The above mentioned functionality are available to every new entry page from its template.

However, in the individual entry page developer must implement FINCH\$Wizard\$OnPageLoad function for initializing the page with the required page specific logic :

```
/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx > Expand  
var FINCH$Wizard$OnPageLoad = function() {  
    $employeeEntry$context = $('#commandForm');  
    ..  
    //initializing logic goes here  
}
```

[source](#)

Within the FINCH\$Wizard\$OnPageLoad function, developer can do the following :

1. Load the locale specific finch-i18n js file for supporting i18n. For example, in employee entry, the page specific key value pairs need to be included in the module specific finch i18n files -finch-i18n_en.js and finch-i18n_ja.js. These locale specific js files are loaded in "userEntry.jspx" as shown below :

```
/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx  
FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/inf/finch-i18n.js', async: false}]);
```

Reference

Refer to [Multilingual i18n Support - i18n implementation for client-side validation messages](#) section & [UI Core](#) for more details on i18n and FINCH.loadLocalizedScript method.

2. Load the finch-util-validator.js for using the utility functions in validation.

```
/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx  
FINCH.loadScript([  
    {path: FINCH.context.path + '/scripts/inf/finch-util-validator.js', async: false}  
])
```

3. Define one or more hooks. There are some predefined actions like 'reset' and 'submit'. These hooks perform some entry page-specific logic like input data validations before the predefined actions.

```
/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx > Expand  
function validateSubmit() {  
    FINCH.clearNotice();  
    var validationMessages = [];  
    var messageFieldMap = new Object();  
    var emailPattern = /^[a-zA-Z0-9_\.]+@[([a-zA-Z0-9\-\_\.])+]+([a-zA-Z0-9]{2,4})+\$/;  
    var defaultBranch = $('#defaultBranch option:selected', $employeeEntry$context);  
}
```

[source](#)

```

var userId = $('#userId', $employeeEntry$context).val();
var startDateValidation = true;
var endDateValidation = true;
var isValidate = true;
var validPattern = /^[0-9a-zA-Z_]*$/;
userId=jQuery.trim(userId);
if (typeof FINCH.RegEx.userId !== 'undefined'){
    validPattern = FINCH.RegEx.userId;
}
$('#userId', $employeeEntry$context).val(userId);
if((userId == "") ) {
    messageFieldMap['employeeDTO.userId'] = FINCH.i18n.employee.generalinfo.user_id;
} else {
    if((userId.length <![CDATA[<]]> 5) ||(userId.length>20)|| !validPattern.test(userId)) {
        messageFieldMap['employeeDTO.userId'] = FINCH.i18n.employee.generalinfo.invalid_userId;
    } else {
        $('#userId', $employeeEntry$context).val(userId.toLowerCase());
    }
}
var firstName = $('#firstName', $employeeEntry$context).val();
firstName = jQuery.trim(firstName);
$('#firstName', $employeeEntry$context).val(firstName);
if(firstName == "") {
    messageFieldMap['employeeDTO.firstName'] = FINCH.i18n.employee.generalinfo.first_name;
} else {
    if((firstName.length>35)||!VALIDATOR.checkValidName(firstName)) {
        messageFieldMap['employeeDTO.firstName'] =
FINCH.i18n.employee.generalinfo.invalid_first_name;
    }
}
var middleInitial = $('#middleInitial', $employeeEntry$context).val();
middleInitial=jQuery.trim(middleInitial);
$('#middleInitial', $employeeEntry$context).val(middleInitial);
if(middleInitial != "") {
    if((middleInitial.length>4)||!VALIDATOR.checkValidName(middleInitial)) {
        messageFieldMap['employeeDTO.middleInitial'] =
FINCH.i18n.employee.generalinfo.middle_initial;
    }
}
var lastName = $('#lastName', $employeeEntry$context).val();
lastName=jQuery.trim(lastName);
$('#lastName', $employeeEntry$context).val(lastName);
if(lastName != "") {
    if((lastName.length>35)||!VALIDATOR.checkValidName(lastName)) {
        messageFieldMap['employeeDTO.lastName'] =
FINCH.i18n.employee.generalinfo.invalid_last_name;
    }
}
if(defaultBranch.text() =="" ) {
    messageFieldMap['employeeDTO.defaultBranchPk'] =
FINCH.i18n.employee.generalinfo.default_branch;
} else if(!defaultBranch.val()) {
    messageFieldMap['employeeDTO.defaultBranchPk'] = '['+ defaultBranch.text() +']' +
FINCH.i18n.employee.generalinfo.default_branch_nondefault_value;
}
var title = $('#title', $employeeEntry$context).val();
title=jQuery.trim(title);
$('#title', $employeeEntry$context).val(title);
if(title != "") {

```

```

        if((title.length>5)||VALIDATOR.checkValidName(title)) {
            messageFieldMap['employeeDTO.title'] = FINCH.i18n.employee.generalinfo.title;
        }
    }
    var startDate = $('#startDate', $employeeEntry$context).val();
    startDate = $.trim(startDate);
    $('#startDate', $employeeEntry$context).val(startDate);

    if(VALIDATOR.isNullValue(startDate)) {
        messageFieldMap['employeeDTO.startDateStr'] = FINCH.i18n.employee.generalinfo.start_date;
    } else {
        if(!isValidDate(startDate)) {
            startDateValidation = false;
        }
    }

    var endDate = $('#endDate', $employeeEntry$context).val();
    endDate = $.trim(endDate);
    $('#endDate', $employeeEntry$context).val(endDate);

    if(!VALIDATOR.isNullValue(endDate)) {
        if(!isValidDate($.trim(endDate))) {
            endDateValidation = false;
        }
    }
    var compDate = compareDateStr(startDate,endDate);
    if(compDate == -1) {
        messageFieldMap['employeeDTO.startDateStr'] =
FINCH.i18n.employee.generalinfo.start_date_greater_than_open_date;
    }
    var suffix= $('#suffix', $employeeEntry$context).val();
    suffix=jQuery.trim(suffix);
    $('#suffix', $employeeEntry$context).val(suffix);
    if(suffix != "") {
        if((suffix.length>5)||VALIDATOR.checkValidName(suffix)) {
            messageFieldMap['employeeDTO.suffix'] = FINCH.i18n.employee.generalinfo.suffix;
        }
    }
    var email= $('#email', $employeeEntry$context).val();
    email=jQuery.trim(email);
    $('#email', $employeeEntry$context).val(email);
    if(email != "") {
        if((email.length>60)||!emailPattern.test(email)) {
            messageFieldMap['employeeDTO.mail'] = FINCH.i18n.employee.generalinfo.email;
        }
    }
    var password = $('#password', $employeeEntry$context).val();
    password=jQuery.trim(password);
    $('#password', $employeeEntry$context).val(password);
    var confirmPassword = $('#confirmPassword', $employeeEntry$context).val();
    confirmPassword=jQuery.trim(confirmPassword);
    $('#confirmPassword', $employeeEntry$context).val(confirmPassword);
    if (password == "") {
        messageFieldMap['employeeDTO.password'] =
FINCH.i18n.employee.generalinfo.empty_password;
    }
    if (confirmPassword == "") {
        messageFieldMap['employeeDTO.confirmPassword'] =
FINCH.i18n.employee.generalinfo.empty_confirm_password;
    }

```

```
        }
        if (password != "" && confirmPassword != "" && password != confirmPassword) {
            messageFieldMap['employeeDTO.confirmPassword'] =
                FINCH.i18n.employee.generalinfo.invalid_password;
        }
        var theGridObject = $userApplRoleGrid.data('theUserApplRoleGrid');
        var theGridData = theGridObject.getData();
        if (theGridData.length == 0) {
            messageFieldMap['employeeDTO.applicationRole'] =
                FINCH.i18n.employee.generalinfo.empty_app_role;
        }
        if ($.isEmptyObject(messageFieldMap) !== true){
            validationMessages.push(messageFieldMap);
            FINCH.postNotice(FINCH.notice.type.error, validationMessages, true);
            isValidate = false;
        }
        else {
            $('.formHeader').find('.formTabErrorIco').css('display', 'none');
            isValidate = true;
        }
    }
} else {
    validationMessages.push(messageFieldMap);
    FINCH.postNotice(FINCH.notice.type.error, validationMessages, true);
    isValidate = false;
}
});
```

```

    }
    return (isValid <![CDATA[&&]]> startDateValidation <![CDATA[&&]]> endDateValidation);
}

```

We can see from the code snippet above that for employee entry, a validateHook function is defined which validates the user input , pushes the corresponding error messages into the messageFieldMap. It performs the mandatory field validations and matches password & confirm password input values.

4. Display the error messages

```

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx   > Expand
if ($.isEmptyObject(messageFieldMap) !== true){
    validationMessages.push(messageFieldMap);
    FINCH.postNotice(FINCH.notice.type.error, validationMessages, true);
    isValid = false;
}
else {
    $('.formHeader').find('.formTabErrorIco').css('display', 'none');
    isValid = true;
}

```

[source](#)

In order to display the error messages, call the FINCH.postNotice method and pass the following parameters : error type - FINCH.notice.type.error, error messages - validationMessages array, persistence value - true

5. Register the hook.

```

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx   > Expand
$finch$wizard.register('previous',validateHook);
$finch$wizard.register('next',validateHook);
$finch$wizard.register('submit',validateHook);
$finch$wizard.register('unload',unloadHook);

```

[source](#)

Here we register a hook for "submit" action, that means validateHook function will be called before the entry form is submitted.

6. Deregister the registered hooks.

```

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx   > Expand
var unloadHook = function(){
    $finch$wizard.deregister('previous',validateHook);
    $finch$wizard.deregister('next',validateHook);
    $finch$wizard.deregister('submit',validateHook);
    $finch$wizard.deregister('unload',unloadHook);
};

```

[source](#)

SlickGrid Component

If a entry page uses SlickGrid, then in "unload" hook developer has to destroy the grid by calling destroy() of SlickGrid component, otherwise in IE SlickGrid may not work properly.

7. Provide a common request handler which can be used for all server communications required for this entry page. The request handler will have page specific configuration values.

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx [Expand](#) [source](#)

```
Employee$Handler$RequestHandler.generic(undefined, { requestUri: requestUrl,
  settings: {data : populateBranchRequestParam()},
  onJsonContent : function(e, options, $target, content) {
    if(content.success == true){
      if(content.value[0] !== ""){
        $('#role').removeAttr('disabled');
      }
      populateRoleForBranch(content.value[0]);
    } else {
      FINCH.postNotice(FINCH.notice.type.error, content.value, true);
    }
  }
});
```

8. Provide the handlers for actions specific to that page

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx [Expand](#) [source](#)

```
function addBranchRoleHandler(e) {
  var validation=validationForBranchRole();
  var localvalidation=localValidationForBranchRole();
  if(validation) {
    if(localvalidation) {
      var requestUrl = $('form').attr('action') + "/addBranchRole.json?commandFormId=" +
      $('[name=commandFormId]').val();
      Employee$Handler$RequestHandler.generic(e, { requestUri: requestUrl,
        settings: {data : populateBranchRoleRequestParam()},
        onJsonContent : function(e, options, $target, content) {
          if(content.success == true){
            updateBranchRoleRequest(content.value[0]);
          } else {
            FINCH.postNotice(FINCH.notice.type.error, content.value, true);
          }
        }
      });
      $('#role').val("");
      /*
      Changing the value of dropdown using jQuery val() function
      does not cause the change event to be fired. So the tooltip
      has to be changed manually.
      */
      $('#role').attr('title', $('#role option:first-child').attr('title'));
    }
  }
}
//Binds the handler to the addEmpAppRoleBtn
$('#addBranchRoleBtn').bind('click', addBranchRoleHandler);
```

9. If a grid is required in the entry page, provide its configuration, column definition and then initialize it.

```

var userApplRoleGridConf = {
    editMode      : 'both',
    editCallback   : editBranchRoleHandler,
    deleteCallback : removeUserApplRoleMappingHandler
};
var userApplRoleColumns = [
    {name:<spring:message code="inf.employee.label.branchid" htmlEscape="false" />, field:"branch",
    id:"branch"}, 
    {name:<spring:message code="inf.employee.label.applicationsrole" htmlEscape="false" />, field:"role",
    id:"role"}
];
$UserApplRoleGrid = $('#userApplRoleGrid', $employeeEntry$context);
var $UserApplRoleGridObject = $UserApplRoleGrid.fincheditablegrid(userApplRoleData,
userApplRoleColumns, userApplRoleGridConf);

```

10. Populate the grid.

/finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx → [Expand](#)

[source](#)

```

<c:forEach items="${commandForm.userApplRoleWithBranchList}" var="dl">
    item = {};
    rowId  += 1;
    item.id  = "finch_" + rowId;
    item.branch  = "<c:out value='${dl.branch}' />";
    item.branchPk  = "<c:out value='${dl.branchPk}' />";
    item.role  = "<c:out value='${dl.role}' />";
    item.rolePk  = "<c:out value='${dl.rolePk}' />";
    userApplRoleData.push(item);
</c:forEach>

```

See [Grid Component](#) document for more details on using the finch grid component .

4.3.5 Create user confirmation page for new entry

As mentioned above after successful submission of the entry form, user navigates to the user confirmation page.

So next the developer needs to create the markup for the entry details page .

For example, for employee entry, developer needs to create "userDetail.jspx".

The markup required to create the non-editable user confirmation page is different from the editable employee entry page.

See the section on [Detail Page Layout](#) to get the details about the markup needed to create the details page.

The user confirmation page is non-editable. It allows user to

- Go back to the entry page using Back button
- Submit the form values for persistence using Confirm button

The list of hooks, defined in finch-wizard.js, which is inherited from the wizard template to support the above mentioned actions are :

- back - It allows the user to go back to the editable mode of entry page and change the provided input values.
- confirm - It submits the form for permanently saving the values in database. On successful submission user navigates to the system

confirmation page.

In the user confirmation page if any page specific logic is required then developer must include it within FINCH\$Wizard\$OnPageLoad function.

Apart from the above mentioned page specific requirement, the rest of the actions in the employee entry user confirmation page is derived from the wizard template. It has the following actions :

Back button - The user navigates back to the employee entry page to change the input values

Confirm button - The user submits the employee entry details and the data is successfully entered in the database. User navigates to the employee entry system confirmation page.

4.3.6 Create system confirmation page for new entry

After successful submission of the user confirmation page, user navigates to the system confirmation page.

The markup created for user confirmation page is used for system confirmation page as well.

For example, for employee entry, "employeeDetail.jspx" is used for both user confirmation & system confirmation pages.

The system confirmation page is non-editable. It only allows user to

- Go back to the fresh entry page using the Ok button

The hook, defined in finch-wizard.js, which is inherited from the wizard template to support the above mentioned action is :

- ok- It allows the user to go back to the fresh entry page and start a new wizard life cycle

For example,

In the employee entry flow, after successful form submission, user navigates to the system confirmation page where the confirmation message is shown.

4.4 Create the controller

4.3.1 Follow Inheritance rule

In finch, a Controller should extend base class com.nrft.finч.inf.web.controller.FinchAbstractEditWizardController.java. The annotation '@Controller' is being used to consider a simple class as a Controller.

Create a logger object in the controller. See proper usage of logger in [Logging](#).

The following Controller has been added for Employee entry:

EmployeeEntryController.java Expand

public class EmployeeEntryController extends FinchAbstractEditWizardController<EmployeeEntryCommandForm> implements MessageSourceAware {
 private static final Logger log = LoggerFactory.getLogger(EmployeeEntryController.class);
 ...
}

[source](#)

Inside the controller, developer has to implement the following methods to complete the entry operation:

EmployeeEntryController.java

› [Expand](#)

```
//This method will contain screen specific initialization logic.
doInit(C commandForm) source

//This method will contain validation logic.
doValidate(C commandForm, int index)

//This method will contain page reinitialize logic.
doReset(C commandForm, int index)

//This method provide a hook to implement logic(s) , that will be applicable during the navigation from entry
page to user confirmation page.
doSubmit(C commandForm, int index)

//This method will contain the logic related to data persistence and message generation.
doConfirm(C commandForm)
```

4.4.2 Provide request URL mapping

The developer should specify the request mapping at the class level. A URL mapping should be specified for each and every controller in Finch. The annotation '@RequestMapping' is being used to set the request URL. The URL should follow the Action URL Pattern specified (in database) for the Menu item clicked. By using the annotation @SessionAttribute the CommandForm has been specified as a session attribute. The following mapping has been used for Employee entry:

EmployeeEntryController.java

› [Expand](#)

```
@Controller
@SessionAttributes("commandForm")
@RequestMapping(value = "/inf/employee/entry/**")
public class EmployeeEntryController extends
    FinchAbstractEditWizardController<EmployeeEntryCommandForm> implements
    MessageSourceAware {
    ..
}
```

4.4.3 Define screen specific initialization logic

The base class FinchAbstractEditWizardController.java has a no-op implementation of the method doInit(), so that the sub-classes can override it as per requirement. Different controllers provide functionality specific initialization logic in this method. Basically, the initialization should include:

- Initializing the member variables of CommandForm
- Setting some default values to the CommandFrom/Screen
- Populating the possible values for the drop downs in the screen. Most of the time the values are being populated using the Constraints. See [Constraint List of Values](#)

EmployeeEntryController

› Expand

```
@Override
protected void doInit(EmployeeEntryCommandForm commandForm)
    throws FinchException {
try {
    log.debug("Initialising Employee Entry Command Form ...");
    if (!commandForm.isSaveTemplate()) {
        // CommandForm Reset
        commandForm.reset();
    }
    // commandForm.reset();
    commandForm.setInFirstScreen(Boolean.TRUE);
    List<Branch> branches = getService(BranchService.class)
        .getBranchForEnterprise(
            Operation.getInstance().getContext()
                .getEnterpriseId());
    Map<Long, String> branchMap = new LinkedHashMap<Long, String>();
    for (Branch br : branches) {
        branchMap.put(br.getBranchPk(), br.getBranchId());
    }
    commandForm.setBranches(branchMap);
    if (commandForm.getApplicationRoleValues() == null) {
        List<ApplicationRole> allRoles = getService(
            ApplicationRoleService.class).getAllRoles();
        Map<Long, String> applicationRoleValues = new LinkedHashMap<Long, String>();
        for (ApplicationRole role : allRoles) {
            applicationRoleValues.put(role.getApplRolePk(),
                role.getApplicationRoleName());
        }
        commandForm.setApplicationRoleValues(applicationRoleValues);
    }
    if (ConfigurationUtils.isApplicationDateSupported()) {
        ApplicationDateService applicationDateService = getService(ApplicationDateService.class);
        commandForm.setEmployeeOpenDate(applicationDateService
            .getApplicationDate(getComponent()));
    } else {
        commandForm.setEmployeeOpenDate(DbUtils.getSystemDate());
    }
    if (!commandForm.isSaveTemplate()) {
        commandForm.setEmployeeDTO(new EmployeeDTO());
    }
    commandForm.getEmployeeDTO().setEmployeeOpenDate(
        commandForm.getEmployeeOpenDate());
    commandForm.setMode(Constants.ENTRY);
} catch (Exception e) {
    log.error("Failed to make a view for the Employee data", e);
    throw new FinchException(
        "inf.employee.entry.action.error.populate", e);
}
}
```

source

4.4.4 Define server side validation logic

When user submits an entry form after specifying values to the fields, the form needs to be validated first. This validation is being done in `d`
`oValidate()` method of a controller. The base class `FinchAbstractWizardController.java` has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality. The developer only has to implement the method in the

controller, he doesn't need to call it explicitly. This method has been called implicitly by the submit() method of the base class, which is being called by submitting the entry form.

In finch, in case of validation failure, module specific Keyed Exception is thrown. The developer has to make sure, that the key used in the Keyed Exception has a proper message in ErrorResources.properties of that module.

During validation, normally one should check:

- Whether all the mandatory fields have been filled up or not
- Whether the specified values are valid or not (like in case of dates, static data etc.)
- In finch, few text fields or text areas allow limited number of character sets. For details of this implementation please see [Character Validation](#)

EmployeeEntryController.java » Expand

source

```
@Override
protected void doValidate(EmployeeEntryCommandForm commandForm, int index)
    throws FinchException {
    // Marking that the Wizard is no longer in the First Screen
    commandForm.setInFirstScreen(false);
    // Validating the current Wizard page before Navigation
    switch (index) {
        case EmployeeEntryCommandForm.GENERAL_TAB:
            validatePage1(commandForm);
            break;
        default:
            break;
    }
}
```

FinchAbstractEditWizardController.java

› Expand

```
@RequestMapping(value = "/submit", method = RequestMethod.POST) source
public String submit(ModelMap map, @ModelAttribute("commandForm") C commandForm,
    @RequestParam(value = "validate", required = false) boolean validate) {
    FinchModelMap modelMap = new FinchModelMap(map);
    modelMap.putToModelMap("commandForm", commandForm);
    String module = commandForm.getModule();
    FinchWizard wizard = commandForm.getWizard();
    FinchWizardMode mode = wizard.currentMode();
    commandForm.setValidateThisPage(null);
    boolean error = false;
    try {
        preSubmit(modelMap, commandForm, validate);
        validate = (commandForm.getValidateThisPage() != null) ? commandForm.getValidateThisPage() :
validate;
        if (validate) {
            // validate
            doValidate(commandForm, wizard.getPageIndex());
        }
        doSubmit(commandForm, wizard.getPageIndex());
        // switch to next mode
        if (!wizard.isOnLastMode()) {
            if (commandForm.isSkipNextMode())
                wizard = skipNextWizardMode(commandForm);
            mode = wizard.nextMode();
        }
        setScreenName(commandForm);
        // Access Log
        logUserAccessInfo(getScreenId(commandForm));
    } catch (Exception e) {
        error = true;
        log.error("Failed to submit wizard page", e);
        modelMap.addError(e, Component.getComponent(module).get(), getDefaultErrorKey(),
            ArrayUtils.EMPTY_STRING_ARRAY);
    }
    FinchWizardPage pageN = error ? wizard.currentPage() : wizard.nthPage(0);
    return pageN.getTitle(mode);
}
```

4.4.5 Define form submit and user confirmation logic

The submission of an entry form is being handled by the method `submit()` of the base class `FinchAbstractWizardController.java`. This method includes:

- Validation of the values specified in the entry form.
- (If required) Population of few values according to the functionality from the specified values in the entry form.
- Finally, returning the tile name of the JSPX for User Confirmation screen.

As mentioned earlier, for the validation a developer should implement the `doValidate()` in his controller. This method will be implicitly called by the method `submit()` of the base class. Functionality specific activities should be done by the developer by implementing the `doSubmit()` in the controller. The returning of the tile name of the JSPX for User Confirmation is being handled by the base class.

4.4.6 Define system confirmation id generation and data persistence logic

The confirmation of an entry page is being handled by the method `confirm()` of the base class `FinchAbstractEditWizardController.java`. This method includes:

- Persist all the data into the database.
- Finally, returning the tile name of the JSPX for System Confirmation screen.

The implementation of persisting data, handling XML messages differ from controller to controller as per functionality. So the base class has a no-op doConfirm() method, by implementing which the developer completes all these activities in the controller. The returning of the tile name of the JSPX for System Confirmation is being handled by the base class.

FinchAbstractEditWizardController.java
source

```

@RequestMapping(value = "/confirm", method = RequestMethod.POST)
public String confirm(ModelMap map, @ModelAttribute("commandForm") C commandForm) {
    FinchModelMap modelMap = new FinchModelMap(map);
    modelMap.putToModelMap("commandForm", commandForm);
    String module = commandForm.getModule();
    FinchWizard wizard = commandForm.getWizard();
    FinchWizardMode mode = wizard.currentMode();
    boolean error = false;
    try {
        doConfirm(commandForm);
        // switch to next mode
        if (!wizard.isOnLastMode()) {
            if(commandForm.isSkipNextMode())
                wizard = skipNextWizardMode(commandForm);
            mode = wizard.nextMode();
        }
        setScreenName(commandForm);
        // Access Log
        logUserAccessInfo(getScreenId(commandForm));
    } catch (Exception e) {
        error = true;e.printStackTrace();
        log.error("Failed to save wizard pages", e);
        modelMap.addError(e, Component.getComponent(module).get(), getDefaultErrorKey(),
            ArrayUtils.EMPTY_STRING_ARRAY);
    }
    FinchWizardPage pageN = error ? wizard.currentPage() : wizard.nthPage(0);
    return pageN.getTile(mode);
}

```

4.4.7 Define page navigation logic

The implementation of the abstract method getPages() in the controller, provides the tile names of the JSPX pages used for the entry operation. For single page entry operation, this method returns only one page which is being set to the CommandForm by the base class. Now, from submit() and commit() method of the base class, the page is being extracted from the CommandForm for returning the tile name of the redirected screen.

4.4.8 Implement Abstract methods

The abstract methods getModule() and getDefaultErrorKey() should be implemented as following:

4.4.9 Transaction Management

Transaction has been properly managed by the base class. The developer has nothing to do with this. For details of Transaction Management see [Transaction Management](#).

4.3.10 Access Log

In finch, whenever a Single Page Entry screen is being added, the Access Log implementation should be done properly. From this Access Log, we can get history of accessing of the screen.

For Access Log of a Single Page Entry, the developer has to only use the standard URL pattern for entry operation. The rest has been handled by the framework. Standard URL pattern for an operation is like "secure/<module_in_small>/<entity_in_small>/<operation_in_sm all>/*", where ref, trd, stl are the modules; account, customer, trade, settlement are the entities; entry, amend, query are the operations.

For example, the URL for Market Price entry should be like 'secure/ref/marketprice/entry'.

If any URL does not follow the standard pattern, then one has to configure that pattern in database tables for the Access Log implementation. For details of this setting see [Access Log](#).

4.5 Create the Service

In finch Controller/Bach call Service method to perform any particular actions like Retrieves the Employee object against a given employee id or Cancel given employee.Controller takes care of user request and call proper service to do the user operation

```
com.nrft.finch.inf.domain.service.EmployeeService › Expand
source

..
public interface EmployeeService {

    /**
     * Retrieves the Employee object against a given employee id
     *
     * @param employeeId String
     * @return Employee
     * @throws FinchException throws exception in case of failure to retrieve the Employee
     */
    Employee getEmployeeByUserId(String userId) throws FinchException;
    /**
     * Retrieves the Employee object against a given employee id and status
     *
     * @param employeeId String
     * @param status String
     * status of the employee.Status can be NORMAL, CANCEL_ALL.
     * @return Employee
     * @throws FinchException throws exception in case of failure to retrieve the Employee
     */
    Employee getEmployeeByUserIdAndStatus(String userId, String status) throws FinchException;

    /**
     * Retrieves the list of {@link Employee}s by a given enterprise id whos status is normal
     *
     * @param enterpriseId must be not null.
     * @return list of users/employees in a particular enterprise.
     */
    List<Employee> getEmployees(String enterpriseId);

    /**
     * Retrieves the list of {@link Employee}s by a given enterprise id whos status is normal
     *
     * @param enterpriseId must be not null.
     * @return list of users/employees in a particular enterprise.
     */
    List<Employee> getAllEmployees(String enterpriseId);

    /**
     * Retrieves the Employee object against a given employee PK
     *
     * @param employeePk
     * @return Employee
     * @throws FinchException throws exception in case of failure to retrieve the Employee
     */
    public Employee findEmployeeByPk(long employeePk) throws FinchException;
```

```
/**  
 * Saves a new Employee into the database <br>  
 * <p>  
 * <b>Responsibility</b>  
 * <li>Database Persistence</li>  
 */  
public Employee enterEmployee(EmployeeDTO empDto) throws DataInputException;  
  
/**  
 * Populates the EmployeeAmendCommandForm from the given employeePk  
 */  
public EmployeeDTO populateEmployeeDTO(long employeePk)  
    throws FinchException;  
  
/**  
 * Updates a Employee Entity in the database by comparing the Employee  
 * Information from the original EmployeeDTO and the latest information  
 * supplied inside the updated EmployeeDTO <br>  
 * <p>  
 * <b>Responsibility</b>  
 * <li>Database Persistence</li>  
 */  
public void updateEmployee(long originalEmployeePk, EmployeeDTO empDTO) throws FinchException;  
  
/**  
 * Cancel given employee  
 */  
void cancelEmployee(long employeePk) throws FinchException;  
/**  
 * Cancel given employee  
 */
```

```
    void cancelEmployee(Employee employee) throws FinchException;  
}  
..
```

com.nrft.fin.ch.inf.domain.service.EmployeeServiceImpl

 [Expand](#)

```
..  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
  
    private static final Logger log = LoggerFactory  
        .getLogger(EmployeeServiceImpl.class);  
  
    @Autowired  
    private EmployeeRepository empRepo;  
  
    @Autowired  
    ApplicationRoleRepository applRoleRepo;  
  
    /*  
     * (non-Javadoc)  
     *  
     * @see  
     * com.nrft.fin.ch.inf.domain.service.EmployeeService#getEmployee(java.  
     * lang.String)  
     */  
    @Override  
    @Transactional(value = "global-tm", readOnly = true)  
    public Employee getEmployeeByUserId(String userId) throws FinchException {  
        Optional<Employee> emp = empRepo.getEmployeeByUserId(userId);  
        if (emp.isPresent())  
            return emp.get();  
        else  
            return null;  
    }  
  
    @Override  
    public Employee getEmployeeByUserIdAndStatus(String userId, String status)  
        throws FinchException {  
        Optional<Employee> emp = empRepo.getEmployeeByUserId(userId, status);  
        if (emp.isPresent())  
            return emp.get();  
        else  
            return null;  
    }  
..  
..  
/*  
 * (non-Javadoc)  
 *  
 * @see  
 * com.nrft.fin.ch.inf.domain.service.EmployeeService#findEmployeeByPk(long)  
 */  
    @Override  
    @Transactional(value = "global-tm", readOnly = true)  
    public Employee findEmployeeByPk(long employeePk) throws FinchException {
```

[source](#)

```

Employee emp = empRepo.find(Employee.class, employeePk);
emp.getEmpApplnRoleParticipants().size();
emp.getEmployeeNameXrefs().size();
return emp;
}

/*
 * (non-Javadoc)
 *
 * @see
 * com.nrft.fin.ch.inf.domain.service.EmployeeService#enterEmployee(com.
 * nrft.fin.ch.inf.web.form.EmployeeEntryCommandForm)
 */
@Override
@Transactional(value = "global-tm", readOnly = false, rollbackFor = { FinchException.class })
public Employee enterEmployee(EmployeeDTO empDTO) throws DataInputException {}

...
...

/***
 * Employee Role Amend populate helper method
 *
 * @param employee
 * @param empDTO
 * @throws FinchException
 */
private void amendRoleParticipant(Employee employee, EmployeeDTO empDTO)
    throws FinchException {}

/*
 * (non-Javadoc)
 *
 * @see
 * com.nrft.fin.ch.inf.domain.service.EmployeeService#updateEmployee(long,
 * com.nrft.fin.ch.inf.web.form.EmployeeDTO)
 */
@Override
@Transactional(value = "global-tm", readOnly = false, rollbackFor = { FinchException.class })
public void updateEmployee(long originalEmployeePk, EmployeeDTO empDTO)
    throws FinchException {}
}

@Override
public void cancelEmployee(long employeePk) throws FinchException {
    Employee employee = findEmployeeByPk(employeePk);
    empRepo.cancelEmployee(employee);
}

@Override
@Transactional(value = "global-tm", readOnly = false, rollbackFor = { FinchException.class })
public void cancelEmployee(Employee employee) throws FinchException {}

private <T> T getService(Class<T> cl) {
    return Operation.getInstance().getReference(cl);
}

```

```
}
```

```
..
```

4.5 Create the Repository

While creating the repository class, ensure about the followings:

- The class must be annotated as @Repository
- While defining the EntityManager we must use the @PersistenceContext annotation and specify the context. It can be either global-em or sample-em. global-em is used for INF entities and sample-em is used for sample app entities.

Repository contains all the necessary methods to performing create, read, update and delete operation in database.

Create the repository class, as shown below:

EmployeeUIRepository

» [Expand](#)

[source](#)

```
@Repository
public class EmployeeUIRepository extends RepositoryBase<Employee, Long> {
    /**
     * The <code>log</code> instance of this class.
     */
    private static final Logger log = LoggerFactory
        .getLogger(EmployeeUIRepository.class);
    @PersistenceContext(unitName = Globals.ENTITY_MANAGER)
    private EntityManager entityManager;
    /**
     * Gets the entity manager corresponding to the persistence context
     *
     * @return {@link EntityManager}
     */
    @Override
    public EntityManager getEntityManager() {
        return this.entityManager;
    }
    /**
     * Fetches the list of employee for report view.
     *
     * @return list of {@link EmployeeQueryResultView}
     * @throws FinchException
     */
    public List<EmployeeQueryResultView> getEmpReportViewList()
        throws FinchException {
        return this.executeQuery(new EmployeeDTO());
    }
    /**
     *
     * @param employeeDTO
     * @return
     * @throws FinchException
     */
    public List<EmployeeQueryResultView> executeQuery(EmployeeDTO employeeDTO)
        throws FinchException {
```

```

CriteriaBuilder builder = getEntityManager().getCriteriaBuilder();
CriteriaQuery<EmployeeQueryResultView> cq = builder
    .createQuery(EmployeeQueryResultView.class);
Root<Employee> empRoot = cq.from(Employee.class);
cq.multiselect(empRoot.get("employeePk"), empRoot.get("firstName"),
    empRoot.get("lastName"), empRoot.get("userId"),
    empRoot.get("startDate"), empRoot.get("title"),
    empRoot.get("employeeCloseDate"),
    empRoot.get("employeeClosedBy"), empRoot.get("employeeId"),
    empRoot.get("employeeOpenDate"),
    empRoot.get("employeeOpenedBy"), empRoot.get("endDate"),
    empRoot.get("mail"), empRoot.get("middleInitial"),
    empRoot.get("status"));
List<Predicate> predList = Lists.newArrayList();
// Criteria
if (!Strings.isNullOrEmpty(employeeDTO.getUserId())) {
    predList.add(builder.equal(empRoot.get("userId"),
        StringUtils.lowerCase(employeeDTO.getUserId().trim())));
}
if (!Strings.isNullOrEmpty(employeeDTO.getFirstName())) {
    predList.add(builder.equal(empRoot.get("firstName"), employeeDTO
        .getFirstName().trim()));
}
if (!Strings.isNullOrEmpty(employeeDTO.getLastName())) {
    predList.add(builder.equal(empRoot.get("lastName"), employeeDTO
        .getLastName().trim()));
}
if (employeeDTO.getStartDate() != null) {
    predList.add(builder.equal(empRoot.get("startDate"),
        employeeDTO.getStartDate()));
}
if (!Strings.isNullOrEmpty(employeeDTO.getTitle())) {
    predList.add(builder.equal(empRoot.get("title"), employeeDTO
        .getTitle().trim()));
}
if (employeeDTO.getEmployeeCloseDate() != null) {
    predList.add(builder.equal(empRoot.get("employeeCloseDate"),
        DateUtils.formatDate(employeeDTO.getEmployeeCloseDate())));
}
if (!Strings.isNullOrEmpty(employeeDTO.getEmployeeClosedBy())) {
    predList.add(builder.equal(empRoot.get("employeeClosedBy"),
        employeeDTO.getEmployeeClosedBy().trim()));
}
if (!Strings.isNullOrEmpty(employeeDTO.getEmployeeId())) {
    predList.add(builder.equal(empRoot.get("employeeId"), employeeDTO
        .getEmployeeId().trim()));
}
if (employeeDTO.getEmployeeOpenDate() != null) {
    predList.add(builder.equal(empRoot.get("employeeOpenDate"),
        employeeDTO.getEmployeeOpenDate()));
}
if (!Strings.isNullOrEmpty(employeeDTO.getEmployeeOpenedBy())) {
    predList.add(builder.equal(empRoot.get("employeeOpenedBy"),
        employeeDTO.getEmployeeOpenedBy().trim()));
}
if (employeeDTO.getEndDate() != null) {
    predList.add(builder.equal(empRoot.get("endDate"),
        DateUtils.formatDate(employeeDTO.getEndDate())));
}

```

```

        if (!Strings.isNullOrEmpty(employeeDTO.getMail())) {
            predList.add(builder.equal(empRoot.get("mail"), employeeDTO
                .getMail().trim()));
        }
        if (!Strings.isNullOrEmpty(employeeDTO.getMiddleInitial())) {
            predList.add(builder.equal(empRoot.get("middleInitial"),
                employeeDTO.getMiddleInitial().trim()));
        }
        if (!Strings.isNullOrEmpty(employeeDTO.getStatus())) {
            predList.add(builder.equal(empRoot.get("status"), employeeDTO
                .getStatus().trim()));
        }
        if (!predList.isEmpty()) {
            cq.where(builder.and(predList.toArray(new Predicate[] {})));
        }
        List<EmployeeQueryResultView> result = getEntityManager().createQuery(
            cq).getResultList();
        return result;
    }
    /**
     *
     * @param queryInput
     * @return
     * @throws FinchException
     */
    public long executeQueryCount(Map<String, Object> queryInput)
        throws FinchException {
        CriteriaBuilder builder = getEntityManager().getCriteriaBuilder();
        CriteriaQuery<Long> cq = builder.createQuery(Long.class);
        Root<Employee> empRoot = cq.from(Employee.class);
        cq.select(builder.count(empRoot));
        List<Predicate> predList = Lists.newArrayList();
        // Criteria
        if (!Strings.isNullOrEmpty(getInputParam("userId", String.class,
            queryInput))) {
            predList.add(builder.equal(empRoot.get("userId"),
                StringUtils.lowerCase(getInputParam("userId", String.class,
                    queryInput))));
        }
        if (!Strings.isNullOrEmpty(getInputParam("firstName", String.class,
            queryInput))) {
            predList.add(builder.equal(empRoot.get("firstName"),
                getInputParam("firstName", String.class, queryInput)));
        }
        if (!Strings.isNullOrEmpty(getInputParam("lastName", String.class,
            queryInput))) {
            predList.add(builder.equal(empRoot.get("lastName"),
                getInputParam("lastName", String.class, queryInput)));
        }
        if (!Strings.isNullOrEmpty(getInputParam("startDate", String.class,
            queryInput))) {
            try {
                Date fromDate = DateUtils.parseDate(getInputParam("startDate",
                    String.class, queryInput));
                String formatedDate = DateUtils.formatDate(fromDate,
                    "yyyy-MM-dd");
                predList.add(builder.equal(empRoot.get("startDate"),
                    formatedDate));
            } catch (Exception e) {

```

```

        log.error("Failed to parse start date", e);
        throw new InfException("error.start.date", getInputParam(
            "startDate", String.class, queryInput));
    }
}
if (!Strings.isNullOrEmpty(getInputParam("title", String.class,
    queryInput))) {
    predList.add(builder.equal(empRoot.get("title"),
        getInputParam("title", String.class, queryInput)));
}
if (!Strings.isNullOrEmpty(getInputParam("employeeCloseDate",
    String.class, queryInput))) {
    try {
        Date fromDate = DateUtils.parseDate(getInputParam(
            "employeeCloseDate", String.class, queryInput));
        String formatedDate = DateUtils.formatDate(fromDate,
            "yyyy-MM-dd");
        predList.add(builder.equal(empRoot.get("employeeCloseDate"),
            formatedDate));
    } catch (Exception e) {
        log.error("Failed to parse employee close date", e);
        throw new InfException("error.close.date", getInputParam(
            "employeeCloseDate", String.class, queryInput));
    }
}
if (!Strings.isNullOrEmpty(getInputParam("employeeClosedBy",
    String.class, queryInput))) {
    predList.add(builder.equal(empRoot.get("employeeClosedBy"),
        getInputParam("employeeClosedBy", String.class, queryInput)));
}
if (!Strings.isNullOrEmpty(getInputParam("employeeId", String.class,
    queryInput))) {
    predList.add(builder.equal(empRoot.get("employeeId"),
        getInputParam("employeeId", String.class, queryInput)));
}
if (!Strings.isNullOrEmpty(getInputParam("employeeOpenDate",
    String.class, queryInput))) {
    try {
        Date fromDate = DateUtils.parseDate(getInputParam(
            "employeeOpenDate", String.class, queryInput));
        String formatedDate = DateUtils.formatDate(fromDate,
            "yyyy-MM-dd");
        predList.add(builder.equal(empRoot.get("employeeOpenDate"),
            formatedDate));
    } catch (Exception e) {
        log.error("Failed to parse employee open date", e);
        throw new InfException("error.open.date", getInputParam(
            "employeeOpenDate", String.class, queryInput));
    }
}
if (!Strings.isNullOrEmpty(getInputParam("employeeOpenedBy",
    String.class, queryInput))) {
    predList.add(builder.equal(empRoot.get("employeeOpenedBy"),
        getInputParam("employeeOpenedBy", String.class, queryInput)));
}
if (!Strings.isNullOrEmpty(getInputParam("endDate", String.class,
    queryInput))) {
    try {
        Date fromDate = DateUtils.parseDate(getInputParam("endDate",

```

```

        String.class, queryInput));
    String formatedDate = DateUtils.formatDate(fromDate,
        "yyyy-MM-dd");
    predList.add(builder.equal(empRoot.get("endDate"), formatedDate));
} catch (Exception e) {
    log.error("Failed to parse end date", e);
    throw new InfException("error.end.date", getInputParam(
        "endDate", String.class, queryInput));
}
}

if (!Strings.isNullOrEmpty(getInputParam("mail", String.class,
    queryInput))) {
    predList.add(builder.equal(empRoot.get("mail"),
        getInputParam("mail", String.class, queryInput)));
}

if (!Strings.isNullOrEmpty(getInputParam("middleInitial", String.class,
    queryInput))) {
    predList.add(builder.equal(empRoot.get("middleInitial"),
        getInputParam("middleInitial", String.class, queryInput)));
}

if (!Strings.isNullOrEmpty(getInputParam("status", String.class,
    queryInput))) {
    predList.add(builder.equal(empRoot.get("status"),
        getInputParam("status", String.class, queryInput)));
}

if (!predList.isEmpty()) {
    cq.where(builder.and(predList.toArray(new Predicate[] {})));
}

return getEntityManager().createQuery(cq).getSingleResult();
}

/**
 *
 * @param paramName
 * @param clazz
 * @param queryInput
 * @return
 */
private <T> T getInputParam(String paramName, Class<T> clazz,
    Map<String, Object> queryInput) {
    return clazz.cast(queryInput.get(paramName));
}

/**
 * @param builder
 * @param fieldPathMap
 * @param field
 * @param fieldOrder
 * @return
 */
Order prepareOrder(CriteriaBuilder builder,
    Map<String, Path<?>> fieldPathMap, String field, String fieldOrder) {
    Order order = null;
    if (!Strings.isNullOrEmpty(field)) {
        if ("DESC".equals(fieldOrder)) {
            order = builder.desc(fieldPathMap.get(field));
        } else {
            order = builder.asc(fieldPathMap.get(field));
        }
    }
    return order;
}

```

```
}

/**
 * @param entityManager
 *      the entity manager to set
 */
@PersistenceContext(unitName = "global-em")
public void setEntityManager(EntityManager entityManager) {
    this.entityManager = entityManager;
    this.entityManager.setFlushMode(FlushModeType.COMMIT);
    log.debug("Entity manager = " + this.entityManager + " is open = "
        + this.entityManager.isOpen());
}
```

```
}
```

5. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

6. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	19-02-2015	Initial Version	Rahul c
0.2	24-02-2015	Reviewd	Suvasish Samaddar

3.3.4.2 Single Page Cancel

1. Scope

In finch application, some entities are created in a single step. The entities like Employee, Holidays etc., which have lesser number of fields associated with them, are ideal candidates for single page entry . Generally only a single table or a table with very few linked child tables are used to store the data for such entities. In order to gather all information related to these entities, the input fields are included in a single page. To cancel such an entity like, Employee, Market Price, Country, SLR Contract, Exchange Rate etc, the single page cancel flow is used. Like single page entry screen, single page cancel is also followed by the user confirmation and system confirmation screens. For cancel, user starts by query and then into the cancel flow from the query result screen. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation for single page cancel only. It outlines all the necessary steps, both on the client side and server side, to cancel an existing entity. As the cancel screen is launched from query screen, some query related flow, relevant to cancel, is also mentioned in this document.

2. Prerequisites

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception Handling](#) for proper handling of exceptions.

Refer to [Single Page Entry](#) for the details on how to create a single page entry screen.

Refer to [Query/Query Result](#) for the details on the features and components of query screen.

Refer to [Single Page Amend](#) for the details on screen consolidation.

Refer to [Handle of Concurrent operation](#) for the details on the features of concurrency handling in finch.

Refer to [Add a new module](#) for details on how to add a new module in finch.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

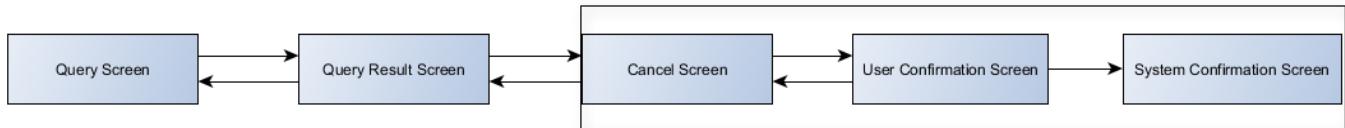
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The single page cancel is launched from query screen. The complete screen flow for single page cancel, with the target screens highlighted, is shown below :



Screen	Purpose
Query	It is used to query for a particular entity based on some or all of its associated fields.
Query Result	It is used to display the result obtained based on the query criteria. User can select a record from this screen and initiate cancel sub-action on it.
Cancel	It is used to view the details of a particular record before cancelling it. It also allows user to submit the form for user confirmation.
User Confirmation	It is used to confirm whether the user wants to continue with the cancellation and choose a suitable cancellation reason code, if any, before proceeding further for system confirmation. User can go back to the cancel screen to reconsider the cancellation.
System Confirmation	It confirms that the status of the cancelled record has been successfully saved in the database. It shows the system confirmation message along with the reference number, wherever applicable, for the completed transaction.

2.3 UI components

The amend page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing amend views, should be retained to get the basic layout, look & feel and functionality of form forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side.

The form input data validations are handled on the client side and the business validations are handled on the server side.

See the [Exception Management](#) document for more details on error handling.

Please note that we have used the Employee Cancel use case as a reference throughout this document, for all the explanations.

3. Sequence of Steps

3.1 Create the model

3.1.1 Create Tables and Sequences

See the [Single Page Entry - 4.1.1 Create Tables and Sequences](#) for the new entity section for Create Tables and Sequences for the new entity.

3.1.1 CRUD Generation Tool

See the [Single Page Entry - 4.1.2 CRUD Generation Tool](#) section for CRUD Generation Tool details.

3.1.3 Create Command Form

The command form and page class created for single page amend will be reused for cancel operation

See the [Single Page Entry - 4.1.3 Create Command Form](#) section for complete details.

3.2 Create the view

3.2.1 Screen Consolidation

In order to reduce the number of menu items and consolidate the finch screens, cancel action is not available as a separate menu item. In menu, only entry and query items are available and cancel action is associated with each record in the query result screen.

Developer needs to set up roles, actions and menu items for every new cancel page.

See the [Single Page Amend - Screen Consolidation](#) section for complete details.

The actionUrl associated with the cancel action is used to navigate to the cancel page.

For example,

For employee disable/cancel, the ACTION_URL mapped to "EMPLCX" is "/inf/employee/cancel/actionable.json". The SCREEN_ID and VERSION_NUMBER mapped to "EMPLCX" having SCREEN_KEY="13" is "EMPCX" and "1.1" respectively.

So the actionUrl formed is "inf/employee/cancel/actionable.json?screenId=EMPCX&versionNo=1&pk=1223&type=cancel". On clicking the disable icon from the employee query result page, user navigates to the employee cancel page.

See the [Query/Query Result - Initialize new query result page](#) section for details regarding action consolidation.

3.2.2 Basic components of single page cancel screen

Query Page
Query Result Page
Query result page showing the sub-action menu
Cancel page
Cancel User Confirmation Page
Cancel System Confirmation Page

Search Menu 

EMPQR:Employee Query [20140303]

User ID	<input type="text"/>	First Name	<input type="text"/>	Middle Initial	<input type="text"/>	Last Name
Title	<input type="text"/>	Start Date From - To	<input type="text"/>  <input type="text"/> 	Employee Open Date From - To	<input type="text"/>  <input type="text"/> 	Employee Status
Status	<input type="text"/>					
Sort Criteria	① <input type="text"/>  <input type="checkbox"/> DESC	② <input type="text"/>  <input type="checkbox"/> DESC	③ <input type="text"/>  <input type="checkbox"/> DESC			
PERSONALIZE SAVE QUERY						

For details regarding the query page blocks, components and features see [Query/Query Result - Basic components of query/query result page section](#).

Search Menu

EMPRS:Employee Query Result [20140303]

Start Date	Employee ID	Employee Name	Title	Middle Initial	Employee Last Name	Employee First Name	Status
201210...							NORMAL
201401...	20140106	finch	ABCD	film			NORMAL
20140811	20140806	finch	asd	mi			NORMAL
201401...			Mr.				NORMAL
201403...	20140305	finch		lnf			NORMAL
201407...			Mr.				NORMAL
201401...	20140109	finch		Gnt			NORMAL
201402...	20140227	finch					NORMAL
201402...	20140228	kabira					NORMAL
201402...	20140228	kabiratna					NORMAL

For details regarding the query result page components and features see [Query/Query Result - Basic components of query/query result page](#) section.

Search Menu 

EMPRS:Employee Query Result [20140303]

Page 1 of 27 Records: 261

Start Date	Employee ID	Employee Name	Title	Middle Initial	Employee Last Name	Employee First Name	Status
201210...	20140106	finch	ABCD	film			NORMAL
20140811	20140806	finch	asd	mi			NORMAL
201401...			Mr.				NORMAL
201403...	20140305	finch		Inf			NORMAL
201407...			Mr.				NORMAL
201401...	20140109	finch		Gnt			NORMAL
201402...	20140227	finch					NORMAL
201402...	20140228	kabira					NORMAL
201402...	20140228	kabiratna					NORMAL

For details regarding the query result page components and features see [Query/Query Result - Basic components of query/query result page & Initialize new query result page](#) section

Search Menu

EMPCX:Employee Cancel [03/03/2014]

User ID	ashu12b	First Name	Finch Dev 1	Middle Initial
Last Name	Sr	Default Branch	FBR	Title
Start Date (UTC)	03/05/2014	End Date (UTC)		Suffix
E-Mail		Employee Opened By	finch	

Application Roles

Branch Id	Application Role
FBR	ROLE1
FBR	ROLE2

1

2

1. The same non-editable details page used for single page entry user confirmation is also used for single page cancel. See the [Single Page Entry - Basic components of single page entry screen](#) section for details.
2. Cancel view is a non-editable page, so there is no need of reset button. Rest of the action area is similar to the action area for single page amend screen. See the [Single Page Amend - Basic components of single page entry screen](#) section

for details.

The screenshot shows a software interface for canceling an employee record. At the top, there's a navigation bar with a search menu and a magnifying glass icon. Below it, the title "EMPCX:Employee Cancel [03/03/2014]" is displayed. The main area contains a table with the following data:

User ID	nrift-kb	First Name	nrift-kb	Middle Initial
Last Name		Default Branch	FBR	Title
Start Date (UTC)	02/27/2014	End Date (UTC)		Suffix
E-Mail		Employee Opened By	kabira	

Below this, there's a section titled "Application Roles" with a table:

Branch Id	Application Role

The bottom of the screen features a dark footer bar with icons for gear, user profile, and other system functions.

1. Section containing cancel user confirmation message- See Create user confirmation page for cancel - Cancel user confirmation message section below for details.
2. The action area is identical to the entry user confirmation page. See the [Single Page Entry - Basic components of single page entry screen](#) section for all details.

Search Menu Cancel [03/03/2014]

User ID	nrift-kb	First Name	nrift-kb	Middle Initial
Last Name		Default Branch	FBR	Title
Start Date (UTC)	02/27/2014	End Date (UTC)		Suffix
E-Mail		Employee Opened By	kabira	

Application Roles

Branch Id	Application Role

This page is identical to the entry system confirmation page. See the [Single Page Entry - Basic components of single page entry screen section](#) and [Single Page Amend - Create system confirmation page for amend section for all details.](#)

3.2.3 Create cancel page

Developer does not need to create a new view for cancel. The same non-editable details page used for single page entry/amend user confirmation is also used for single page cancel.

See the [Detail Page Layout](#) document for further details.

The single page cancel view differs from the entry/amend views in that it is a non-editable view and so it does not have the Reset button in the action area section.

- The request for fetching the cancel view is handled by consolidateLinkHandler method defined in finch-consolidation.js.

See the [Query/Query Result](#) - Initialize new query result page - Define the action consolidation point for details on consolidateLinkHandler method.

The buttons in the action area of the cancel page are determined based on the actionType and mode properties of the options object used to load the wizard plugin.

The cancel page allows users to :

- View the details of the entity
- Navigate back to the query result page using Back button
- Submit the form using Submit button

Cancel page is a non-editable page so it does not have the Reset button. The other action area buttons are similar to the single page amend flow. See the [Single Page Amend \(Consolidation\)](#) - Create amend page section for further details.

For example,

For employee cancel, on clicking the back button user navigates back to the employee query result screen.

The back handler, defined in finch-wizard.js, handles all back operations in the cancel flow as shown below. It applies the back logic depending on the actionType property of the options object used to load the wizard plugin.

finch-wizard.js

› [Expand](#)

[source](#)

```

this.back = function() {
    // check if back is called from 1st mode
    if (self.modelIndex == 0) {
        // if called from 1st mode, then its a call from consolidated action
        var FINCH$Handler$backHandler = FINCH$Handler$function({
            get : {
                requestType : FINCH$Handler$default.requestType.asynchronous,
                target : '#content'
            },
            settings : {
                type : 'POST',
                beforeSend : function(request) {
                    request.setRequestHeader('Accept',
                        'text/html;type=ajax');
                }
            }
        });
        var url = FINCH.context.path;
        var queryCommandFormId = self.$context.find('#queryCommandFormId')
            .val();
        var resultScreenURL = self.$context.find('#resultScreenUrl').val();
        var resultSortAsc = self.$context.find('#resultSortAsc').val();
        var resultSortCol = self.$context.find('#resultSortCol').val();
        var baseQueryURL = resultScreenURL.substring(0, resultScreenURL
            .lastIndexOf("/"));
        var commandFormDetail = {resultSortAsc: resultSortAsc, resultSortCol : resultSortCol };
        var backToResultURL = "/backToCurrentResult?commandFormId="
            + queryCommandFormId + "&fragments=content";
        if (resultScreenURL.length > 0 && queryCommandFormId.length > 0) {
            url = url + baseQueryURL + backToResultURL;
            FINCH$Handler$backHandler.generic(undefined, {
                requestUri : url,
                settings: {data : commandFormDetail},
                onHtmlContent : function(e, options, $target, content) {
                    $target.html(content);
                }
            });
        } else {
            FINCH.postNotice(FINCH.notice.type.error,
                'This feature is not supported !!', true);
        }
    } else {
        if(self.skipWizard === true) {
            self.skipPreviousMode('back');
        } else {
            // asynchronous handler to handle success action
            self.action('back', {
                success : function() {
                    if (self.modelIndex > 0) {
                        self.modelIndex -= 1;
                        self.pageIndex = self.pageIndexes[self.modelIndex];
                    }
                }
            });
        }
    }
};

```

3.2.4 Create user confirmation page for cancel

As mentioned above, after successful submission of the cancel form, user navigates to the user confirmation page.

Developer needs to create a new view for cancel user confirmation page. On submitting the single page cancel form for user confirmation, the corresponding controller should return the user confirmation view tile name. In views.xml, this tile name needs to be mapped to the new view.

The action area buttons of the cancel user confirmation page are similar to the buttons displayed in single page entry/amend user confirmation page.

See the [Single Page Entry - Create user confirmation for new entry section](#) for further details..

3.2.5 Create system confirmation page for cancel

After successful submission of the user confirmation page, user navigates to the system confirmation page.

The system confirmation page for cancel differs from entry system confirmation page in allowing the user to go back to the query result screen using the Ok button.

For example,

In the employee cancel flow, after successful form submission, user navigates to the system confirmation page where the confirmation message is shown.

On clicking the Ok button here, user goes back to the employee query result page.

See the [Single Page Amend - Create system confirmation page for amend section](#) for further details.

3.3 Create the controller

3.3.1 Follow Inheritance rule

See the [Single Page Amend \(Consolidation\) - 4.3.1 Follow Inheritance rule](#) section for complete details.

3.3.2 Provide request URL mapping

See the [Single Page Amend \(Consolidation\) - 4.3.2 Provide request URL mapping](#) section for complete details.

3.3.3 Check the requested action is to be performed or not

user can't perform a cancel operation on already cancelled record.

See the [Single Page Amend \(Consolidation\) - 4.3.3 Check the requested action is to be performed or not](#) section for complete details.

3.3.4 Define screen specific initialization logic

See the [Single Page Amend \(Consolidation\) - 4.3.4 Define screen specific initialization logic](#) section for complete details.

3.3.5 Define server side validation logic

See the [Single Page Amend \(Consolidation\) - 4.3.5 Define server side validation logic](#) section for complete details.

3.3.6 Define form submit and user confirmation logic

See the [Single Page Amend \(Consolidation\) - 4.3.6 Define form submit and user confirmation logic](#) section for complete details.

3.3.7 Define system confirmation and data persistence logic

See the [Single Page Amend \(Consolidation\) - 4.3.7 Define system confirmation and data persistence logic](#) section for complete details.

433.7.1 Concurrency handling

See the [Single Page Amend \(Consolidation\) - 4.3.7.1 Concurrency handling](#) section for complete details.

3.3.8 Transaction Management

Transaction has been properly managed by the base class. The developer has nothing to do with this.

For details of Transaction Management see [Transaction Management](#).

3.3.12 Access Log

In finch, whenever a Single Page Cancel screen is being added the Access Log implementation should be done properly. From this Access Log, we can get history of accessing of the screen.

For Access Log of a Single Page Cancel, the developer has to only use the standard URL pattern for amend operation. The rest has been handled by the framework. Standard URL pattern for an operation is like "secure/<module_in_small>/<entity_in_small>/<operation_in_small>/*", where ref, trd, stl are the modules; account, customer, trade, settlement are the entities; entry, amend, query are the operations. For example, the URL for Strategy amend should be like 'secure/ref/marketprice/cancel/actionable.json'.

If any URL does not follow the standard pattern, then one has to configure that pattern in database tables for the Access Log implementation. For details of

this setting see [Access Log](#).

4. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

5. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	24-02-2015	Initial Version	Rahul c

3.3.4.3 Single Page Amend

1. Scope

In finch application, some entities are created in a single step. The entities like Employee, Holidays etc., which have lesser number of fields associated with them, are ideal candidates for single page entry . Generally only a single table or a table with very few linked child tables are used to store the data for such entities. In order to gather all information related to these entities, the input fields are included in a single page. To amend such an entity like, Employee the single page amend flow is used. Like single page entry screen, single page amend is also followed by the user confirmation and system confirmation screens. For amend, user starts by query and then into the amend flow from the query result screen. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation for single page amend only . It outlines all the necessary steps, both on the client side and server side, to amend an existing entity. As the amend screen is launched from query screen, some query related flow, relevant to amend, is also mentioned in this document.

2. Prerequisites

MUST SEE

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception Handling](#) for proper handling of exceptions.

Refer to [Single Page Entry](#) for the details on how to create a single page entry screen.

Refer to [Query/Query Result](#) for the details on the features and components of query screen.

Refer to [Handle of Concurrent operation](#) for the details on the features of concurrency handling in finch.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

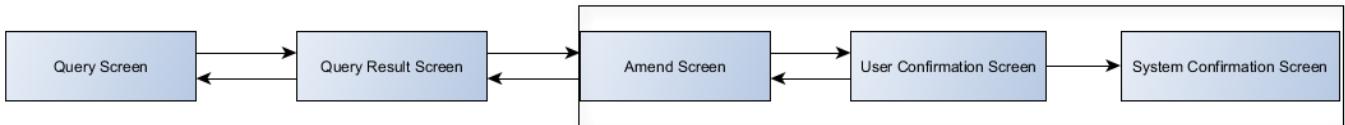
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The single page amend is launched from query screen. The complete screen flow for single page amend, with the target screens highlighted, is shown below :



Screen	Purpose
Query	It is used to query for a particular entity based on some or all of its associated fields.
Query Result	It is used to display the result obtained based on the query criteria. User can select a record from this screen and initiate amend sub-action on it.
Amend	It is used to update/provide input in the form fields for amending a particular entity. It also allows user to submit the updated form for user confirmation.
User Confirmation	It is used to preview and confirm the input data before proceeding further for system confirmation. If the input needs modification then user can go back to the amend screen.
System Confirmation	It confirms that the updated data has been successfully saved in the database. It shows the system confirmation message along with the reference number, wherever applicable, for the completed transaction.

2.3 UI components

The amend page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing amend views, should be retained to get the basic layout, look & feel and functionality of form forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side.

The form input data validations are handled on the client side and the business validations are handled on the server side.

See the [Exception Management](#) document for more details on error handling.

Please note that we have used the Employee Amend use case as a reference throughout this document, for all the explanations.

3.New and Modified File List

Layer	Package/Location(sample)	File type/purpose	File name (sample)
Entity	/finch-core/src/main/java/com/nrift/finch/inf/domain/entity	The persistent class for database table	Employee.java
Repository	/finch-core/src/main/java/com/nrift/finch/inf/domain/repository	Contains methods for retrieving domain objects	EmployeeRepository.java
	/finch-web-infra/src/main/java/com/nrift/finch/inf/domain/repository	Contains all the necessary methods to perform CRUD operation in database	EmployeeUIRepository.java

Service	/finch-core/src/main/java/com/nrft/finch/inf/domain/service/	Interface that provides methods related to Employee	EmployeeService.java
ServiceImplementation	/finch-core/src/main/java/com/nrft/finch/inf/domain/service/	Service implementation for Employee Service	EmployeeServiceImpl.java
Model	/finch-web-infra/src/main/java/com/nrft/finch/inf/web/employee	Command Form (i.e. POJO) class	EmployeeAmendCommandForm.java
View	/finch-web-infra/src/main/web/WEB-INF/views/inf	Module specific xml file to provide view tile definition	views.xml
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for en locale	finch-i18n_en.js
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for ja locale	finch-i18n_ja.js
	/finch-web-infra/src/main/web/WEB-INF/views/inf/user/	jsp for editable entry view	userAmend.jspx
	/finch-web-infra/src/main/web/WEB-INF/views/inf/user/	jsp for non-editable user & system confirmation view	userDetail.jspx
Controller	/finch-web-infra/src/main/java/com/nrft/finch/inf/web/controller/	Controller class, contains all the action APIs for Entry	EmployeeAmendController.java
Repository	/finch-web-infra/src/main/java/com/nrft/finch/inf/domain/repository	contains all the necessary methods to perform CRUD operation	EmployeeUIRepository.java

Some other Helper and Utility classes can be used, but they differ from functionality to functionality.

4. Sequence of Steps

4.1 Create the model

4.1.1 Create Tables and Sequences

See the [Single Page Entry - 4.1.1 Create Tables and Sequences](#) for the new entity section for Create Tables and Sequences for the new entity.

4.1.1 CRUD Generation Tool

See the [Single Page Entry - 4.1.2 CRUD Generation Tool](#) section for CRUD Generation Tool details.

4.2.2 Create Command Form

For sending and receiving the Form/Request values between client and server side, finch uses POJO classes i.e. CommandForms. A CommandForm contains all the field names (JSP field names used in the screen) along with their getter() and setter() method. One should be very careful about the data types of the member variables of a CommandForm. If there exists any mismatch between the data types of the variables of CommandForm and the variables used in the screen, it'll lead to an exception.

In finch , a CommandForm for a single page entry should always extend the base class com.nrft.fin.ch.inf.web.FinchAbstractWizardCommandForm. For Employee entry, the EmployeeAmendCommandForm.java has been defined in the package "com.nrft.fin.ch.inf.web.employee"

Note

For entry & amend single command form can be reused.

A CommandForm should always have a method which'll reset the member variables like the following:

EmployeeAmendCommandForm.java

Expand

[source](#)

```
public class EmployeeAmendCommandForm extends FinchAbstractWizardCommandForm {  
    private static final long serialVersionUID = 1L;  
    public static final int GENERAL_TAB = 0;  
    public static final int DETAILS_TAB = 1;  
    // hidden fields  
    private String mode = null;  
    private String message = StringUtils.EMPTY;  
    private Employee employee;  
    private String applPasswd;  
    ....  
    * @return the employeeld  
    */  
    public String getEmployeeld() {  
        return this.employeeld;  
    }  
    /**  
     * @param employeeld  
     *          the employeeld to set  
     */  
    public void setEmployeeld(String employeeld) {  
        this.employeeld = employeeld;  
    }  
    public void reset() {  
    }  
    ....  
}
```

To enable character set encoding validation use annotation @CharsetEncoding. For details of this implementation please see [CharacterSet Validation](#)

4.2 Create the view

4.2.1 Screen Consolidation

In order to reduce the number of menu items and consolidate the finch screens, amend action is not available as a separate menu item. In menu, only entry and query items are available and amend action is associated with each record in the query result screen.

Based on the application role of the logged in user, some, all or none of the actions will be available. Developer needs to set up roles and menu items for every new entry/amend page.

For example,

Let's consider Employee menu and actions entries for user finch.

INF_UI_MENU						
MENU_PK	SCREEN_KEY	MENU_NAME	MENU_PARENT_PK	DISPLAY_SEQ	COMPONENT_ID	ACTION_ID
10008	NULL	Employee	10006	3	INF	NULL
5	4	Entry	10008	1	INF	EMPLEN
6	5	Query	10008	1	INF	EMPLQR

INF_ACTION				
ACTION_ID	ACTION_NAME	ACTION_KEY	ACTION_URL	ACTION_URL_PATTERN
EMPLAM	Employee Amend	10	/inf/employee/amend/actionable.json	/inf/employee/amend/*
EMPLCX	Employee Cancel	13	/inf/employee/cancel/actionable.json	/inf/employee/cancel/*
EMPLEN	Employee Entry	4	/inf/employee/entry/init	/inf/employee/entry/*
EMPLQR	Employee Query	5	/inf/employee/query/init	/inf/employee/query/*

INF_ACTION_CONSOLIDATION		
ACTION_ID	SUB_ACTION_ID	DISPLAY_SEQ
EMPLQR	EMPLAM	1
EMPLQR	EMPLCX	2

For the sub actions available under each main action, the SCREEN_ID and VERSION_NUMBER are retrieved from INF_SCREEN table by matching the values of ACTION_KEY from the INF_ACTION table and SCREEN_KEY from the INF_SCREEN table.

INF_SCREEN		
SCREEN_KEY	SCREEN_ID	VERSION_NUMBER
4	EMPEN	1
5	EMPQR	2
10	EMPAM	1

For access control, INF_ACTION_ROLE_PARTICIPANT table will be considered.

REF_ACTION_ROLE_PARTICIPANT	
ACTION_ID	APP_ROLE_PK
EMPLEN	10
EMPLQR	10
EMPLAM	10
EMPLQR	10

Users with application role 1 will see both Employee Entry and Employee Query menu items.

While rendering the employee query result screen, it should query INF_ACTION with SCREEN_KEY = 5 to get ACTION_ID of the query screen. Now, with ACTION_ID = EMPLQR, INF_ACTION_CONSOLIDATION will be queried. There are two sub actions e.g. EMPLAM and EMPLCX. For each of these sub actions the SCREEN_ID and VERSION_NUMBER are retrieved from INF_SCREEN table. If the logged in user has application role 10, then all the actions will be available to the logged in user. If the application role is different for any action, say for "EMPLAMX" if the application role is "12", then that action will not be available to the user with application role 10.

See the [Add new Menu/Action](#) document and [Security Architecture - Application Role & Authorization](#) section for details on

screen consolidation and role setup.

Developer needs to include an action consolidation section in every new query result page to display icons for amend, enable and cancel actions as shown below :

userQueryResult.jspx

» [Expand](#) [source](#)

```
<div id="consolidateAct" class="consolidateActionArea"
    style="display: none;">
    <c:forEach items="${navAction}" var="record">
        <a class="consolidateActLink" href="${record.actionUrl}"> <c:choose>
            <c:when
                test="${fn:containsIgnoreCase(record.actionId, 'EMPLAM')}">
                    <spring:message code="inf.employee.amendaction.label.employeemend" htmlEscape="true"
var="amend"/>
                    
                </c:when>
                <c:when
                    test="${fn:containsIgnoreCase(record.actionId, 'EMPLCX')}">
                        <spring:message code="inf.employee.cancelaction.label.employeecancel" htmlEscape="true"
var="cancel"/>
                        
                </c:when>
                <c:otherwise>
                    <spring:message code="inf.employee.cancelaction.label.employeecancel" htmlEscape="true"
var="cancel"/>
                    
                </c:otherwise>
            </c:choose>
        </a>
    </c:forEach>
</div>
```

The href attribute of all the available sub actions link is set to the actionUrl property of navAction variable. For each of the sub-actions available to the user, this actionUrl is generated in EmployeeScreenActionRole.java by appending the ACTION_URL retrieved from the INF_ACTION table with the SCREEN_ID and VERSION_NUMBER retrieved from REF_SCREEN table. See Create the Controller section below for further details.

The actionUrl associated with the amend action is used to navigate to the amend page.

For example,

For employee amend, the ACTION_URL mapped to "EMPAM" is "/inf/employee/amend/actionable.json". The SCREEN_ID and VERSION_NUMBER mapped to "EMPLAM" having SCREEN_KEY="10" is "EMPAM" and "1.1" respectively.

So the actionUrl formed is "inf/employee/amend/actionable.json?screenId=EMPAM&versionNo=1&pk=1206&type=amend". On clicking the amend icon from the employee query result page, user navigates to the employee amend page.

See the [Query/Query Result - Initialize new query result page](#) section for details regarding action consolidation.

Query PageQuery Result PageQuery result page showing the sub-action menuAmend page blocksAmend User Confirmation Page
Amend System Confirmation Page

The screenshot displays the Oracle Database Application Express (APEX) interface. At the top, there is a navigation bar with a search menu icon and a magnifying glass icon. Below the navigation bar, the title "EMPQR:Employee Query [20140303]" is shown. The main content area contains several input fields and dropdown menus for searching employees. These fields include "User ID" (with a blue selection box), "First Name", "Middle Initial", "Last Name", "Title", "Start Date From - To" (with calendar icons), "Employee Open Date From - To" (with calendar icons), "Status" (with a dropdown arrow), and "Sort Criteria" (with three numbered sections, each containing a dropdown and a "DESC" checkbox). At the bottom of the search form are two buttons: "PERSONALIZE" and "SAVE QUERY". On the far left, there is a vertical sidebar with various application icons, including a gear icon at the bottom.

For details regarding the query page blocks, components and features see [Query/Query Result - Basic components of query/query result page section](#).

Search Menu 

EMPRS:Employee Query Result [20140303]

Page 1 of 27 Records: 261

	Start Date	Employee ID	Employee Name	Title	Middle Initial	Employee Last Name	Employee First Name	Status
 	201210...							NORMAL
 	201401...	20140106	finch	ABCD	film			NORMAL
 	20140811	20140806	finch	aed	mi			NORMAL
 	201401...			Mr.				NORMAL
 	201403...	20140305	finch		lnf			NORMAL
 	201407...			Mr.				NORMAL
 	201401...	20140109	finch		Gnt			NORMAL
 	201402...	20140227	finch					NORMAL
 	201402...	20140228	kabira					NORMAL
 	201402...	20140228	kabiratna					NORMAL

For details regarding the query result page components and features see [Query/Query Result - Basic components of query/query result page](#) section.

Search Menu 

EMPRS:Employee Query Result [20140303]

Page 1 of 27 Records: 261

Start Date	Employee ID	Employee Name	Title	Middle Initial	Employee Last Name	Employee First Name	Status
201210...	20140106	finch	ABCD	film			NORMAL
20140811	20140806	finch	asd	mi			NORMAL
201401...			Mr.				NORMAL
201403...	20140305	finch		lnf			NORMAL
201407...			Mr.				NORMAL
201401...	20140109	finch		Gnt			NORMAL
201402...	20140227	finch					NORMAL
201402...	20140228	kabira					NORMAL
201402...	20140228	kabiratna					NORMAL

For details regarding the query result page components and features see [Query/Query Result - Basic components of query/query result page](#) & [Initialize new query result page](#) section

EMPAM:Employee Amend [20140303]

User ID	ashu12b	First Name	Finch Dev 1	Middle Initial	Inf	Last Name
Default Branch	FBR	Title		Start Date (UTC)	20140305	End Date
Suffix	Suff	E-Mail				

Reset Password

<input type="checkbox"/> Password		Confirm Password	
-----------------------------------	--	------------------	--

Application Roles

Branch Id		Application Role	
		1	
Actions	Branch Id	Application Role	
	FBR	ROLE1	
	FBR	ROLE2	

1. Block containing the form items and grid of single page amend.
 2. Action area of amend form. Here apart from Reset & Submit, Back button is also displayed. See the Create amend page section below for details.
- For details regarding the UI components used in the amend form in both the blocks mentioned above, see the [Single](#)

Page Entry - Basic components of single page entry screen section .

Search Menu

EMAUT:Employee Amend User Confirmation [20140303]

User ID	ashu12b	First Name	Finch Dev 1	Middle Initial
Last Name	Str	Default Branch	FBR	Title
Start Date (UTC)	20140305	End Date (UTC)		Suffix
E-Mail		Employee Opened By	finch	

Application Roles

Branch Id	Application Role
FBR	ROLE1
FBR	ROLE2

This page is identical to the entry user confirmation page. See the [Single Page Entry - Basic components of single page entry screen section](#) for all details.

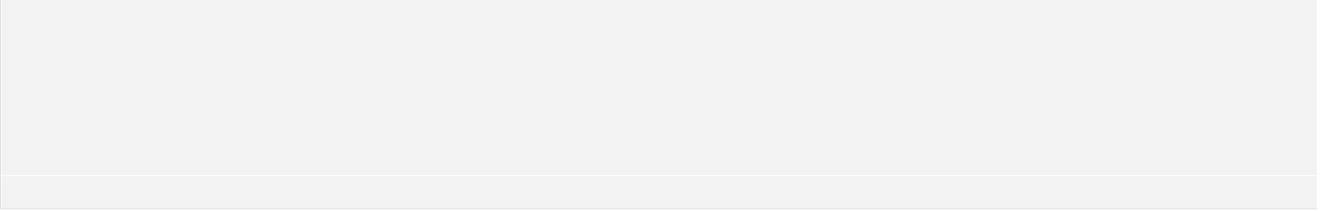
Search Menu 

EMASC:Employee Amend System Confirmation [20140303]

User ID	ashu12b	First Name	Finch Dev 1	Middle Initial
Last Name	Sr	Default Branch	FBR	Title
Start Date (UTC)	20140305	End Date (UTC)		Suffix
E-Mail		Employee Opened By	finch	

Application Roles

Branch Id	Application Role
FBR	ROLE1
FBR	ROLE2




This page is identical to the entry system confirmation page. See the [Single Page Entry - Basic components of single page entry screen section](#) and [Create system confirmation page for amend section](#) below for all details.

4.2.3 Create amend page

Developer does not need to create a new view for amend. The same view is used for single page entry and amend. The difference lies in

the pre-populated form field values and the action area section.

- The request for fetching the amend form field values is handled by `consolidateLinkHandler` method defined in `finch-consolidation.js`

See the [Query/Query Result - Initialize new query result page - Define the action consolidation point for details on `consolidateLinkHandler` method.](#)

- The buttons in the action area of the entry/amend page are determined based on the `actionType` and `mode` properties of the `options` object used to load the wizard plugin.
For example,

The amend page allows users to :

- Change pre-populated inputs and /or provide new inputs
- Navigate back to the query result page using Back button - This back button is not shown in the entry screen
- Submit the form using Submit button
- Reset the form using Reset button

As mentioned above, there is a back button in the amend page. The hook inherited from the wizard template to support this action is :

- `back` - It allows the user to go back to the query result page.

For example,

For employee amend, on clicking the back button user navigates back to the employee query result screen.

The back handler, defined in `finch-wizard.js`, handles all back operations in the entry/amend flow as shown below. It applies the back logic depending on the `actionType` property of the `options` object used to load the wizard plugin

`finch-wizard.js` ↗ [Expand](#)

[source](#)

```

this.back = function() {
    // check if back is called from 1st mode
    if (self.modelIndex == 0) {
        // if called from 1st mode, then its a call from consolidated action
        var FINCH$Handler$backHandler = FINCH$Handler$function({
            get : {
                requestType : FINCH$Handler$default.requestType.asynchronous,
                target : '#content'
            },
            settings : {
                type : 'POST',
                beforeSend : function(request) {
                    request.setRequestHeader('Accept',
                        'text/html;type=ajax');
                }
            }
        });
        var url = FINCH.context.path;
        var queryCommandFormId = self.$context.find('#queryCommandFormId')
            .val();
        var resultScreenURL = self.$context.find('#resultScreenUrl').val();
        var resultSortAsc = self.$context.find('#resultSortAsc').val();
        var resultSortCol = self.$context.find('#resultSortCol').val();
        var baseQueryURL = resultScreenURL.substring(0, resultScreenURL
            .lastIndexOf("/"));
        var commandFormDetail = {resultSortAsc: resultSortAsc, resultSortCol : resultSortCol };
        var backToResultURL = "/backToCurrentResult?commandFormId="
            + queryCommandFormId + "&fragments=content";
        if (resultScreenURL.length > 0 && queryCommandFormId.length > 0) {
            url = url + baseQueryURL + backToResultURL;
            FINCH$Handler$backHandler.generic(undefined, {
                requestUri : url,
                settings: {data : commandFormDetail},
                onHtmlContent : function(e, options, $target, content) {
                    $target.html(content);
                }
            });
        } else {
            FINCH.postNotice(FINCH.notice.type.error,
                'This feature is not supported !!', true);
        }
    } else {
        if(self.skipWizard === true) {
            self.skipPreviousMode('back');
        } else {
            // asynchronous handler to handle success action
            self.action('back', {
                success : function() {
                    if (self.modelIndex > 0) {
                        self.modelIndex -= 1;
                        self.pageIndex = self.pageIndexes[self.modelIndex];
                    }
                }
            });
        }
    }
};

```

See the [Single Page Entry - Usage of Tiles](#), [Create new entry page](#) and [Initialize new entry page sections for view details common to single page entry and amend flow](#).

4.2.4 Create user confirmation page for amend

As mentioned above, after successful submission of the amend form, user navigates to the user confirmation page.

Developer does not need to create a new view for amend user confirmation page. The view used for entry user confirmation page is same as the one used for amend user confirmation page.

See the [Single Page Entry - Create user confirmation for new entry section for further details](#).

4.2.5 Create system confirmation page for amend

After successful submission of the user confirmation page, user navigates to the system confirmation page.

See the [Single Page Entry - Create system confirmation for new entry section for further details](#).

The system confirmation page for amend differs from entry system confirmation page in allowing the user to go back to the query result screen using the Ok button.

For example,

In the employee amend flow, after successful form submission, user navigates to the system confirmation page where the confirmation message is shown.

On clicking the Ok button here, user goes back to the employee query result page.

The ok handler, defined in finch-wizard.js, handles all exit operations from the system confirmation page as shown below. It applies the exit logic depending on the actionType property of the options object used to load the wizard plugin.

finch-wizard.js

› [Expand](#)

```
this.ok = function() {
    // hooks
    if (self.invoke('ok') == false) {
        self.renderer.update();
        return;
    }
    self.invoke('unload');
    if (self.options.exitUri) {
        var url = self.options.exitUri;
        if (url.search('exitURI') != -1) {
            var queryCommandFormId = self.$context.find(
                '#queryCommandFormId').val();
            if (queryCommandFormId !== 'undefined'
                && queryCommandFormId.length > 0)
                if (url.indexOf('?') != -1)
                    url = url + '&commandFormId=' + queryCommandFormId;
                else
                    url = url + '?commandFormId=' + queryCommandFormId;
        }
        FINCH$Handler$asynchronous.generic(undefined, {
            requestUri : url
        });
        self.options.exitUri = undefined;
    } else {
        if (self.source === 'wizard-query-page') {
            url = FINCH.context.path + self.options.ok
            + "/ok/result?commandFormId="
            + self.$context.find('input[name=commandFormId]').val();
            FINCH$Handler$asynchronous.generic(undefined, {
                requestUri : url
            });
        } else {
            // no-op
        }
    }
};
```

[source](#)

4.4 Create the controller

4.3.1 Follow Inheritance rule

In finch, a Controller should extend base class com.nrft.finч.inf.web.controller.FinchAbstractEditWizardController.java. The annotation '@Controller' is being used to consider a simple class as a Controller.

Create a logger object in the controller. See proper usage of logger in [Logging](#).

For entry & amend single controller can be reused.

The following Controller has been added for Employee Amend:

EmployeeEntryController.java

public class EmployeeAmendController extends FinchAbstractEditWizardController<EmployeeAmendCommandForm> implements MessageSourceAware {
 private static final Logger log = LoggerFactory.getLogger(EmployeeAmendController.class);
 ...
 }

› [Expand](#)

[source](#)

Inside the controller, developer has to implement the following methods to complete the Amend operations:

FinchAbstractEditWizardController

```
//This method will contain screen specific initialization logic.  

doInit(C commandForm)  

//This method will contain validation logic.  

doValidate(C commandForm, int index)  

//This method will contain page reinitialize logic.  

doReset(C commandForm, int index)  

//This method provide a hook to implement logic(s) , that will be applicable during the navigation from entry page to user confirmation page.  

doSubmit(C commandForm, int index)  

//This method will contain the logic related to data persistence and message generation.  

doConfirm(C commandForm)
```

› [Expand](#)

[source](#)

4.3.2 Provide request URL mapping

The developer should specify the request mapping at the class level. A URL mapping should be specified for each and every controller in Finch. The annotation '@RequestMapping' is being used to set the request URL. The URL should follow the Action URL Pattern specified (in database) for the Menu item clicked. By using the annotation @SessionAttribute the CommandForm has been specified as a session attribute. The following mapping has been used for Employee Amend:

EmployeeAmendController.java

```
@Controller  

@SessionAttributes("commandForm")  

@RequestMapping(value = "/inf/employee/amend/**")
```

› [Expand](#)

[source](#)

4.3.3 Check the requested action is to be performed or not

When the user clicks on the consolidation icon (Amend / Enable) in the consolidation page at first this need to be checked that whether the particular record is eligible for the selected action or not.(Ex : user can't perform a amend action on already closed record).

This validation is done based on the boolean status of isActionable flag . Developer have to check the the above mentioned condition explicitly.

EmployeeAmendController

› Expand

```
@RequestMapping(value = "/actionable", method = { RequestMethod.GET,
    RequestMethod.POST })
public void amendment(@RequestParam("pk") long employeePk,
    @RequestParam("type") String actionType, ModelMap map)
    throws FinchException {
    // Validating action execution
    boolean isActionable = false;
    if (Objects.equal(actionType.toLowerCase(),
        Globals.CANCEL_STATUS.toLowerCase())
        || Objects.equal(actionType.toLowerCase(),
            Globals.AMEND_STATUS.toLowerCase())) {
        isActionable = true;
    }
    try {
        EmployeeService empService = getService(EmployeeService.class);
        Employee employee = empService.findEmployeeByPk(employeePk);
        if (Objects.equal(employee.getStatus().toLowerCase(),
            Globals.CANCEL_STATUS.toLowerCase())) {
            isActionable = false;
        } else {
            Identity identity = Operation.getInstance().getContext()
                .getCallerIdentity();
            if ((identity != null)
                && (identity.getEmployeePk() == employeePk)) {
                isActionable = false;
            }
        }
    } catch (FinchException e) {
        log.error("Failed to find Employee of employeePk = ["
            + employeePk + "]", e);
        throw new FinchException("", e);
    }
}
```

[source](#)

4.3.4 Define screen specific initialization logic

When the user clicks on the consolidation icon (Amend / Enable) the next screen need to be initialize with detail information of the target record. The initialization have been done in performAction() method of the controller which is being called by clicking on the Amend consolidation icon .

Basically, the initialization should include:

- Fetch the detail information from the database for the target record (The record in the query result screen for which the action (amend / close / reopen) will be performed)
- Prepare the CommandForm/screen from the detail information fetched from the database.
- Populating the possible values for the drop downs in the screen. Most of the time the values are being populated using the Constraints.

EmployeeAmendController.java

› Expand

```
@RequestMapping(value = "/performAction", method = { RequestMethod.GET,
    RequestMethod.POST })
public String performAction(@RequestParam("pk") long employeePk,
    @RequestParam("screenId") String screenId, ModelMap map)
    throws FinchException {
    // Creating an instance of the CommandForm
    EmployeeAmendCommandForm form = new EmployeeAmendCommandForm();
    // Getting an instance of the Employee Service
    EmployeeService empService = getService(EmployeeService.class);
    EmployeeDTO employeeDTO = empService.populateEmployeeDTO(employeePk);
    form.setEmployeeDTO(employeeDTO);
    // setting password and confirm password field blank
    // as in amend screen these fields value will not be displayed.
    form.getEmployeeDTO().setPassword(StringUtils.EMPTY);
    form.getEmployeeDTO().setConfirmPassword(StringUtils.EMPTY);
    // original employee data is stored in OriginalEmployeeDTO field
    form.setOriginalEmployeeDTO(empService.populateEmployeeDTO(employeePk));
    form.setApplPasswd(employeeDTO.getConfirmPassword());
    form.setEmployeePk(employeeDTO.getEmployeePk());
    form.setDefaultBranch(employeeDTO.getDefaultBranch());
    form.setDefaultBranchName(employeeDTO.getDefaultBranch().getBranchId());
    form.setUserId(employeeDTO.getUserId());
    Employee emp = new Employee();
    emp.setUserId(employeeDTO.getUserId());
    emp.setEmployeePk(employeePk);
    form.setEmployee(emp);
    form.setFirstName(employeeDTO.getFirstName());
    form.setMiddleInitial(employeeDTO.getMiddleInitial());
    form.setLastName(employeeDTO.getLastName());
    form.setEmployeeCloseDate(employeeDTO.getEmployeeCloseDate());
    form.setEmployeeClosedBy(employeeDTO.getEmployeeClosedBy());
    form.setStartDate(employeeDTO.getStartDate());
    form.setEndDate(employeeDTO.getEndDate());
    form.setEmployeeOpenDate(employeeDTO.getEmployeeOpenDate());
    form.setEmployeeOpenedBy(employeeDTO.getEmployeeOpenedBy());
    form.setEmployeeId(employeeDTO.getUserId());
    form.setMail(employeeDTO.getMail());
    form.setUserApplRoleWithBranchList(employeeDTO
        .getUserApplRoleWithBranchList());
    form.setScreenId(screenId);
    return init(map, form);
}
```

[source](#)

4.3.5 Define server side validation logic

When user submits an amend form after specifying values to the fields, the form needs to be validated first. This validation is being done in `doValidate()` method .The base class `FinchAbstractEditWizardController.java` has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality. The developer only has to implement the method in the controller, he doesn't need to call it explicitly. This method has been called implicitly by the `submit()` method in base class `FinchAbstractEditWizardController.java`.

FinchAbstractEditWizardController.java

[Expand](#)

```
@RequestMapping(value = "/submit", method = RequestMethod.POST) source
public String submit(ModelMap map, @ModelAttribute("commandForm") C commandForm,
    @RequestParam(value = "validate", required = false) boolean validate) {
    FinchModelMap modelMap = new FinchModelMap(map);
    modelMap.putToModelMap("commandForm", commandForm);
    String module = commandForm.getModule();
    FinchWizard wizard = commandForm.getWizard();
    FinchWizardMode mode = wizard.currentMode();
    commandForm.setValidateThisPage(null);
    boolean error = false;
    try {
        preSubmit(modelMap, commandForm, validate);
        validate = (commandForm.getValidateThisPage() != null) ? commandForm.getValidateThisPage() :
validate;
        if (validate) {
            // validate
            doValidate(commandForm, wizard.getPageIndex());
        }
        doSubmit(commandForm, wizard.getPageIndex());
        // switch to next mode
        if (!wizard.isOnLastMode()) {
            if (commandForm.isSkipNextMode())
                wizard = skipNextWizardMode(commandForm);
            mode = wizard.nextMode();
        }
        setScreenName(commandForm);
        // Access Log
        logUserAccessInfo(getScreenId(commandForm));
    } catch (Exception e) {
        error = true;
        log.error("Failed to submit wizard page", e);
        modelMap.addError(e, Component.getComponent(module).get(), getDefaultErrorKey(),
            ArrayUtils.EMPTY_STRING_ARRAY);
    }
    FinchWizardPage pageN = error ? wizard.currentPage() : wizard.nthPage(0);
    return pageN.getTitle(mode);
}

protected void doValidate(C commandForm, int index) throws FinchException {
    // no-op;
}
```

EmployeeAmendController

› Expand

```
@Override  
protected void doValidate(EmployeeAmendCommandForm form, int index)  
    throws FinchException {  
    form.setInFirstScreen(Boolean.FALSE);  
    form.setMode(Constants.AMENDMENT);  
    form.setEmployeeMode(Globals.AMEND_STATUS);  
    form.setInFirstScreen(Boolean.FALSE);  
    switch (index) {  
        case EmployeeAmendCommandForm.GENERAL_TAB:  
            validatePage1(form);  
            break;  
        case EmployeeAmendCommandForm.DETAILS_TAB:  
            // Validating the last page upon form submission  
            break;  
        default:  
            break;  
    }  
}
```

[source](#)

4.3.6 Define form submit and user confirmation logic

The submission of an amend form is being handled by the method submit() of the base class FinchAbstractEditWizardController.java. This method includes:

- Validation of the values specified in the entry form.
- (If required) Population of few values according to the functionality from the specified values in the entry form.
- Finally, returning the tile name of the JSPX for User Confirmation screen.

FinchAbstractEditWizardController.java

› Expand

```
@RequestMapping(value = "/submit", method = RequestMethod.POST) source
public String submit(ModelMap map, @ModelAttribute("commandForm") C commandForm,
    @RequestParam(value = "validate", required = false) boolean validate) {
    FinchModelMap modelMap = new FinchModelMap(map);
    modelMap.putToModelMap("commandForm", commandForm);
    String module = commandForm.getModule();
    FinchWizard wizard = commandForm.getWizard();
    FinchWizardMode mode = wizard.currentMode();
    commandForm.setValidateThisPage(null);
    boolean error = false;
    try {
        preSubmit(modelMap, commandForm, validate);
        validate = (commandForm.getValidateThisPage() != null) ? commandForm.getValidateThisPage() :
validate;
        if (validate) {
            // validate
            doValidate(commandForm, wizard.getPageIndex());
        }
        doSubmit(commandForm, wizard.getPageIndex());
        // switch to next mode
        if (!wizard.isOnLastMode()) {
            if (commandForm.isSkipNextMode())
                wizard = skipNextWizardMode(commandForm);
            mode = wizard.nextMode();
        }
        setScreenName(commandForm);
        // Access Log
        logUserAccessInfo(getScreenId(commandForm));
    } catch (Exception e) {
        error = true;
        log.error("Failed to submit wizard page", e);
        modelMap.addError(e, Component.getComponent(module).get(), getDefaultErrorKey(),
            ArrayUtils.EMPTY_STRING_ARRAY);
    }
    FinchWizardPage pageN = error ? wizard.currentPage() : wizard.nthPage(0);
    return pageN.getTitle(mode);
}

protected void doValidate(C commandForm, int index) throws FinchException {
    // no-op;
}
```

4.3.7 Define system confirmation and data persistence logic

The confirmation of an amend page is being handled by the method `doConfirm()` of the base class `FinchAbstractEditWizardController.java`. The base class `FinchAbstractEditWizardController.java` has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality.

FinchAbstractEditWizardController

```
/**  
 * Does the confirmation. Implementation should override this.  
 *  
 * @param connection  
 *      connection  
 * @param queueHandler  
 *      queue handler  
 * @param commandForm  
 *      command form  
 * @throws FinchException  
 *      in case of any error  
 */  
// TODO : need to handle message Queue  
protected void doConfirm(C commandForm) throws FinchException {  
    // no-op;  
}
```

EmployeeAmendController.java

› Expand

```
@Override
protected void doConfirm(EmployeeAmendCommandForm form)
    throws FinchException {
// Updating Employee
EmployeeService employeeService = getService(EmployeeService.class);
// Getting original Employee PK
long originalEmployeePk = form.getEmployeePk();
EmployeeDTO empAmend = form.getEmployeeDTO();
if (form.getEmployeeDTO().getResetPasswordFlag() == null) {
    empAmend.setPassword(form.getOriginalEmployeeDTO().getPassword());
    empAmend.setConfirmPassword(form.getOriginalEmployeeDTO()
        .getConfirmPassword());
}
// Employee Amend action commencement
employeeService.updateEmployee(originalEmployeePk, empAmend);
form.setEmployeeDTO(employeeService.populateEmployeeDTO(form
    .getEmployeeDTO().getEmployeePk()));
FinchWizard wizard = form.getWizard();
if (!wizard.isOnLastMode())
    wizard.nextMode();
// Setting user mode as confirmed. This mode is checked at the client
// side for displaying the success message
form.setMode(Constants.CONFIRMED);
// Getting Locale from User preference
Locale userLocale = new Locale(Operation.getInstance().getContext()
    .getCallerIdentity().getPreferenceBean()
    .getPreference(PreferenceConstants.LOCALE));
// Array of parameters for message source
Object[] param = new Object[] { form.getEmployee().getUserId() };
// Creating success message for Employee Entry action
String successMessage = messageSource.getMessage(
    "inf.employee.amend.action.success", param, userLocale);
// Setting the Success message to the form
form.setSuccessMessage(successMessage);
}
```

source

4.3.7.1 Concurrency handling

Proper concurrency handling is the important part of amend / cancel / operation.

See [Handle of Concurrent operation - 3.5 Steps to implement Locking for UI](#) for details.

4.3.8 Transaction Management

Transaction has been properly managed by the base class. The developer has nothing to do with this.

For details of Transaction Management see [Transaction Management](#).

4.3.12 Access Log

In finch, whenever a Single Page Amend screen is being added, the Access Log implementation should be done properly. From this Access Log, we can get history of accessing of the screen.

For Access Log of a Single Page Amend, the developer has to only use the standard URL pattern for amend operation. The rest has been handled by the framework. Standard URL pattern for an operation is like "secure/<module_in_small>/<entity_in_small>/<operation_in_sm all>/*", where ref, trd, stl are the modules; account, customer, trade, settlement are the entities; entry, amend, query are the operations. For example, the URL for Strategy amend should be like 'secure/ref/strategy/amend/actionable.json'.

If any URL does not follow the standard pattern, then one has to configure that pattern in database tables for the Access Log implementation. For details of

this setting see [Access Log](#).

5. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

6. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	23-02-2015	Initial Version	Rahul c

3.3.4.4 Wizard Based Entry

1. Scope

In finch application, some entities are created in multiple steps. The entities like Trade, Accounts etc., which have a large number of fields associated with them, are ideal candidates for wizard based entry . The table used to store the data for such entities are generally linked to a number of child tables. In order to gather all information related to these entities, the input fields are logically grouped under several tabs rather than including all of them in a single page. To create such an entity like, Trade, Application Role, and capture all their required information through UI, the wizard based entry model is used. The wizard based entry screen is followed by the user confirmation and system confirmation screens. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation details for wizard based entry. It outlines all the necessary steps, both on the client side and server side, to create a new entity using wizard based entry model.

2. Prerequisites

MUST SEE

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception Handling](#) for proper handling of exceptions.

Refer to [Single Page Entry](#) for the details on how to create a single page entry screen.

Refer to [Handle of Concurrent operation](#) for the details on the features of concurrency handling in finch.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

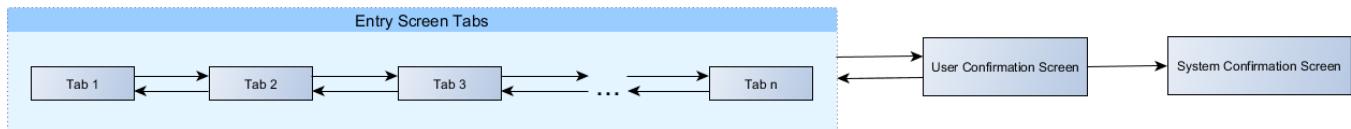
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The screen flow for wizard based entry is shown below :



In wizard based entry, user can :

- Navigate back and forth between the tabs.
- Submit the entry form for user confirmation from any tab.
- Navigate back from the user confirmation screen to that entry screen tab from which the form was submitted.
- Finally submit the entry form for system confirmation.

Screen	Purpose
Entry screen spanning multiple tabs	It is used to provide input in the entry form fields, present in multiple tabs, for creating a particular entity. It also allows user to navigate back and forth between the tabs and submit the entry form for user confirmation.
User Confirmation spanning multiple tabs	It is used to preview and confirm the input data before proceeding further for system confirmation. If the input, provided in any of the tabs, needs modification then user can go back to that particular tab of entry screen.
System Confirmation spanning multiple tabs	It confirms that the entry data has been successfully saved in the database. It shows the system confirmation message along with the reference number, wherever applicable, for the completed transaction.

2.3 UI components

The entry page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing entry views, should be retained to get the basic layout, look & feel and functionality of finch entry forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side. The form input data validations are handled on the client side and the business validations are handled on the server side. See the [Exception Management](#) document for more details on error handling.

2.7 Menu creation & Action URL patterns

A predefined and uniform URL pattern is essential in a web application.

In finch , all the controller actions are mapped to a unique URL.This URL follows a strict pattern to ensure cleaner code and secure access. See the [Add new Menu/Action](#) document for more details on URL pattern conventions and menu creation.

Please note that we have used the Trade Entry use case as a reference throughout this document, for all the explanations.

3.New and Modified File List

Layer	Package/Location (sample)	File type/ purpose	File name (sample)	New/Modified/
Model	/sample-trd/src/main/java/com/nrft/sample/trd/web/form/	Command Form (i.e. POJO) class	TradeEntryCommandForm.java	New
View	/sample-trd/src/main/web/WEB-INF/views/trd/	Module specific xml file to provide view tile definition	views.xml	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for en locale	sample-trd-i18n_en.js	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for ja locale	sample-trd-i18n_ja.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for en locale	finch-i18n_en.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for ja locale	finch-i18n_ja.js	Modify
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade general entry view	tradeGeneralEntry.jspx	New
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade Detailed entry view	tradeDetailsEntry.jspx	New
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade details view	tradeGeneralDetail.jspx	New
	/sample-trd/src/main/web/scripts/trd/	js file for handling query result consolidation by setting consolidation parameters	sample-trd-query-result.js	New
	/sample-trd/src/main/web/scripts/trd/	js file for including a new formatter for the query result grid column	sample-trd-formatters.js	Modify

Controller	/sample-trd/src/main/java/com/nrift/sample/trd/web/controller/	Controller class, contents all the action APIs for Trade Entry	TradeEntryController.java	New
Service	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Interface that provides method related to Trade	TradeService.java	New
	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Implementation for TradeService Interface	TradeServiceImpl.java	New
Repository	/sample-trd/src/main/java/com/nrift/sample/trd/domain/repository	contains all the necessary methods to perform CRUD operation	TradeRepository.java	New

4. Sequence of Steps

4.1 Create the model

4.1.1 Create Tables and Sequences for the new entity

One needs to create the main and the associate tables for saving the information of the new entity. Information of most of the entities in finch have been stored through multiple tables. The following tables have been used to store information of Trade in finch:

- TRD_TRADE
- TRD_TRADE_TAX_FEE_AMOUNT
- TRD_TRADE_RR

See [Single Page Entry - 4.1.1 Create Tables and Sequences for the new entity](#) section for more details.

4.1.1 CRUD Generation Tool

See the [Single Page Entry - 4.1.2 CRUD Generation Tool](#) section for CRUD Generation Tool details.

4.2.2 Create Command Form

For sending and receiving the Form/Request values between client and server side, finch uses POJO classes i.e. CommandForms. A CommandForm contains all the field names (JSP field names used in the screen) along with their getter() and setter() method. One should be very careful about the data types of the member variables of a CommandForm. If there exists any mismatch between the data types of the variables of CommandForm and the variables used in the screen, it'll lead to an exception.

In finch , a CommandForm for a single page entry should always extend the base class com.nrift.finч.inf.web.FinchAbstractWizardCommandForm. For Employee entry, the TradeCancelCommandForm.java has been defined in the package "com.nrift.sample.trd.web.form".

A CommandForm should always have a method which'll reset the member variables like the following:

TradeCancelCommandForm.java

› Expand

[source](#)

```
public class TradeCancelCommandForm extends FinchAbstractWizardCommandForm {  
    public static final int GENERAL_TAB = 0;  
    public static final int DETAILS_TAB = 1;  
    // hidden fields  
    private String mode = null;  
    private String message = "";  
  
    private String method = null;  
    private long tradePk;  
  
    ..  
    public String getSuccessMessage(){  
        return message;  
    }  
  
    public void setSuccessMessage(String message){  
        this.message = message ;  
    }  
  
    /**  
     * Current Trade Screen Data  
     */  
    public TradeDTO getTradeDTO() {  
        return tradeDTO;  
    }  
  
    ..  
    public void reset() {  
        reset(GENERAL_TAB);  
        reset(DETAILS_TAB);  
    }  
  
    public void reset(int pageIndex){  
        switch (pageIndex) {  
        case GENERAL_TAB:  
            break;  
        case DETAILS_TAB:  
            break;  
  
        default:  
            break;  
        }  
    }  
    ..  
}
```

To enable character set encoding validation use annotation `@CharsetEncoding`. For details of this implementation please see [Character set Validation](#)

4.2 Create the view

4.2.1 Usage of Tiles

Wizard based entry has multiple number of tabs. The tab title, edit tile name and view tile name for each of these tabs is obtained from the `getPages()` method defined in the corresponding wizard entry controller. See [Create the Controller](#) section below for further details.

For each of these tabs there exists :

- An editable page for entry
- A non-editable page for user and system confirmation

So in order to create the layout for any wizard based entry, developer needs to provide the tile definitions of all its corresponding views in views.xml.

For example, in trade entry, developer needs to provide tile definitions for four views :

- Editable trade general entry tab
- Editable trade details entry tab
- Non-editable trade general tab
- Non-editable trade details tab

The view corresponding to trade entry needs to be defined in views.xml as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/views.xml > Expand source
<tiles-definitions>
  <!-- Trade General entry page -->
  <definition name="tradeGeneralEntry" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeGeneralEntry.jspx"/>
  </definition>

  <!-- Trade General detail page -->
  <definition name="tradeGeneralDetail" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeGeneralDetail.jspx"/>
  </definition>

  <!-- Trade view entry page -->
  <definition name="tradeViewEntry" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeViewEntry.jspx"/>
  </definition>

  <!-- Trade view detail page -->
  <definition name="tradeViewDetail" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeViewDetail.jspx"/>
  </definition>

  <!-- Trade Details entry page -->
  <definition name="tradeDetailsEntry" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeDetailsEntry.jspx"/>
  </definition>

  <!-- Trade detail page -->
  <definition name="tradeDetailsParticular" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeDetailsParticular.jspx"/>
  </definition>
</tiles-definitions>
```

All the new views need to extend "wizard-page". By doing so, the new views inherit all the common functionality and markup from "wizardPage.jspx" & "wizardContainer.jspx".

Trade entry wizard inherits the following features/elements from its wizard template :

- Form tag and all its attributes
- Header content
- Form action area containing all the buttons
- Breadcrumb section

Breadcrumb section

The placeholder for the breadcrumb section is defined in "wizard-container.jspx" as shown below :

/finch-web-infra/src/main/web/WEB-INF/views/inf/wizardContainer.jspx › [Expand](#)

```
<div id="carousel" class="es-carousel-wrapper" style="display: none">
<div id="wizardNavigation" class="es-carousel">
<ul id="wizStep">
<spring:message text="" htmlEscape="false" />
<c:forEach items="${commandForm.wizard.pages}" var="page"
varStatus="row">
<spring:message code="${page.title}" htmlEscape="false"
var="page_title" />
<li pageId="page${row.index}"><a href="#" title="${page_title}" pageIndex="page${row.index}"><span
class="counter">${row.index + 1}</span><span class="wizText"><spring:message
code="${page.title}" htmlEscape="false" /></span></a></li>
</c:forEach>
</ul>
</div>
</div>
```

[source](#)

Breadcrumb section is displayed for wizard based entry depending on the length of the "pages" variable. Its value is obtained from the commandForm and is used to create a new instance of the wizard plugin in "wizardContainer.jspx". The markup for the individual items within the breadcrumb section is defined in finch-wizard.js as shown below :

finch-wizard.js

› Expand

[source](#)

```
this.doRender = function(action) {  
    ..  
    $navigation.find('a').off('click');  
    jQuery.each(self.object.pages, function(index, page) {  
        if (mode == 'EDIT') {  
            if (pageHistory[pageIndex]) {  
                $navigation.find('[pageId=page' + index + ']').addClass('active');  
                $navigation.find('[pageId=page' + index + ']').find('a').on('click', self.switchTo);  
            }  
        } else {  
            $navigation.find('[pageId=page' + index + ']').removeClass('active');  
            $navigation.find('[pageId=page' + index + ']').find('a').on('click', self.switchTo);  
        }  
        if (pageIndex == index) {  
            $navigation.find('[pageId=page' + index + ']').addClass('current');  
            $navigation.find('[pageId=page' + index + ']').find('a').off('click');  
        } else {  
            $navigation.find('[pageId=page' + index + ']').removeClass('current');  
        }  
    });  
    ..  
    if (pageIndex == 0)  
        self.hideActions([ 'previous' ]);  
    if (pageIndex == (self.object.pages.length - 1))  
        self.hideActions([ 'next' ]);  
    self.modelIndex = modelIndex;  
    // fix input element focus  
    if(self.object.$context.find(".formTabErrorIco").is(":hidden") || FINCH.focusFirstField === true) {  
        if (self.object.source != 'wizard-query-page')  
            FINCH.focusFirst('#' + self.object.source);  
    }  
    self.object.pageComplete();  
    ..  
}
```

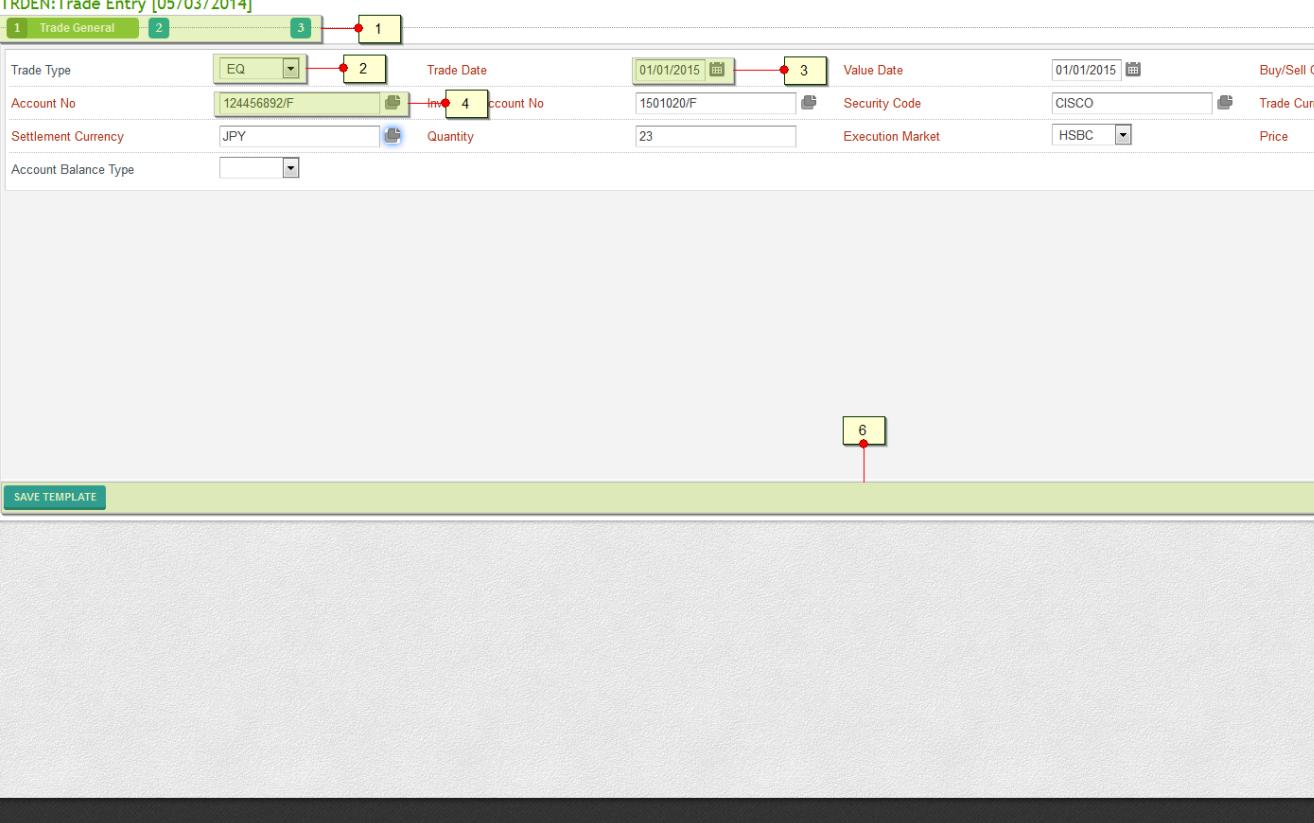
See the [View templates - Screen Composition](#) section to get further details.

4.2.2 Basic components of wizard based entry screen

[Entry Page Components -Tab1](#) [Entry Page Components -Tab2](#) [Entry User Confirmation](#) [Entry System Confirmation](#)

Search Menu 

TRDEN:Trade Entry [05/03/2014]



1 Trade General 2 3 1

Trade Type: EQ Trade Date: 01/01/2015 Value Date: 01/01/2015 Buy/Sell C

Account No: 124456892/F Inv: 4 Account No: 1501020/F Security Code: CISCO Trade Curr:

Settlement Currency: JPY Quantity: 23 Execution Market: HSBC Price:

Account Balance Type:

SAVE TEMPLATE

1. Breadcrumb section - See the Create the Controller and Usage of tiles section for further details.
2. Drop-down input control - See [Form Elements](#) - Drop-down input control section for further details.
3. Date Picker control - See [Form Elements](#) - Date Picker control section for further details.
4. Account Popup - See [Use a pop-up](#) - Account popup section for further details.
5. Currency popup - See [Use a pop-up](#) - Currency popup section for further details.
6. Action area of entry form - See the Initialize new entry page section below for details.

Search Menu 

TRDEN:Trade Entry [05/03/2014]

1 **2** **3 Trade Details**

Trade Type	EQ	Trade Date	01/01/2015	Value Date
Buy/Sell Orientation	AB	Account No	124456892/F	Inventory Account No
Security Code	CISCO	Quantity	23	Price
Execution Market	B001	Trade Currency	JPY	Settlement Currency
Account Balance Type				

Tax Fee

 ID	Rate	Rate Type	Amount
 COUPON	50	CPS	

RR

RR Role	RR Number
Executing RR	12
RR	
Split RR1	
Split RR2	
Trader	

Net Amount  External Reference No

SAVE TEMPLATE

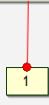
1. Detail block - This section contains the inputs from the first tab of trade entry in the non-editable format. See Create new entry page section below and [Detail Page Layout](#) for further details.
2. Editable Grid section mentioned above see [Grid Component](#) for further details.

Search Menu 

TRDEN:Trade Entry [05/03/2014]

1 Trade General 2 Trade View 3 Trade Details

Trade Type	EQ	Inventory Account No	1501020	Inventory A/C Short Name	Account B
Account No	124456892	Short Name		Security Code	CISCO
Price	12233	Trade Date	01/01/2015	Value Date	01/01/2015
Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY

 1

 2

1. Trade general detail block - See [Detail Page Layout](#) for further details.
2. Action area of user confirmation page - See [Create user confirmation page](#) for new entry section below for further details.

TRDEN:Trade Entry [05/03/2014]

1 Trade General 2 Trade View 3 Trade Details

Trade Type	EQ	Inventory Account No	1501020	Inventory A/C Short Name		Account B
Account No	124456892	Short Name		Security Code	CISCO	Quantity
Price	12233	Trade Date	01/01/2015	Value Date	01/01/2015	Buy/Sell C
Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY	

1. System confirmation message - See Create system confirmation page for new entry section below for details.
2. Trade general detail block - See [Detail Page Layout](#) for further details.

4.2.3 Create new entry page

After providing the tile definition, the developer needs to create the new entry views "tradeGeneralEntry.jspx" and "tradeDetailsEntry.jspx".

The new entry views need to be placed in a specific folder. Its location will depend on the type of data it collects.

See Single Page Entry - Create new entry page section for the general convention that decides the location of a new jspx.

In the "tradeGeneralEntry.jspx" and "tradeDetailsEntry.jspx", developer needs to create the form elements using specific classes and ids.

See the [Entry/Amend Form](#) section to get the details about the markup needed to create an entry page.

Now that the markup is ready, developer needs to map the form attributes and element values to a data model. In finch application commandForms have been used for sending the form values from client to the server.

The variable names used in the commandForms should be identical to the names used in the form elements and attributes.

For example :

For trade type, the variable defined in TradeEntryCommandForm class is "tradeDTO.tradeType".

In "tradeGeneralEntry.jspx", the trade type element options should be mapped to this variable using "\${commandForm.tradeDTO.tradeType}" as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeGeneralEntry.jspx
```

» [Expand](#)

source

```
<div class="formItem">
  <form:label path="commandForm.tradeDateStr" class="required"><spring:message
    code="trd.tradeentryaction.label.tradedate" htmlEscape="false"/></form:label>
  <span><form:input id="tradeDate" path="commandForm.tradeDateStr" class="dateinput" /></span> <!--
COPY functionality has change functionality, yet not implemented -->
  <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
```

See the section on [Create Command Form](#) and [Create the Controller](#) for further details.

commandForm.uniqueId is an unique id assigned to each instance of an entry page. This unique id is required for multi-tab control.

4.2.4 Initialize new entry page

The new entry page has a life cycle with the following phases:

- data
- create
- init
- destroy

See [UI Core](#) for more details on finch plugin structure and related methods.

The new entry allows users to

- Provide inputs
- Submit the form using Submit button
- Reset the form using Reset button
- Navigate to the next tab using Next button
- Navigate to the previous tab using Previous button
- Navigate directly to a particular tab

The list of hooks, defined in finch-wizard.js, which is inherited from the wizard template to support the above mentioned actions are :

- submit - It submits the form and on successful submission navigates to user confirmation page of the entire wizard based entry.
- reset - It resets the form element values to the original state of the tab where the user clicked reset button.
- next - It navigates the user to the next tab. Internally it calls a moveTo() method which renders the next tab by incrementing the page cursor value by 1.
- previous - It navigates the user to the previous tab. Internally it calls a moveTo() method which renders the previous tab by decrementing the page cursor value by 1.

- switchTo - It navigates the user to the selected tab. Internally it calls a moveTo() method which renders the previous or next tab by decrementing or incrementing the page cursor value by 1.

The above mentioned functionality are available to every new entry page from its template.

However, in the individual entry page developer must implement FINCH\$Wizard\$OnPageLoad function for initializing the page with the required page specific logic :

tradeGeneralEntry.jspx
[Expand](#)

source

```

var FINCH$Wizard$OnPageLoad = function(){
    FINCH.loadLocalizedScript([
        {path: FINCH.context.path + '/scripts/ref/sample-ref-i18n.js', async: false}
    ]);
    FINCH.loadLocalizedScript([
        {path: FINCH.context.path + '/scripts/trd/sample-trd-i18n.js', async: false}
    ]);
    FINCH.loadScript(
        [
            {path: FINCH.context.path + '/scripts/inf/finch-exchange.js', async: false},
            {path: FINCH.context.path + '/scripts/inf/finch-util-validator.js', async: false},
            {path: FINCH.context.path + '/scripts/inf/finch-datevalidation.js', async: false},
            {path: FINCH.context.path + '/scripts/inf/finch-util-commons.js', async: false},
            {path: FINCH.context.path + '/scripts/inf/finch-popquery-form.js', async: false},
            {path: FINCH.context.path + '/scripts/inf/finch-form-utils.js', async: false}
        ], {success: function() {
            $('#input.numVal').numVal();
            $tradeEntry$context = $('#commandForm');
            $('.dateinput',$tradeEntry$context).finchdatepicker();
            // Security Handlers
            // var $security = $('#securityCode', $tradeEntry$context);
            // $security.unbind('change', securityCodeChangeHandler);
            // $security.bind('change',securityCodeChangeHandler);
            // Quantity Handlers
            var $quantity = $('#quantity', $tradeEntry$context);
            // $quantity.unbind('blur', formatQty);
            // $quantity.bind('blur',formatQty);
            // Price Handlers
            var $price = $('#price', $tradeEntry$context);
            // $price.unbind('blur', formatQty);
            // $price.bind('blur', formatQty);
            /**
             * Popup Button Configurations
             */
            function configurePopup($buttonObj, id, title, url, extUrlParams, tgt) {
                $buttonObj.die();
                $buttonObj.live('click', function(e){
                    var setting = {'id': id,
                        'title': title,
                        'url': url,
                        'extUrlParams': extUrlParams,
                        'tgt': tgt
                    };
                    openPopUpForm(e, setting);
                });
            }
            function configureMultiPopup($buttonObj, id, title, url, extUrlParams, tgt) {
                $buttonObj.die();
                $buttonObj.live('click', function(e){
                    var multiSelectHandler = function(settings, data) {

```

```

        }
        var setting = {'id': id,
                      'title': title,
                      'url': url,
                      'extUrlParams': extUrlParams,
                      'tgt': tgt,
                      'multiSelect' : true
                    };
        openPopUpForm(e, setting);
      });
    }
    // Account Popup Button
    var $accountPopupBtnObj = $('#accountPopupBtn');
    configurePopup($accountPopupBtnObj, 'accountNo', SAMPLE$REF$i18n.title.accPopup,
                  '/ref/invaccount/popup/query', prepareAccDependentParams($accountPopupBtnObj),
                  '#accountNo');

    // Inventory Popup Button
    var $invAccountPopupBtnObj = $('#invAccountPopupBtn');
    configurePopup($invAccountPopupBtnObj, 'invAccountNo', SAMPLE$REF$i18n.title.invAccPopup,
                  '/ref/invaccount/popup/query', prepareAccDependentParams($invAccountPopupBtnObj),
                  '#invAccountNo');

    // Security Button
    var $securityPopupBtnObj = $('#securityPopupBtn');
    configurePopup($securityPopupBtnObj, 'securityCode', SAMPLE$REF$i18n.title.instrPopup,
                  '/ref/security/popup/query', prepareInstrumentDependentParams($securityPopupBtnObj),
                  '#securityCode');

    // Trade Currency Popup
    var $trdCcyPopupButtonObj = $('#trdCcyPopupButton');
    configurePopup($trdCcyPopupButtonObj, 'tradeCcy', SAMPLE$REF$i18n.title.instrPopup,
                  '/ref/instrument/popup/query/ccy', "#tradeCcy");

    // Settlement Currency Popup
    var $stlCcyPopupButtonObj = $('#stlCcyPopupButton');
    configurePopup($stlCcyPopupButtonObj, 'settlementCcy', SAMPLE$REF$i18n.title.instrPopup,
                  '/ref/instrument/popup/query/ccy', "#settlementCcy");
  /**
   * Trigger events or show/hide components depending upon server side
   * values.
   */
  var tradeType = "<c:out value=\"${commandForm.tradeDTO.tradeType}\" />";
  if(tradeType){
    $('#tradeType',$tradeEntry$context).trigger('change');
  }
  var actionType = "<c:out value=\"${commandForm.tradeMode}\" />";
  if ($.trim(actionType) == "amend") {
    $('.amendReadOnlyItem', $tradeEntry$context).each(function(index, el) {
      if($(el).hasClass('dropdowninput')){
        $(el).prop('disabled',true);
      } else if($(el).hasClass('textBox') || $(el).hasClass('dateinput')) {
        $(el).prop('disabled',true);
      } else if($(el).hasClass('amendReadOnlyPopup')) {
        $(el).hide();
      } else {
        // do-nothing
      }
    })
  }
  var validateHook = function() {
    var valid = validateGeneralTab();
    if(valid){

```

```
        return true;
    } else {
        return false;
    }
};

var unloadHook = function(){
    // cleanup code
    $finch$wizard.deregister('next',validateHook);
    $finch$wizard.deregister('unload',unloadHook);
};

var $finch$wizard = jQuery('#' + '${commandForm.uniqueId}').parent().parent().data('FINCH$Wizard');
$finch$wizard.register('next',validateHook);
$finch$wizard.register('unload',unloadHook);
if(actionType == "cancel" || actionType == "amend")
    $finch$wizard.apply({exitUri: FINCH.context.path + '/trd/query/exitURI'});
else
    $finch$wizard.apply({exitUri: FINCH.context.path});
```

```
        }  
    );  
};
```

Within the FINCH\$Wizard\$OnPageLoad function, developer can do the following :

1. Load the locale specific sample-trd-i18n js file for supporting i18n. For example, in trade entry, the page specific key value pairs need to be included in the module specific i18n files - sample-trd-i18n_en.js and sample-trd-i18n_ja.js.

In "tradeGeneralEntry.jspx" and "tradeDetailsEntry.jspx", developer needs to load "sample-trd-i18n.js" as shown below :

tradeGeneralEntry.jspx

```
FINCH.loadLocalizedScript([  
    {path: FINCH.context.path + '/scripts/trd/sample-trd-i18n.js', async: false}  
]);
```

Refer to [Multilingual i18n Support](#) and [UI Core](#)for more details on i18n and FINCH.loadLocalizedScript method.

2. Define one or more hooks. There are some predefined actions like "previous", "next", "reset" and "submit". These hooks, defined in "tradeGeneralEntry.jspx", perform some entry page-specific logic like mandatory input data validations before the predefined actions as shown below :

tradeGeneralEntry.jspx

» [Expand](#)

[source](#)

```
var validateHook = function() {  
    var valid = validateGenaralTab();  
    if(valid){  
        return true;  
    } else {  
        return false;  
    }  
};
```

We can see from the code snippet above that for trade entry, a validateHook function is defined which in turn calls the validateGenaralTab function. validateGenaralTab returns true if all the validation conditions are satisfied. So, if variable "valid", which holds the return value of validateGenaralTab function, is true then validateHook returns true and its associated predefined action is executed. If validation fails then the corresponding validation message is shown and no further action is executed.

validateGenaralTab function validates the user input in the mandatory fields and pushes the corresponding error messages into the messageFieldMap object which implements the concept of focus first as shown below :

tradeGeneralEntry.jspx

» [Expand](#)

[source](#)

```
function validateGeneralTab(){
    //At first trim all text fields
    ..
    var validationMessages = [];
    var messageFieldMap = new Object();
    ...
    if(VALIDATOR.isNullValue(tradeDate)){
        messageFieldMap['tradeDateStr'] = SAMPLE$TRD$i18n.tradeEntry.generalinfo.tradedate_empty;
    } else {
        if(!isValidDate(tradeDate)) {
            tradeDateValidation = false;
        }
    }
};


```

- messageFieldMap Object is then pushed to validationMessages array . The error message display logic is defined in the validateGeneralTab() function of "tradeGeneralEntry.jspx" as shown below :

tradeGeneralEntry.jspx

» [Expand](#)

[source](#)

```
function validateGeneralTab(){
    ..
    if ($.isEmptyObject(messageFieldMap) !== true){
        validationMessages.push(messageFieldMap);
        FINCH.postNotice(FINCH.notice.type.error, validationMessages, true);
        isValid = false;
    } else {
        isValid = true;
    }
    ..
}


```

- Register the hooks.

tradeGeneralEntry.jspx

```
$finch$wizard.register('next',validateHook);
$finch$wizard.register('unload',unloadHook);
```

- In "tradeGeneralEntry.jspx" we register hooks for "next" and "unload" actions
- Deregister the registered hooks.

```
tradeGeneralEntry.jspx

var unloadHook = function(){
    // cleanup code
    $finch$wizard.deregister('next',validateHook);
    $finch$wizard.deregister('unload',unloadHook);
};
```

Deregistering the previously registered hooks.

6. Include the page specific event handlers and functions in a page specific js file and load it in the corresponding jspx. Such page specific event handlers include common request handler which can be used for all server communications required for this entry page, handlers for form elements like text input, select box, buttons etc.

For example, in trade entry the page specific actions are included in a trade entry specific js file "sample-trd-entry.js" and loaded into "tradeGeneralEntry.jspx" as shown below:

```
tradeGeneralEntry.jspx

<jsp:element name="script">
<jsp:attribute name="type">text/javascript</jsp:attribute>
<jsp:attribute name="src"><c:url value="/scripts/trd/sample-trd-entry.js"/></jsp:attribute>
<jsp:body></jsp:body>
</jsp:element>
```

sample-trd-entry.js contains page specific event handlers like the common request handler, trade type change handler as shown below :

sample-trd-entry.js Expand

source

```
/** A common request handler to be used server communication.
 */
var Trade$Handler$RequestHandler = FINCH$Handler$function({
    get: {
        contentType: 'json',
        requestType: FINCH$Handler$default.requestType.asynchronous
    },
    settings: {
        beforeSend: function(request) {
            request.setRequestHeader('Accept', 'text/html;type=ajax');
        },
        type: 'POST'
    }
});
```

The trade type change handler displays or hides some form components depending upon trade type.

sample-trd-entry.js

› Expand

```
function tradeTypeChangeHandler(e){  
    var value = $(this).val();  
    showWhenIssue('HIDE');  
    showDpRepo('HIDE');  
    swapingAccounts();  
    if(value=="WHEN_ISSUE"){  
        showWhenIssue('SHOW');  
    }  
}  
  
function showWhenIssue(param){  
    var $target = $('.whenissueindicatorItem', $tradeEntry$context);  
    showMbsTba('SHOW');  
    if(param=="SHOW") {  
        $target.show();  
    } else {  
        $target.hide();  
    }  
}
```

[source](#)

Provide the handlers for actions specific to that page. The TRADE\$Handler\$RequestHandler request handler mentioned above is used in "tradeDetailsEntry.jspx".

sample-trd-entry.js

› Expand

```
function recalculateHandler(e){  
    var requestUrl = FINCH.context.path + $('form').attr('action') + "/recalculation.json?commandFormId=" + $('[name=commandFormId]').val();  
    Trade$Handler$RequestHandler.generic(e, { requestUri: requestUrl,  
        settings: {data : $('#commandForm').serialize()},  
        onJsonContent : function(e, options, $target, content) {  
            if(content.success == true){  
                //do nothing  
            } else {  
                FINCH.postNotice(FINCH.notice.type.error, content.value, true);  
            }  
        }  
    });  
}
```

[source](#)

See [UI Core](#) document for more details on FINCH\$Handler\$function options .

4.2.5 Create user confirmation page for new entry

As mentioned above after successful submission of the entry form, user navigates to the user confirmation page.

So next the developer needs to create the markup for the non-editable versions of the entry page .

For example, for trade entry, developer needs to create a non-editable views - "tradeGeneralDetail.jspx" .

The markup required to create the non-editable user confirmation page is different from the editable entry pages.

See the section on [Detail Page Layout](#) to get the details about the markup needed to create the details page.

In the wizard based entry, the look & feel of the tabs in the non-editable user confirmation & system confirmation pages is controlled based on the mode property of the options object mapped to the commandForm values. See the Create the Model section for further details on options object.

"tabStepArea" and "wizStepArea" class gives the wizard entry tabs the non-editable and editable look & feel respectively.

Based on the mode value class is added and "wizStepArea" class is removed from the tab navigation container to give the tabs the non-editable look & feel as shown below:

finch-wizard.js

 [Expand](#)

[source](#)

```
if (self.modelIndex != modelIndex) {  
    var $tabNav = $navigation.find('#wizStep li a');  
    if (mode == 'EDIT') {  
        $navigation.removeClass('tabStepArea');  
        $navigation.addClass('wizStepArea');  
        $navigation.find('.wizText').removeClass('txtShow');  
        $stabNav.mouseover(function() {  
            $(this).find('.wizText').addClass('txtShow');  
        }).mouseout(function() {  
            $(this).find('.wizText').removeClass('txtShow');  
        });  
    } else {  
        $navigation.removeClass('wizStepArea');  
        $navigation.addClass('tabStepArea').addClass('detailsNav');  
        $navigation.find('.wizText').addClass('txtShow');  
        $stabNav.unbind('mouseover').unbind('mouseout');  
    }  
}
```

The wizard based user confirmation is similar to the single page entry user confirmation.

See the [Single Page Entry - Create user confirmation page for new entry](#) section for further details

4.2.6 Create system confirmation page for new entry

After successful submission of the user confirmation page, user navigates to the system confirmation page.

The markup created for user confirmation page is used for system confirmation page as well.

For example, for trade entry, "tradeGeneralDetail.jspx" is used for both user confirmation & system confirmation pages.

The wizard based system confirmation is similar to the single page entry system confirmation.

See the [Single Page Entry - Create user confirmation page for new entry](#) section for further details

4.3 Create the controller

4.3.1 Follow Inheritance rule

See [Single Page Entry - 4.3.1 Follow Inheritance rule](#) for details.

The following Controller has been added for Trade entry:

TradeEntryController

› Expand

```
public class TradeEntryController extends FinchAbstractEditWizardController<TradeEntryCommandForm>
implements
MessageSourceAware {
    private static final Logger log = LoggerFactory.getLogger(TradeEntryController.class);
    ..
}
```

[source](#)

4.3.2 Provide request URL mapping

See [Single Page Entry](#) - 4.3.2 Provide request URL mapping for details.

The following URL mapping has been added to controller for Trade entry:

TradeEntryController.java

› Expand

```
@Controller
@SessionAttributes("commandForm")
@RequestMapping(value = "/trd/entry/**")
public class TradeEntryController extends FinchAbstractEditWizardController<TradeEntryCommandForm>
implements
MessageSourceAware {
    private static final Logger log = LoggerFactory.getLogger(TradeEntryController.class);
    private MessageSource messageSource;
    ..
}
```

[source](#)

4.3.3 Define screen specific initialization logic

When user clicks on the menu Trade-->Entry the page needs to be initialized first. This initialization is being done in `dolnit()` method of a controller. The base class `FinchAbstractEditWizardController.java` has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality. The developer only has to implement the method in the controller, he doesn't need to call it explicitly.

Basically, the initialization should include:

- Initializing the member variables of CommandForm
- Setting some default values to the CommandForm/entry screen tabs
- Populating the possible values for the drop downs in the entry screen tabs. Most of the time the values are being populated using the Constraints. See [Handle Constraint List & Values](#).

TradeEntryController.java

Expand

```
@Override
protected void doInit(TradeEntryCommandForm commandForm) throws FinchException {
    try {
        log.debug("Initialising TradeEntryCommandForm ...");
        // Getting an instance of the UserConstraintService
        UserConstraintService bean = Operation.getInstance().getGlobalContext()
            .getBean(UserConstraintService.class);
        // Enterpriseld
        String enterpriseld = Operation.getInstance().getContext().getEnterpriseld();
        // Getting the Default TradeType from UserConstraintService
        String defaultTradeType = bean.getDefaultConstraintValue(TrdConstants.CONSTRAINT_TRADE_TYPE,
enterpriseld);
        // Adding a TradeDTO instance to the CommandForm
        // to be used in reset method
        commandForm.setOldTradeDTO(new TradeDTO());
        commandForm.getOldTradeDTO().setTradeType(defaultTradeType);
        if (!commandForm.isSaveTemplate()) {
            // CommandForm Reset
            commandForm.reset();
        }
        // Marking First Screen as current position
        commandForm.setInFirstScreen(true);
        // Adding a TradeDTO instance to the CommandForm
        if (!commandForm.isSaveTemplate()) {
            commandForm.setTradeDTO(new TradeDTO());
        }
        if (!commandForm.isSaveTemplate()) {
            // Setting the Default TradeType
            commandForm.getTradeDTO().setTradeType(defaultTradeType);
        }
        // Entry mode status is set to empty
        commandForm.setMode("");
        // TradeMode is set to Trade Entry
        commandForm.setTradeMode(TrdConstants.ENTRY);
        commandForm.setOriginalTaxFeeAmounts(commandForm.getTaxFeeAmounts());
        commandForm.setSkipWizard(true);
    } catch (Exception e) {
        log.error("Failed to make a view for the trade data", e);
        throw new TrdException("trd.tradeentryaction.error.populate", e);
    }
}
```

[source](#)

4.3.4 Define server side validation logic

When user submits / navigates (by clicking next/previous) an entry form after specifying values to the fields, the form needs to be validated first. This validation is being done in doValidate() method of a controller. The base class FinchAbstractWizardController.java has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality. The developer only has to implement the method in the controller, he doesn't need to call it explicitly.

During validation, normally one should check:

- Whether all the mandatory fields have been filled up or not
- Whether the specified values are valid or not (like in case of dates, static data etc.)
- Whether the specified values are valid or not from the business perspective

TradeEntryController.java

› Expand

```
@Override
protected void doValidate(TradeEntryCommandForm commandForm, int index) throws FinchException {
    // Marking that the Wizard is no longer in the First Screen
    commandForm.setInFirstScreen(false);
    // Validating the current Wizard page before Navigation
    switch (index) {
        case TradeEntryCommandForm.GENERAL_TAB:
            validatePage1(commandForm);
            break;
        case TradeEntryCommandForm.DETAILS_TAB:
            // Validating the last page upon form submission
            // if(commandForm.getMode().equals(TrdConstants.COMPLETE))
            // validatePage2(commandForm);
            break;
        default:
            break;
    }
}
```

[source](#)

4.3.5 Define form submit and user confirmation logic

See [Single Page Entry - 4.3.5 Define form submit and user confirmation logic](#) for details.

4.3.6 Transaction Management

Transaction has been properly managed by the base class. The developer has nothing to do with this.

For details of Transaction Management see [Transaction Management](#).

4.3.12 Access Log

In finch, whenever a whenever a Wizard Based Entry screen is being added the Access Log implementation should be done properly. From this Access Log, we can get history of accessing of the screen.

For Access Log of a Single Page Cancel, the developer has to only use the standard URL pattern for amend operation. The rest has been handled by the framework. Standard URL pattern for an operation is like "secure/<module_in_small>/<entity_in_small>/<operation_in_sm all>/*", where ref, trd, stl are the modules; account, customer, trade, settlement are the entities; entry, amend, query are the operations. For example, the URL for Strategy amend should be like 'secure/ref/marketprice/cancel/actionable.json'.

If any URL does not follow the standard pattern, then one has to configure that pattern in database tables for the Access Log implementation. For details of

this setting see [Access Log](#).

4.4 Create the Service

See [Single Page Entry - 4.5 Create the Service](#) for details.

4.5 Create the Repository

See [Single Page Entry - 4.5 Create the Repository](#) for details.

4. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

5. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	25-02-2015	Initial Version	Rahul c

3.3.4.5 Wizard Based Amend

1. Scope

In finch application, some entities are created in multiple steps. The entities like Trade, Accounts etc., which have a large number of fields associated with them, are ideal candidates for wizard based entry . The table used to store the data for such entities are generally linked to a number of child tables. In order to gather all information related to these entities, the input fields are logically grouped under several tabs rather than including all of them in a single page. To amend such an entity like, Trade, Application Role, Instrument, Customer etc., the wizard based amend model is used. The wizard based amend screen is followed by the user confirmation and system confirmation screens. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation for wizard based amend only. It outlines all the necessary steps, both on the client side and server side, to update an entity using wizard based amend model. As the amend screen is launched from query screen, some query related flow, relevant to amend, is also mentioned in this document.

2. Prerequisites

MUST SEE

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception Handling](#) for proper handling of exceptions.

Refer to [Wizard Based Entry](#) for the details on how to create a wizard based entry screen.

Refer to [Query/Query Result](#) for the details on the features and components of query screen.

Refer to [Handle of Concurrent operation](#) for the details on the features of concurrency handling in finch.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

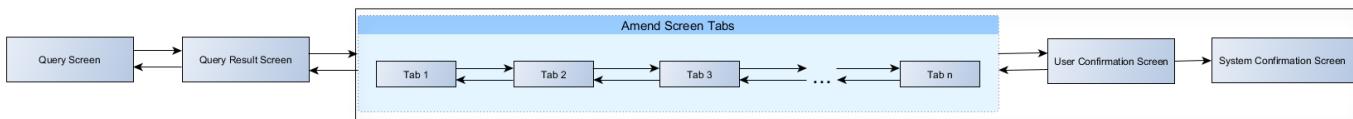
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The wizard based amend is launched from query screen. The complete screen flow for wizard based amend, with the target screens highlighted, is shown below :



Screen	Purpose
Query	It is used to query for a particular entity based on some or all of its associated fields.
Query Result	It is used to display the result obtained based on the query criteria. User can select a record from this screen and initiate amend sub-action on it.
Amend screen spanning multiple tabs	It is used to update/provide input in the form fields, present in multiple tabs, for amending a particular entity. It also allows user to submit the updated form for user confirmation.
User Confirmation spanning multiple tabs	It is used to preview and confirm the input data before proceeding further for system confirmation. If the input, provided in any of the tabs, needs modification then user can go back to that particular tab of amend screen.
System Confirmation spanning multiple tabs	It confirms that the updated data has been successfully saved in the database. It shows the system confirmation message along with the reference number, wherever applicable, for the completed transaction.

2.3 UI components

The Amend page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing entry views, should be retained to get the basic layout, look & feel and functionality of finch entry forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side.

The form input data validations are handled on the client side and the business validations are handled on the server side.

See the [Exception Management](#) document for more details on error handling.

Please note that we have used the Trade Amend use case as a reference throughout this document, for all the explanations.

3. New and Modified File List

Layer	Package/Location (sample)	File type/ purpose	File name (sample)	New/Modified
Model	/sample-trd/src/main/java/com/nrift/sample/trd/web/form/	Command Form (i.e. POJO) class	TradeAmendCommandForm.java	New
View	/sample-trd/src/main/web/WEB-INF/views/trd/	Module specific xml file to provide view tile definition	views.xml	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for en locale	sample-trd-i18n_en.js	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for ja locale	sample-trd-i18n_ja.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for en locale	finch-i18n_en.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for ja locale	finch-i18n_ja.js	Modify
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade general entry view	tradeGeneralEntry.jspx	New
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade Detailed entry view	tradeDetailsEntry.jspx	New
	/sample-trd/src/main/web/WEB-INF/views/trd	jspx for trade details view	tradeGeneralDetail.jspx	New
	/sample-trd/src/main/web/scripts/trd/	js file for handling query result consolidation by setting consolidation parameters	sample-trd-query-result.js	New
Controller	/sample-trd/src/main/java/com/nrift/sample/trd/web/controller/	js file for including a new formatter for the query result grid column	sample-trd-formatters.js	Modify
	/sample-trd/src/main/java/com/nrift/sample/trd/web/controller/	Controller class, contents all the action APIs for Trade Amend	TradeAmendController.java	New
Service	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Interface that provides method related to Trade	TradeService.java	New
	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Implementation for TradeService Interface	TradeServiceImpl.java	New
Repository	/sample-trd/src/main/java/com/nrift/sample/trd/domain/repository	contains all the necessary methods to perform CRUD operation	TradeRepository.java	New

4. Sequence of Steps

4.1 Create the model

4.1.1 Create Tables and Sequences

See [Wizard Based Entry- 4.1.1 Create Tables and Sequences](#) for the new entity section for more details.

4.1.1 CRUD Generation Tool

See the [Single Page Entry - 4.1.2 CRUD Generation Tool](#) section for CRUD Generation Tool details.

4.1.3 Create Command Form

See the [Wizard Based Entry - 4.1.3 Create Command Form](#) section for Create Command Form.

4.2 Create the view

The wizard based amend view differs from wizard based entry view in the following ways :

- There is a back button in the amend page. It allows the user to go back to the query result page.
- In wizard based amend by clicking the Ok button from system confirmation screen, user goes back to query result screen instead of new entry screen as in wizard based Amend.

4.2.1 Screen Consolidation

In order to reduce the number of menu items and consolidate the finch screens, amend action is not available as a separate menu item. In menu, only entry and query items are available and amend action is associated with each record in the query result screen.

Based on the application role of the logged in user, some, all or none of the actions will be available. Developer needs to set up roles and menu items for every new entry/amend page.

See the [Single Page Amend - Screen Consolidation](#) section for complete details.

The actionUrl associated with the amend action is used to navigate to the amend page.

For example,

For trade amend, the ACTION_URL mapped to "TRDTAM" is "trd/amend/actionable.json". The SCREEN_ID and VERSION_NUMBER mapped to "TRDTAM" having SCREEN_KEY="10004" is "TRDAM" and "1.2" respectively.

So the actionUrl formed is "trd/amend/actionable.json?screenId=TRDAM&versionNo=1&pk=3941&type=amend". On clicking the amend icon from the trade query result page, user navigates to the trade amend page.

See the [Query/Query Result - Initialize new query result page](#) section for details regarding action consolidation.

4.2.2 Basic components of wizard based amend screen

Query Page
Query result page
Query result page showing the sub-action menu
Amend Page Tab1
Amend Page -Tab2
Amend User Confirmation
Amend System Confirmation

Search Menu

TRDQR:Trade Query [20140503]

Trade Type	Trade Reference No	Account No	Inventoried
Security Code	Trade Currency	Settlement Currency	Trade Date
Value Date From - To	Instrument Type	Execution Market	
Principal/Agent Flag	Entry Date Form - To	Last Entry Date From - To	External ID
Cancel Reference No	Account Balance Type		Trade Status
Data Source			
Buy/Sell Flag	RESTORE ALL		
Sort Criteria ① Trade ... DESC	② DESC	③ DESC	
④ DESC	⑤ DESC	⑥ DESC	
⑦ DESC	⑧ DESC		

PERSONALIZE

For details regarding the query page blocks, components and features see [Query/Query Result - Basic components of query/query result page section](#).

Search Menu

TRDRS:Trade Query Result [20140503]

Trade D...	Reference No	Trade ...	Trade Typ...	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fi...	External R...	Trade Stat...	Instrumen...	
20131114	T20131009001555	USD	EQ	20131115	1	121212007	USD	S	TRDENTR01	NORMAL	ST	
20131114	1 09001559	USD	EQ	20131115	1	121212007	USD	S	TRD01	NORMAL	ST	
20131114	T20131009002223	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST	
3	013114	T20131009002235	USD	EQ	20131115	1	121212007	USD	S	1g2w3e	NORMAL	ST
20131114	T20131009002196	USD	EQ	20131115	1	121212007	USD	S	2222	NORMAL	ST	
20131114	T20131009002290	USD	EQ	20131115	1	121212007	USD	S	3333	NORMAL	ST	
20131114	T20131009002330	USD	EQ	20131115	1	121212007	USD	S	222	NORMAL	ST	
20131114	T20131009002318	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002213	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002348	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST	

- Pagination section of query result grid. See [Page and Record Count component](#) document for further details.
- Button section of query result grid with Export as pdf, Export as excel, Column Picker and Save icons. See [Grid Component](#) document for further details.
- Consolidation link for displaying jqDock menu for entry amend and cancel operations. See [Wizard Based Amend, Single Page Cancel](#) documents and Initialize new query result page section below for further details.

Search Menu 

TRDRS:Trade Query Result [20140503]

Page 1 of 199 Records: 1989

Trade D...	Trade ...	Trade Typ...	Value Date	Account B...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Reference No	Instrumen...
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	9988	NORMAL	T20131009002233	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	1q2w3e	NORMAL	T20131009002235	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	2222	NORMAL	T20131009002195	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	222	NORMAL	T20131009002330	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	333	NORMAL	T20131009002213	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	11	NORMAL	T20131009002348	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	2222e	NORMAL	T20131009002193	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	555	NORMAL	T20131009002185	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	22	NORMAL	T20131009002306	ST
20131114	USD	EQ	20131115	1	12/12/2007	USD	S	1111	NORMAL	T20131009002294	ST


Atech

For details regarding the query result page components and features see [Query/Query Result - Basic components of query/query result page](#) & [Initialize new query result page](#) section.

Search Menu

TRDEN:Trade Entry [05/03/2014]

The form consists of several input fields and dropdown menus. Callouts point to specific elements:

- Callout 1:** Points to the 'Trade General' button.
- Callout 2:** Points to the 'EQ' dropdown under 'Trade Type'.
- Callout 3:** Points to the date '01/01/2015' in the 'Trade Date' field.
- Callout 4:** Points to the account number '124456892/F' in the 'Account No' field.
- Callout 5:** Points to the date '01/01/2015' in the 'Value Date' field.
- Callout 6:** Points to the 'SAVE TEMPLATE' button at the bottom.

Fields and controls visible include:

- Trade Type: EQ
- Trade Date: 01/01/2015
- Value Date: 01/01/2015
- Buy/Sell C
- Account No: 124456892/F
- Security Code: CISCO
- Trade Curr
- Settlement Currency: JPY
- Quantity: 23
- Execution Market: HSBC
- Price
- Account Balance Type

Buttons: SAVE TEMPLATE

1. Breadcrumb section - See the Create the Controller and Usage of tiles section for further details.
2. Drop-down input control - See [Form Elements](#) - Drop-down input control section for further details.
3. Date Picker control - See [Form Elements](#) - Date Picker control section for further details.
4. Account Popup - See [Use a pop-up](#) - Account popup section for further details.
5. Currency popup - See [Use a pop-up](#) - Currency popup section for further details.
6. Action area of entry form - See the Initialize new entry page section below for details.

Search Menu 

TRDEN:Trade Entry [05/03/2014]

1 2 3 Trade Details

Trade Type	EQ	Trade Date	01/01/2015	Value Date
Buy/Sell Orientation	AB	Account No	124456892/F	Inventory Account No
Security Code	CISCO	Quantity	23	Price
Execution Market	B001	Trade Currency	JPY	Settlement Currency
Account Balance Type				

Tax Fee

+  ID	Rate	Rate Type	Amount
 COUPON	50	CPS	

RR

RR Role	RR Number
Executing RR	12
RR	
Split RR1	
Split RR2	
Trader	

Net Amount  External Reference No

SAVE TEMPLATE



1. Detail block - This section contains the inputs from the first tab of trade entry in the non-editable format. See Create new entry page section below and [Detail Page Layout](#) for further details.
2. Editable Grid section mentioned above see [Grid Component](#) for further details.

The screenshot shows a software interface for 'TRDEN:Trade Entry [05/03/2014]'. At the top, there is a navigation bar with a search menu icon and a magnifying glass icon. Below the navigation bar, the title 'TRDEN:Trade Entry [05/03/2014]' is displayed. A horizontal tab bar below the title contains three items: '1 Trade General' (highlighted in green), '2 Trade View', and '3 Trade Details'.
The main content area displays trade details in a table format:

Trade Type	EQ	Inventory Account No	1501020	Inventory A/C Short Name	Account B
Account No	124456892	Short Name		Security Code	CISCO
Price	12233	Trade Date	01/01/2015	Value Date	01/01/2015
Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY

Below the table, there is a large empty space. Two red arrows point from callout boxes to specific areas: one arrow points to a yellow box labeled '1' located in the middle of the empty space, and another arrow points to a yellow box labeled '2' located at the bottom of the empty space.

1. Trade general detail block - See [Detail Page Layout](#) for further details.
2. Action area of user confirmation page - See [Create user confirmation page](#) for new entry section below for further details.

Trade Type	EQ	Inventory Account No	1501020	Inventory A/C Short Name	
Account No	124456892	Short Name		Security Code	CISCO
Price	12233	Trade Date	01/01/2015	Value Date	01/01/2015
Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY

1. System confirmation message - See Create system confirmation page for new entry section below for details.
2. Trade general detail block - See [Detail Page Layout](#) for further details.

4.2.3 Create amend page

Developer does not need to create a new view for amend. The same view is used for wizard based entry and amend. The difference lies in the pre-populated form field values and the action area section.

- The request for fetching the amend form field values is handled by `consolidateLinkHandler` method defined in `finch-consolidation.js`.

See the [Query/Query Result](#) - Initialize new query result page - Define the action consolidation point for details on `consolidateLinkHandler` method.

- The buttons in the action area of the entry/amend page are determined based on the `actionType` and `mode` properties of the `options` object used to load the wizard plugin.

The amend page allows users to :

- Change pre-populated inputs and /or provide new inputs
- Navigate back to the query result page using Back button
- Submit the form using Submit button
- Reset the form using Reset button

For example,

For trade amend, on clicking the back button user navigates back to the trade query result screen.

The back handler, defined in `finch-wizard.js`, handles all back operations in the entry/amend flow as shown below. It applies the back logic depending on the `actionType` property of the `options` object used to load the wizard plugin.

See the [Wizard Based Entry](#) - Usage of Tiles, Create new entry page and Initialize new entry page sections for view details common to wizard based entry and amend flow.

4.2.4 Create user confirmation page for amend

As mentioned above, after successful submission of the amend form, user navigates to the user confirmation page.

Developer does not need to create a new view for amend user confirmation page. The view used for entry user confirmation page is same as the one used for amend user confirmation page.

See the [Wizard Based Entry - Create user confirmation for new entry section](#) for further details.

4.2.5 Create system confirmation page for amend

After successful submission of the user confirmation page, user navigates to the system confirmation page.

See the [Wizard Based Entry - Create system confirmation for new entry section](#) for further details.

The system confirmation page for amend differs from entry system confirmation page in allowing the user to go back to the query result screen using the Ok button.

For example,

In the trade amend flow, after successful form submission, user navigates to the system confirmation page where the confirmation message is shown.

On clicking the Ok button here, user goes back to the trade query result page.

The ok handler, defined in finch-wizard.js, handles all exit operations from the system confirmation page as shown below. It applies the exit logic depending on the actionType property of the options object used to load the wizard plugin.

finch-wizard.js

» [Expand](#)

```
this.ok = function() {
    // hooks
    if (self.invoke('ok') == false) {
        self.renderer.update();
        return;
    }
    self.invoke('unload');
    if (self.options.exitUri) {
        var url = self.options.exitUri;
        if (url.search('exitURI') != -1) {
            var queryCommandFormId = self.$context.find(
                '#queryCommandFormId').val();
            if (queryCommandFormId !== 'undefined'
                && queryCommandFormId.length > 0)
                if (url.indexOf('?') != -1)
                    url = url + '&commandFormId=' + queryCommandFormId;
                else
                    url = url + '?commandFormId=' + queryCommandFormId;
        }
        FINCH$Handler$asynchronous.generic(undefined, {
            requestUri : url
        });
        self.options.exitUri = undefined;
    } else {
        if (self.source === 'wizard-query-page') {
            url = FINCH.context.path + self.options.ok
            + "/ok/result?commandFormId="
            + self.$context.find('input[name=commandFormId]').val();
            FINCH$Handler$asynchronous.generic(undefined, {
                requestUri : url
            });
        } else {
            // no-op
        }
    }
}
```

[source](#)

4.3 Create the controller

4.3.1 Follow Inheritance rule

In finch, a Controller should extend base class com.nrft.fin.ch.inf.web.controller.FinchAbstractEditWizardController.java. The annotation '@Controller' is being used to consider a simple class as a Controller.

Create a logger object in the controller. See proper usage of logger in [Logging](#).

For entry & amend single controller can be reused.

The following Controller has been added for Trade Amend :

TradeAmendController.java

› [Expand](#)

```
@Controller  
@SessionAttributes("commandForm")  
@RequestMapping(value = "/trd/amend/**")  
public class TradeAmendController extends FinchAbstractEditWizardController<TradeAmendCommandForm>  
implements  
    MessageSourceAware {  
    private static final Logger log = LoggerFactory.getLogger(TradeAmendController.class);  
    ..  
}
```

[source](#)

Inside the controller, developer has to implement the following methods to complete the Amend operation

TradeAmendController.java

› [Expand](#)

```
protected void doInit(C commandForm) throws FinchException {  
    // no-op;  
}  
protected void doNavigate(C commandForm, int index) throws FinchException {  
    // no-op;  
}  
protected void initializeScreens(C commandForm) throws FinchException {  
    // no op  
}  
protected void setScreenName(C commandForm) throws FinchException {  
    // no op  
}  
protected void doValidate(C commandForm, int index) throws FinchException {  
    // no-op;  
}
```

[source](#)

4.3.2 Provide request URL mapping

The developer should specify the request mapping at the class level. A URL mapping should be specified for each and every controller in finch. The annotation '@RequestMapping' is being used to set the request URL. The URL should follow the Action URL Pattern specified (in database) for the Menu item clicked. By using the annotation @SessionAttribute the CommandForm has been specified as a session attribute. The following mapping has been used for TradeAmend:

TradeAmendController.java

```
@Controller
@SessionAttributes("commandForm")
@RequestMapping(value = "/trd/amend/**")
public class TradeAmendController extends FinchAbstractEditWizardController<TradeAmendCommandForm>
implements
    MessageSourceAware {
    private static final Logger log = LoggerFactory.getLogger(TradeAmendController.class);
    ..
}
```

4.3.3 Check the requested action is to be performed or not

When the user clicks on the consolidation icon (Amend / Enable) in the consolidation page at first this need to be checked that whether the particular record is eligible for the selected action or not.(Ex : user can't perform a amend action on already closed record).This validation is done based on the boolean status of isActionable flag . Developer have to check the the above mentioned condition explicitly.

4.3.4 Define screen specific initialization logic

When the user clicks on the consolidation icon (Amend / Enable) the next screen need to be initialize with detail information of the target record. The initialization have been done in performAction() method of the controller which is being called by clicking on the Amend consolidation icon .

Basically, the initialization should include:

- Fetch the detail information from the database for the target record (The record in the query result screen for which the action (amend / close / reopen) will be performed)
- Prepare the CommandForm/screen from the detail information fetched from the database.
- Populating the possible values for the drop downs in the screen. Most of the time the values are being populated using the Constraints.

4.3.5 Define server side validation logic

When user submits an amend form after specifying values to the fields, the form needs to be validated first. This validation is being done in doValidate() method .The base class FinchAbstractEditWizardController.java has a no-op implementation of this method, so that individual controllers can implement it in different ways as per functionality. The developer only has to implement the method in the controller, he doesn't need to call it explicitly. This method has been called implicitly by the submit() method in base class FinchAbstractEditWizardController.java.

4.3.6 Define form submit and user confirmation logic

The submission of an amend form is being handled by the method submit() of the base class FinchAbstractEditWizardController.java. This method includes:

- Validation of the values specified in the entry form.
- (If required) Population of few values according to the functionality from the specified values in the entry form.
- Finally, returning the tile name of the JSPX for User Confirmation screen.

4.3.7 Define system confirmation and data persistence logic

The confirmation of an amend page is being handled by the method confirm() of the base class FinchAbstractEditWizardController.java. T his method includes:

- Persist all the data into the database.
- Finally, returning the tile name of the JSPX for System Confirmation screen.

The implementation of persisting data, handling XML messages differ from controller to controller as per functionality. So the base class has a no-op doConfirm() method, by implementing which the developer completes all these activities in the controller. The returning of

the tile name of the JSPX for System Confirmation is being handled by the base class.

4.3.7.1 Concurrency handling

Proper concurrency handling is the important part of amend / cancel / operation.

See [Handle of Concurrent operation - 3.5 Steps to implement Locking for UI](#) for details.

4.3.8 Transaction Management

Transaction has been properly managed by the base class. The developer has nothing to do with this.

For details of Transaction Management see [Transaction Management](#).

4.3.12 Access Log

In Finch, whenever a Wizard Page Amend screen is being added, the Access Log implementation should be done properly. From this Access Log, we can get history of accessing of the screen.

For Access Log of a Single Page Amend, the developer has to only use the standard URL pattern for amend operation. The rest has been handled by the framework. Standard URL pattern for an operation is like "secure/<module_in_small>/<entity_in_small>/<operation_in_small>/*", where ref, trd, stl are the modules; account, customer, trade, settlement are the entities; entry, amend, query are the operations. For example, the URL for Strategy amend should be like 'secure/ref/strategy/amend/actionable.json'.

If any URL does not follow the standard pattern, then one has to configure that pattern in database tables for the Access Log implementation. For details of

this setting see [Access Log](#).

4.4 Create the Service

See [Single Page Entry - 4.5 Create the Service](#) for details.

4.5 Create the Repository

See [Single Page Entry - 4.5 Create the Repository](#) for details.

5. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

6. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	25-02-2015	Initial Version	Rahul c

3.3.4.6 Query/Query Result

1. Scope

finch application provides a query framework. Using this framework user can search an entity like, Trade, Employee, Application Role etc., based on some or all its associated fields. The query result, thus obtained, is displayed in the grid. From the query result page, user can initiate sub-actions like amend and cancel. These sub-actions can be initiated for every entity based on the application roles associated with the user. Developer must implement access control for accessing resources related to this screen.

This document gives a detailed account of the implementation details for query and query result pages. It outlines all the necessary steps, both on the client side and server side, to create a query page for an entity and display the query result.

2. Prerequisites

Must See

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Exception Handling](#) for proper handling of exceptions.

Refer to [UI Online Report](#) documents for PDF/XLS export.

Refer to [DB Utilities](#) document for implementation details of runs the query with the help of query runner, iterates the resultSet and converts resultSet rows into various other objects.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

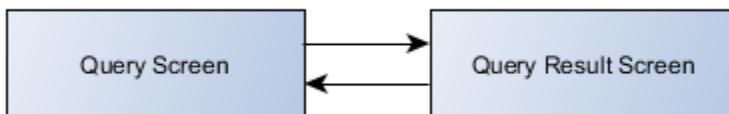
2.1 Templating

In finch , a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The screen flow for query page is shown below :



Screen	Purpose
Query Result	It is used to display the result obtained based on the query criteria. User can select a record from this screen and initiate sub-actions like amend & cancel on it.
Query	It is used to query for a particular entity based on some or all of its associated fields.

2.3 UI components

The amend page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

The classes and ids for the various UI components, used in the existing amend views, should be retained to get the basic layout, look & feel and functionality of form forms.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

2.5 i18n

finch provides i18n support for the following :

- The labels of form elements in the UI
- The menu, screen and section names
- The client side validation and information messages
- The server side validation and error messages
- The constraint values for the user
- The reports

See the [Multilingual i18n Support](#) document for more details on both client side and server side i18n Implementation.

2.6 Exception propagation and error handling

In finch application the error messages or the alert messages shown to the user can be either from the client side or from the server side.

The form input data validations are handled on the client side and the business validations are handled on the server side.

See the [Exception Management](#) document for more details on error handling.

2.7 Menu creation & Action URL patterns

A predefined and uniform URL pattern is essential in a web application.

In finch, all the controller actions are mapped to a unique URL. This URL follows a strict pattern to ensure cleaner code and secure access.

See the [Add new Menu/Action](#) document for more details on URL pattern conventions and menu creation.

Please note that we have used the Trade Query use case as a reference throughout this document, for all the explanations.

3.New and Modified File List

Layer	Package/Location (sample)	File type/purpose	File name (sample)	New/Modified/Created
Model	/sample-trd/src/main/java/com/nrift/sample/trd/web/form/	Command Form (i.e. POJO) class	TrdQueryCommandForm.java	New
View	/sample-trd/src/main/web/WEB-INF/views/trd/	Module specific xml file to provide view tile definition	views.xml	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for en locale	sample-trd-i18n_en.js	Modify
	/sample-trd/src/main/web/scripts/trd/	Module specific i18n file for ja locale	sample-trd-i18n_ja.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for en locale	finch-i18n_en.js	Modify
	/finch-web-infra/src/main/web/scripts/inf/	Module specific i18n file for ja locale	finch-i18n_ja.js	Modify
	/sample-trd/src/main/web/WEB-INF/views/trd	jsp for query view	tradeQueryCriteria.jspx	New
	/sample-trd/src/main/web/WEB-INF/views/trd	jsp for query result view	tradeQueryResult.jspx	New

	/sample-trd/src/main/web/scripts/trd/	js file for handling query result consolidation by setting consolidation parameters	sample-trd-query-result.js	New
	/sample-trd/src/main/web/scripts/trd/	js file for including a new formatter for the query result grid column	sample-trd-formatters.js	Modify
Controller	/sample-trd/src/main/java/com/nrift/sample/trd/web/controller/	Controller class, contains all the action APIs for Query	TrdQueryController.java	New
Service	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Interface that provides method related to Trade Query	TradeService.java	New
	/sample-trd/src/main/java/com/nrift/sample/trd/domain/service	Implementation for TradeService Interface	TradeServiceImpl.java	New
Repository	/sample-trd/src/main/java/com/nrift/sample/trd/domain/repository	contains all the necessary methods to perform CRUD operation	TradeRepository.java	New

4. Sequence of Steps

4.1 Create the Model

For query operation, no database changes are required. Though it has to be decided what are the columns we are going to show in QueryResult screen.

Create Command Form

For sending and receiving the Form/Request values between client and server side, finch uses POJO classes i.e. CommandForms. A CommandForm contains all the field names (JSPX field names used in the screen) along with their getter() and setter() methods. One should be very careful about the data types of the member variables of a CommandForm. If there exists any mismatch between the data types of the variables of CommandForm and the variables used in the screen, it'll lead to an exception.

In finch , a CommandForm for query operation should always extend the base class com.nrift.finck.inf.web.QueryCommandForm.java. For Trade query, the TrdQueryCommandForm.java has been defined in the package "com.nrift.sample.trd.web.form".

A CommandForm should always have a method which'll reset the member variables like the following:

TrdQueryCommandForm.java

› Expand

```
..  
public class TrdQueryCommandForm extends QueryCommandForm {  
    private String tradeType = null;  
    private String accountBalanceType = null;  
  
    ..  
    /**  
     * @return the tradeType  
     */  
    public String getTradeType() {  
        return tradeType;  
    }  
    /**  
     * @param tradeType the tradeType to set  
     */  
    public void setTradeType(String tradeType) {  
        this.tradeType = tradeType;  
    }  
    ..  
    /**  
     * @return to get Account Balance Type  
     */  
    public String getAccountBalanceType() {  
        return accountBalanceType;  
    }  
    /**  
     * @param to set Account Balance Type  
     */  
    public void setAccountBalanceType(String accountBalanceType) {  
        this.accountBalanceType = accountBalanceType;  
    }  
}
```

source

4.2 Create the view

4.2.1 Usage of Tiles

In order to create the layout for any query page, developer needs to provide a tile definition of its corresponding view in views.xml.

For query page, developer needs two views :

- Query page
- Query result page

For example, in Trade Query developer needs to create two views :

- Trade query page
- Trade query result page

The view corresponding to trade query say "tradeQueryCriteria.jspx" needs to be defined in views.xml as shown below :

```
devel/sample-trd/src/main/web/WEB-INF/views/trd/views.xml
```

```
<definition name="tradeQueryCriteria" extends="queryCriteria">
<put-attribute name="content">
  <definition extends="criteria_and_order">
    <put-attribute name="criteria" value="/WEB-INF/views/trd/tradeQueryCriteria.jspx"/>
  </definition>
</put-attribute>
</definition>
```

The new view "tradeQueryCriteria.jspx" needs to extend "queryCriteria". By doing so, the new view inherits all the common functionality and markup from "criteriaOrder.jspx".

Trade query page inherits the following features/elements from its queryCriteria template :

- Form tag and all its attributes
- Header content
- Form action area containing all the buttons

```
devel/sample-trd/src/main/web/WEB-INF/views/trd/views.xml
```

```
<definition name="tradeQueryResult" extends="result">
<put-attribute name="content">
  <definition extends="queryResult">
    <put-attribute name="resultGridHeader" value="/WEB-INF/views/trd/testHeader.jspx"/>
    <put-attribute name="resultGrid" value="/WEB-INF/views/trd/tradeQueryResult.jspx"/>
    <put-attribute name="resultGridFooter" value="/WEB-INF/views/trd/testFooter.jspx"/>
  </definition>
</put-attribute>
</definition>
```

See the [View templates - Screen Composition](#) section to get further details.

4.2.2 Basic components of query/query result page

Using trade query as a reference, the basic sections and UI components of query/query result pages are shown below :

Query page blocks Query page components Personalization Save Query Query result components

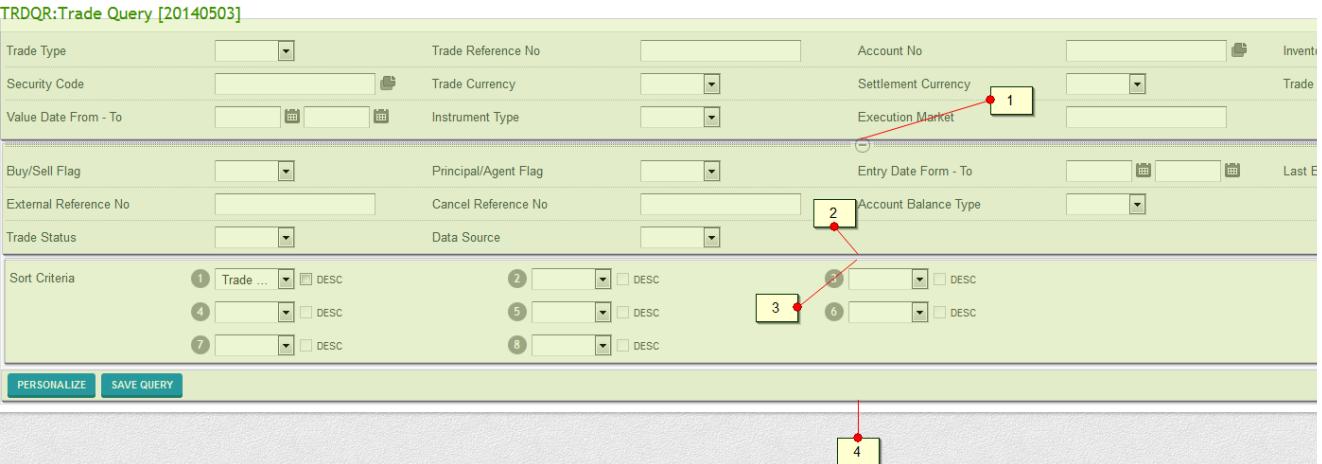
Search Menu 

TRDQR:Trade Query [20140503]

Trade Type	Trade Reference No	Account No	Inventor...
Security Code	Trade Currency	Settlement Currency	Trade...
Value Date From - To	Instrument Type	Execution Market	1
Buy/Sell Flag	Principal/Agent Flag	Entry Date Form - To	Last E...
External Reference No	Cancel Reference No	Account Balance Type	2
Trade Status	Data Source		
Sort Criteria	① Trade ... DESC ④ _____ DESC ⑦ _____ DESC	② _____ DESC ⑤ _____ DESC ⑧ _____ DESC	③ _____ DESC ⑥ _____ DESC

PERSONALIZE **SAVE QUERY**



1. Top block of query form. This block is mandatory for any query page.
2. More block of query form. This block is optional and can be added, if required for any query page. See [Query Form](#) document for further details.
3. Sorting block of query form. See [Form Elements - Sort field control](#) section for further details.
4. The form action area containing the buttons. See [Initialize new query page](#) section below for further details.

Search Menu

TRDQR:Trade Query [20140503]

Trade Type	Trade Reference No	Account No	
Security Code	Trade Currency	Settlement Currency	
Value Date From - To	Instrument Type	Execution Market	
Buy/Sell Flag	Principal/Agent Flag	Entry Date Form - To	
External Reference No	Cancel Reference No	Account Balance Type	
Trade Status	Data Source		
Sort Criteria	① Trade ... DESC ④ _____ DESC ⑦ _____ DESC	② _____ DESC ⑤ _____ DESC ⑧ _____ DESC	③ _____ DESC ⑥ _____ DESC

PERSONALIZE **SAVE QUERY**

1. Popup - See [Use a pop-up section for further details.](#)
2. Drop-down input control - See [Form Elements - Drop-down input control section for further details.](#)
3. Date Picker control - See [Form Elements - Date Picker control section for further details.](#)

Search Menu

TRDQR:Trade Query [20140503]

Trade Type Trade Reference No Account No Inventory

Security Code Trade Currency Settlement Currency Trade

Value Date From - To Instrument Type Execution Market

Principal-Agent Flag Entry Date Form - To Last Entry Date From - To External

Cancel Reference No Account Balance Type Trade

Data Source

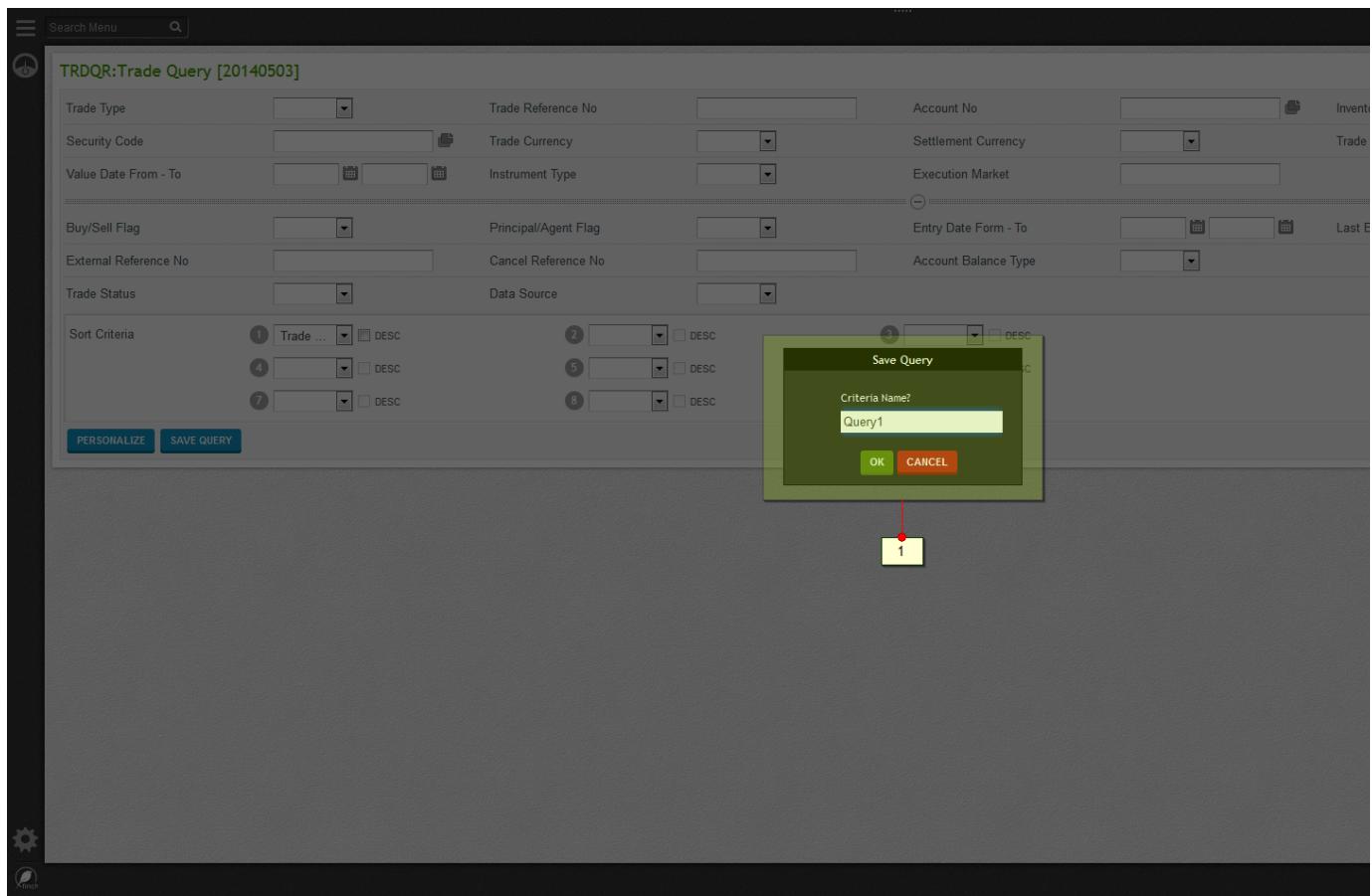
Buy/Sell Flag **RESTORE ALL**

Sort Criteria
 1 Trade ... DESC 2 DESC 3 DESC
 4 DESC 5 DESC 6 DESC
 7 DESC 8 DESC

PERSONALIZE

1. In personalization mode each form item is followed by a cross item to remove the element. All form items are also draggable in this mode.
2. If any form item is deleted then this restore section is displayed.
3. In this mode Personalize button is disabled and Save and Cancel buttons are enabled.

For all the features mentioned above see [Personalization](#) document



1. When Save Query button is clicked, the user is prompted to provide a name for the saved criteria.
See [Rich UI Framework](#) and [Save Query](#) section for further details.

Search Menu 

TRDRS:Trade Query Result [20140503]

Trade D...	Reference No	Trade ...	Trade Typ...	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...
20131114	T20131009001555	USD	EQ	20131115	1	121212007	USD	S	TRDENTR01	NORMAL	ST
20131114	1 09001559	USD	EQ	20131115	1	121212007	USD	S	TRD01	NORMAL	ST
20131114	T20131009002223	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST
3	013114 T20131009002235	USD	EQ	20131115	1	121212007	USD	S	1q2w3e	NORMAL	ST
20131114	T20131009002195	USD	EQ	20131115	1	121212007	USD	S	2222	NORMAL	ST
20131114	T20131009002290	USD	EQ	20131115	1	121212007	USD	S	3333	NORMAL	ST
20131114	T20131009002330	USD	EQ	20131115	1	121212007	USD	S	222	NORMAL	ST
20131114	T20131009002318	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST
20131114	T20131009002213	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST
20131114	T20131009002348	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST

4

1. Pagination section of query result grid. See [Page and Record Count component](#) document for further details.
2. Button section of query result grid with Export as pdf, Export as excel, Column Picker and Save icons. See [Grid Component](#) document for further details.
3. Consolidation link for displaying jqDock menu for entry amend and cancel operations. See [Wizard Based Amend, Single Page Cancel documents](#) and [Initialize new query result page](#) section below for further details.

4.2.3 Create new query page

After providing the tile definition, the developer needs to create the new query page "tradeQueryCriteria.jspx".

The "tradeQueryCriteria.jspx" needs to be placed in a specific folder. Its location will depend on the type of data it queries for.

Whatever be the type of data, all jsp files are kept within views folder. Within each views folder, a folder with the same name as the <module-name> is created.

For example :

"tradeQueryCriteria.jspx" will query for trade related information. Trade related information should always be placed in the "sample-trd" module.

Under the views folder within "sample-trd" create a "trd" folder. So the path for "tradeQueryCriteria.jspx" will be :

```
/devel/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
```

In the "tradeQueryCriteria.jspx" developer needs to create the form elements using specific classes and ids.

See the [Query Form](#) section to get the details about the markup needed to create a query page.

Now that the "tradeQueryCriteria.jspx" markup is ready, developer needs to map the form attributes and element values to a data model. In finch application commandForms have been used for sending the form values from client to the server.

The variable names used in the commandForms should be identical to the names used in the form elements and attributes.

For example :

For trade type list, the variable defined in TrdQueryCommandForm class is "tradeTypeList".

In "tradeQueryCriteria.jspx", the trade type select element options should be mapped to this variable using "\${commandForm.tradeTypeList}" as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
```

```
<div class="formItem">
    <form:label path="tradeType">
        <spring:message code="trade.label.trade.type" htmlEscape="false" />
    </form:label>
    <span>
        <form:select path="tradeType" class="dropdowninput">
            <form:option value="" />
            <form:options items="${commandForm.tradeTypeList}" />
        </form:select>
    </span>
    <span class="remove" title="Remove this field" style="display: none;">
        <spring:message text="" htmlEscape="false" />
    </span>
    <div class="clear">
        <spring:message text="" htmlEscape="false" />
    </div>
</div>
```

4.2.4 Initialize new query page

The query form allows users to :

- Provide query inputs
- Submit the form using Submit button
- Reset the form using Reset button
- Save the query using Save Query button

- Personalize the query form using Personalize button
- Remove query form items using the cross icon
- Save personalization using Save button
- Cancel personalization using Cancel button
- Restore form items deleted during personalization using Restore All

The hooks for all the above mentioned actions are defined in finch-form plugin. The list of hooks available are :

- submitFormHandler - It submits the query form and on successful submission renders the query result grid.
- resetQueryPage - It resets the query form to the original state.
- querySave - It performs the following tasks :
 - Before saving the query form, it performs mandatory field check (mandatoryFieldCheck) and duplicate sort field check (duplicateSortFieldCheck).
 - If these checks fail then an appropriate error message is shown using finch.postNotice method.
 - On passing these checks it prompts the user for providing a saved criteria name.
 - If the provided name length is less than or equal to 30 then the query form is saved successfully and the user is shown the confirmation message "Query Saved Successfully".
- personalize - It performs the following tasks :
 - Makes the form elements non-editable and draggable.
 - Displays the cross icon beside each form element.
 - Allows user to rearrange the form items and delete the form items.
 - The deleted items can be dragged into the form again.
 - If multiple elements items are deleted then all the items can be restored together by clicking on the Restore All button.
 - Allows user to save the personalized form
- removeFormItem - It removes the form item when the cross icon is clicked. The removed item is displayed below the respective blocks within the div having a class "deltopitems" or "delmoreitems".
- savePref - It saves the personalization changes done in the query form. On successful saving, the user is shown the confirmation message "Preference has been saved successfully".
- cancelPref - It cancels the personalization changes done and returns the user back to the editable state of the query form.
- restoreFormItem - It restores a selected deleted form element.
- restoreAllItems - It restores all deleted form elements.

Reference

See [Personalization](#) document for further details.

In the individual query page, developer needs to do the following :

1. Load the locale specific finch-i18n ,sample-trd-i18n and sample-ref-i18n files for supporting i18n. For example, in trade query, the page specific key value pairs need to be included in the specific i18n files - finch-i18n_en.js , finch-i18n_ja.js , sample-trd-i18n_en.js sample-trd-i18n_ja.js ,sample-ref-i18n_en.js and sample-ref-i18n_ja.js. These locale specific js files are loaded in "tradeQueryCriteria.jspx" as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

FINCH$onReady$Array.push(function() {
    FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/inf/finch-i18n', async: false}]);
    FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/trd/sample-trd-i18n', async: false}]);
    FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/ref/sample-ref-i18n', async: false}]);
});
```

Reference

See [Multilingual i18n Support & UI Core](#) for more details on i18n, finch.loadLocalizedScript and finch\$onReady\$Array.

2. If the query form has date fields and query popup plugin , load the finch-datevalidation.js and finch-popquery-form.js for using the date validation functions and query popup plugin.

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
```

```
FINCH.loadScript([
    {path: FINCH.context.path + '/scripts/inf/finch-datevalidation.js', async: false},
    {path: FINCH.context.path + '/scripts/inf/finch-popquery-form.js', async: false}
])
```

3. Initialize the finchform plugin.

In order to initialize finch form plugin, developer needs an unique id for each instance of the query page. The code snippet to generate this unique id is shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
```

```
<spring:eval expression="T(java.lang.System.currentTimeMillis())" var="myId" />
<input type="hidden" id="${myId}" />
```

Using the id generated above developer must initialize the plugin as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
```

```
jQuery('#' + '${myId}').parent().finchform({datepickerOptions:{changeMonth: true,changeYear: true}});
```

4.2.5 Initialize new query result page

The query result page allows users to :

- View the query records
- View the details of individual record in a dialog box
- Export the result as PDF
- Export the result as XLS
- Export the result as CSV
- Save the personalized grid view
- Choose the visible columns of the result grid
- Amend the entry
- Re-open the entry
- Cancel the entry

In the individual query result page like "tradeQueryResult.jspx" , developer needs to do the following:

1. Load sample-trd-formatters.js and then provide the grid result column definition as shown below :

```
/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx
```

```
var scripts = [
    {path: FINCH.context.path + '/scripts/trd/sample-trd-formatters.js', async: false}
];
```

/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx [Expand](#)

source

```
var grid_result_columns = [
    {name:<spring:message code="trade.label.trade.date"
        htmlEscape="false" />, field:"tradeDateStr",id:"tradeDateStr",
width:64,sortable:true,cssClass:'finch-grid-date'},
    {name:<spring:message code="trade.label.reference.no"
        htmlEscape="false" />, field:"tradeReferenceNo",id:"tradeReferenceNo",
width:147,sortable:true,formatter: Slick.Formatters.TradeDetailViewFormater,isForAuth:false },
    {name:<spring:message code="trade.label.trdae.ccy"
        htmlEscape="false" />, field:"tradeCcy",id:"tradeCcy", width:57,sortable:true},
    {name:<spring:message code="trade.label.trade.type"
        htmlEscape="false" />, field:"tradeType",id:"tradeType", width:78,sortable:true},
    {name:<spring:message code="trade.label.value.date"
        htmlEscape="false" />, field:"valueDateStr",id:"valueDateStr",
width:78,sortable:true,cssClass:'finch-grid-date'},
    {name:<spring:message
        code="trade.label.accountbal.type" htmlEscape="false" />,
field:"accountBalanceType",id:"accountBalanceType", width:78,sortable:true},
    {name:<spring:message
        code="trade.label.inventory.accountno" htmlEscape="false" />,
field:"inventoryAccountNo",id:"inventoryAccountNo", width:78,sortable:true},
    {name:<spring:message code="trade.label.settlement.ccy"
        htmlEscape="false" />, field:"settlementCcy",id:"settlementCcy", width:78,sortable:true},
    {name:<spring:message code="trade.label.buysell.flag"
        htmlEscape="false" />, field:"buySellFlag",id:"buySellflag", width:78,sortable:true},
    {name:<spring:message
        code="trade.label.external.referenceno" htmlEscape="false" />,
field:"externalReferenceNo",id:"externalReferenceNo", width:78,sortable:true},
    {name:<spring:message code="trade.label.trade.status"
        htmlEscape="false" />, field:"status",id:"status", width:78,sortable:true},
    {name:<spring:message
        code="trade.label.instrument.type" htmlEscape="false" />,
field:"instrumentType",id:"instrumentType", width:78,sortable:true}
];
```

2. Configure the formatter property.

In the grid columns there must be one column which uniquely identifies the record. In finch query result grid that column is formatted into a hyperlink which opens a dialog box displaying that record details. In order to format that column, the formatter property needs to be configured for that column as shown below :

/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx

```
var grid_result_columns = [
{
..
name:<spring:message code="trade.label.reference.no"
htmlEscape="false" />, field:"tradeReferenceNo",id:"tradeReferenceNo",
width:147,sortable:true,formatter: Slick.Formatters.TradeDetailViewFormater,isForAuth:false
},{name:<spring:message code="trade.label.trdae.ccy" htmlEscape="false" />
..
}
]
```

In trade query result, Trade Reference Number is the unique identifier which is formatted into a hyperlink. In the grid column definition, formatter property of reference number column is set to Slick.Formatters.TradeDetailViewFormater. The grid component formatters for trade query is defined in finch-formatters.js . The TradeDetailViewFormater formats the reference number as shown be

```
/sample-trd/src/main/web/scripts/trd/sample-trd-formatters.js

function TradeDetailViewFormater(row, cell, value, columnDef, dataContext) {
    if (value == null || value === "") {
        return "";
    }
    var urlPath = '/trd/query/';
    if($('form').attr('action').indexOf('/trd/amendcancel') > -1) {
        var urlPath = '/trd/amendcancel/';
    }
    var markup = ""
        + "<span class='detail-view-hyperlink' "
        + "view='tradeDetailView' "
        + "referenceno='" + value + "' "
        + "tradepk='" + dataContext.tradePk + "'"
        + "dialogIdentifier='tradeDetail'" + dataContext.tradePk + "'"
        + "duplicate = '" + FINCH.detailDialogTypeBehaviour+ "'"
        + "dialogTitle='[" + value + "] " + SAMPLE$TRD$i18n.formatter.trade_Details + "'"
        + "href='" + urlPath + "details/" + dataContext.tradePk + "/" + value + "?diffEnableFlag=" +
columnDef.isForAuth +"'>"
        + value
        + "</span>";
    return markup;
}
```

The opening of the details dialog is handled by FINCH\$detailViewHandler defined in finch-detail-dialog.js.

See [Rich UI Framework - Popup framework](#) section for further details.

3. Define the grid result settings.

The grid result settings includes the settings for enabling personalization save, pagination, column picker, export to excel and export to pdf as shown below :

/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx [Expand](#)

```
var grid_result_settings = {  
    enableToolbar:true,  
    consolidateActionFlag:true,  
    buttons:{  
        print:false,  
        xls:true,  
        pdf:true,  
        csv:true,  
        columnPicker:true,  
        save:true,  
        reset:true  
    },  
    pagingInfo:{  
        isNext : isNext,  
        url: urlPath + 'count.json',  
        pageNo: <c:out value="${pageNo}" />,  
        isPrevious: <c:out value="${isPrevious}" />  
    },  
    urls:{  
        nextPage : urlPath + 'result.json?fetch=next',  
        prevPage : urlPath + 'result.json?fetch=previous',  
        pdfReport: urlPath + 'report/trade.pdf?filetype=pdf',  
        xlsReport: urlPath + 'report/trade.xlsx?filetype=xlsx',  
        csvReport: urlPath + 'report/trade.csv?filetype=csv'  
    },  
    consolidationActionCallback: function(rec) {  
        return rec.status !== 'CANCEL';  
    }  
};
```

[source](#)

4. Define the grid result data.

/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx [Expand](#)

```
var row_id = 0;
var rec = {};
<c:forEach items="${value}" var="dl">
rec = {};
row_id+=1;
rec.id = "finch_" + row_id;
rec.tradePk= "<c:out value='${dl.tradePk}' />";
rec.tradeDateStr = "<c:out value='${dl.tradeDateStr}' />";
rec.tradeReferenceNo = "<c:out
value='${dl.tradeReferenceNo}' />";
rec.tradeCcy = "<c:out value='${dl.tradeCcy}' />";
rec.tradeType = "<c:out value='${dl.tradeType}' />";
rec.valueDateStr = "<c:out value='${dl.valueDateStr}' />";
rec.accountBalanceType = "<c:out
value='${dl.accountBalanceType}' />";
rec.inventoryAccountNo = "<c:out
value='${dl.inventoryAccountNo}' />";
rec.settlementCcy = "<c:out value='${dl.settlementCcy}' />";
rec.buySellFlag = "<c:out value='${dl.buySellFlag}' />";
rec.instrumentType = "<c:out
value='${dl.instrumentType}' />";
<c:set var="externalReferenceNo" value="${dl.externalReferenceNo}"/>
rec.externalReferenceNo = "${fn:replace(externalReferenceNo,'','¥¥')}";
rec.status = "<c:out value='${dl.status}' />";
grid_result_data.push(rec);
</c:forEach>
```

[source](#)

Using the parameters defined above the query result grid is initialized and rendered in queryResult.jspx as shown below:
\$('#queryResultForm .finch-grid').finchgrid(grid_result_data, grid_result_columns, grid_result_settings);

Reference

The generic request-response handler - FINCH\$Handler\$default is used to fetch the records of the query result page.
See [UI Core and Grid Component](#) for further details.

User can go back to the query screen from the query result page by clicking on the query tab.

5. Define the action consolidation section to enable entity amend, re-open or cancel.

For entity amend, cancel or re-open, at first user needs to query for that entity. In the query result section, the links for amend, cancel and re-open appear based on the application role associated with the user.

An user can have access to all or some of these operations. For this purpose, an action consolidation section needs to be included in every new query result page as shown below :

/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx [Expand](#)

```
<div id="consolidateAct" class="consolidateActionArea"
style="display: none;">
<c:forEach items="${navAction}" var="record">
<a class="consolidateActLink" href="${record.actionUrl}"> <c:choose>
<c:when
test="${fn:containsIgnoreCase(record.actionId, 'TRDTAM')}">
<spring:message code="trd.tradeamendaction.label.tradeamend" htmlEscape="true" var="amend"/>

</c:when>
<c:when
test="${fn:containsIgnoreCase(record.actionId, 'TRDTCX')}">
<spring:message code="trd.tradecancelaction.label.tradecancel" htmlEscape="true" var="cancel"/>

</c:when>
<c:when
test="${fn:containsIgnoreCase(record.actionId, 'TRDACA')}">
<spring:message code="trd.tradeamendaction.label.tradeamend" htmlEscape="true" var="amend"/>

</c:when>
<c:when
test="${fn:containsIgnoreCase(record.actionId, 'TRDAXC')}">
<spring:message code="trd.tradecancelaction.label.tradecancel" htmlEscape="true" var="cancel"/>

</c:when>
<c:otherwise>

</c:otherwise>
</c:choose>
</a>
</c:forEach>
</div>
```

[source](#)

Based on the "navAction" value obtained from the server side the icons for amend or cancel operations are displayed. See the Create the Controller section for more details.

When the consolidateActionFlag is set to true in grid settings, an additional column with the formatter property set to Slick.Formatters.ConsolidateActFormatter is added in finch-grid.js as shown below :

finch-grid.js

```
var columnDefObj = $.extend({name:"", field:"cnslAct", id:"cnslAct", width:30,
formatter:Slick.Formatters.ConsolidateActFormater,draggable:false}, settings.consolidateAttribute);
columns.unshift(columnDefObj);
```

ConsolidateActFormater formats the newly added column as shown below :

finch-formatters.js

```
function ConsolidateActFormater(row, cell, value, columnDef, dataContext) {
    //var mkp = '<input type="checkbox" id="cnslAct_'+row+'" class="consolidateActCkBox" row="'+row+''
    pk="'+ dataContext.pk + '" />';
    if(dataContext.isConsEnabled === false) return "";
    else {
        var mkp = '<span id="cnslAct_'+row+'" class="select-consolidation" row="'+row+'"
        pk="'+ dataContext.pk + '" title="'+ FINCH.i18n.grid.consolidate +'></span>';
        return mkp;
    }
}
```

On clicking on the select-consolidation link, the amend/cancel jqDock menu is displayed to the user based on the associated application role. The handler for enabling jqDock menu is defined in finch-consolidation.js as shown below :

finch-consolidation.js

```
$('.select-consolidation').grid-canvas').live('click', function(e) {
    dockReady = false;
    //Check if there is any role of this user
    var formContainer = $(this).closest('.formContent');
    var consolidateActionArea = $('.consolidateActionArea',formContainer);
    if($('.a.consolidateActLink',consolidateActionArea).length==0){
        FINCH.postNotice(FINCH.notice.type.info, 'You do not have any role associate with this result
page.');
        return;
    }
    consolidateActionArea.before('<div class="app-overlay"></div>');
    /*enable Jqdoc menu */
    var dockOptions = { align: 'middle', labels: 'bc', size: 70, distance: 120, duration: 50
    ,onReady:function(){dockReady = true;} };
    $('#consolidateAct').jqDock(dockOptions);
    // Page level implementation should give actual implementation of consolidateParam method
    consolidateParmas(e);
});
```

A page specific js file needs to be included for every new query result page to provide the actual implementation of the consolidateParam method. For example, in tradeQueryResult.jspx, sample-trd-query-result.js is included, where the consolidateParmas method is defined as shown below :

sample-trd-query-result.js

```
var consolidateParmas = function(e){  
    var row = $(e.target).attr('row');  
    var gridData = $('.finch-grid').data("gridInstance").getData().getItems();  
    var parmas = '&pk=' + gridData[row]['tradePk'];  
    $('a.consolidateActLink').data("params", parmas);  
};
```

The consolidateParam method sets the data property of consolidate act link used for amend/cancel operation.

The actual handler for consolidation, consolidateLinkHandler, is defined in finch-consolidation.js. It first checks if data.isActionable is true or not. Suppose we click close action for cancel record. Here it is not a doable action. If data.isActionable is true, then the actual action is done in the success method handler as shown below :

finch-consolidation.js

» [Expand](#)

[source](#)

```
var consolidateLinkHandler = function(e) {  
    var action = $(e.target).attr('id');  
    var status = $(e.target).attr('status');  
    if(status == 'ready'){  
        e.preventDefault();  
        var formContainer = $(e.target).closest('.formContent');  
        var formUrl = $(e.target).closest('#queryResultForm').attr('action');  
        formUrl = formUrl.replace("back", "result");  
        var consolidateActionArea = $('.consolidateActionArea', formContainer);  
        var param = $('a.consolidateActLink', consolidateActionArea).data("params");  
        var fragmentName = '&fragments=content';  
        var actionType = '&type=' + action;  
        var actionURL = $(e.target).parent().attr('href').indexOf('/') == 0 ? $(e.target).parent().attr('href') : ('/' +  
            $(e.target).parent().attr('href'));  
        var sortAsc = $('#finchGridSortAsc').val();  
        var sortCol = $('#finchGridSortCol').val();  
        var url = FINCH.context.path + actionURL + param + actionType;  
        var container = $("#content");  
        FINCH$Handler$default.generic(undefined, {  
            requestUri: url,  
            requestType: FINCH$Handler$default.requestType.asynchronous,  
            contentType: FINCH$Handler$default.contentType.json,  
            onJsonContent: function(e, options, $target, data) {  
                if(data.onlineAccessRestriction) {  
                    FINCH.postNotice(FINCH.notice.type.info, FINCH.i18n.message.office_close_message);  
                    return false;  
                }  
                if(data.isActionable){  
                    var commandFormDetail = {queryCommandFormId : $('[name=commandFormId]', container).val(),  
                        queryScreenUrl : formUrl, sortAsc : sortAsc, sortCol : sortCol };  
                    url = url.replace("actionable.json", "performAction");  
                    url = url + fragmentName;  
                    var dialog = $('a.consolidateActLink', consolidateActionArea).data("dialog");  
                    if (!dialog) {  
                        FINCH$Handler$consolidateLinkHandler.generic(undefined, {requestUri: url, settings: {data :  
                            commandFormDetail}});  
                    } else {  
                        FINCH$detailViewHandler(e, {
```

```
view: 'view',
method: 'POST',
href: url + '&popup=true',
dialogTitle: $('a.consolidateActLink',consolidateActionArea).data("dialogTitle"),
dialogHeight: 500,
onOpen: function(){}
});
}
}else{
if(data.success){
    FINCH.postNotice(FINCH.notice.type.error, FINCH.i18n.grid.action_error, true); //Selected action
cannot be performed on this record
}else{
    FINCH.postNotice(FINCH.notice.type.error, data.value[0], true);
}
$('#consolidateAct').jqDock('destroy');
}
}
});
}};

}else{
```

```
        FINCH.postNotice(FINCH.notice.type.info, 'This feature is not yet implemented.');
    }
};
```

Reference

See [Wizard Based Amend](#) document for amend action.

See [Single Page Canceldocument](#) for cancel action.

4.3 Create the Controller

4.3.1 Follow Inheritance rule

In finch , a Query Controller should be added in the package "com.nrft.<module_in_small_letters>.web.controllers" and should extend base class com.nrft.fin.ch.inf.web.controller.AbstractQueryController. The annotation '@Controller' is being used to consider a simple class as a Controller.

Create a logger object in the controller. See proper usage of logger in [Logging](#).

The following Controller has been added for Trade query:

TrdQueryController.java

[Expand](#)

[source](#)

```
@Controller
public class TrdQueryController extends
AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {
/**
 * The <code>Log</code> instance for this class.
 */
..
}
```

The following methods of the base class AbstractQueryController.java should be overridden in the controller:

AbstractQueryController.java

[Expand](#)

[source](#)

```
/**
 * @param form
 *      - Action form
 * @param request
 *      - Servlet request
 * @throws FinchException
 *      on failure and forwards to error mapping.
 */
protected void doInit(C form, FinchModelMap finchModelMap)
throws FinchException {
}
/**
 * Overridable method to execute before query execution.
 *
 * @param form
 *      - Action form
 * @param request
 *      - Servlet request
 */

```

```

protected void doPreQuery(C form) throws FinchException {
    // Override to execute before query execution
}
/**
 * Overridable method to execute after query execution.
 *
 * @param form
 *      - Command Form
 */
protected void doPostQuery(C queryForm) throws FinchException {
    // Override to execute after query execution
}
/**
 * <p>
 * Does any sort of client implementation before exiting from the wizard and
 * returning back to the query result
 * </p>
 */
protected void preExit(C cmdForm, ModelMap map) throws FinchException {
    // no op
}
/**
 * This method does the pre-processing before evaluation of query expression.
 *
 * @param cmdForm Object of type {@link QueryCommandForm} associate with current request
 * @param map Object of type {@link FinchModelMap} associate with current request
 * @throws FinchException
 */
protected void preQueryExpressionEvaluation(C cmdForm, FinchModelMap map) throws FinchException {
    // no op
}
/**
 * This method does the post-processing after evaluation of query expression.
 *
 * @param cmdForm Object of type {@link QueryCommandForm} associate with current request
 * @param map Object of type {@link FinchModelMap} associate with current request
 * @throws FinchException
 */
protected void postQueryExpressionEvaluation(C cmdForm, FinchModelMap map) throws FinchException {
    // no op
}
/** @param commandForm
 *      : the query command form
 * @param map
 *      : the {@link PaginationModelMap}
 * @throws DataInputException
 */
protected void preDashboardQuery(C commandForm, PaginationModelMap map)
    throws DataInputException {
    // no op
}
/**@param the
 *      commandForm C
 * @param the
 *      model map ModelMap
 * @throws FinchException
 */
protected void preProcessSaveQuery(C commandForm, FinchModelMap map)

```

```
throws FinchException {  
// no op  
}
```

4.3.2 Provide request URL mapping

The developer should specify the request mapping at the class level. A URL mapping should be specified for each and every controller in Finch. The annotation '@RequestMapping' is being used to set the request URL. The URL should follow the Action URL Pattern specified (in database) for the Menu item clicked. By using the annotation @SessionAttribute the CommandForm has been specified as a session attribute. The following mapping has been used for Trade query:

TrdQueryController.java

» [Expand](#)

[source](#)

```
@Controller  
@RequestMapping(value = "/trd/query")  
@SessionAttributes({ "commandForm" })  
public class TrdQueryController extends  
AbstractQueryController<TrdQueryCommandForm> implements  
MessageSourceAware {  
..  
}
```

4.3.3 Define screen specific initialization logic

The base class AbstractQueryController.java has a no-op implementation of the method doInit(), so that the sub-classes can override it as per requirement. Different controllers provide functionality specific initialization logic in this method. Basically, the initialization should include:

- Initializing the member variables of CommandForm
- Setting some default values to the CommandFrom/Screen
- Populating the possible values for the drop downs in the screen. Most of the time the values are being populated using the Constraints. See [Constraint List of Values](#).

The following is the screen initialization of Trade query:

TrdQueryController.java

» [Expand](#)

[source](#)

```
protected void doInit(TrdQueryCommandForm form, FinchModelMap modelMap)  
throws FinchException {  
List<LabelValueBean> sortFieldList = new LinkedList<LabelValueBean>();  
try {  
Identity principal = null;  
if (Operation.getInstance() != null  
&& Operation.getInstance().getContext() != null) {  
principal = ((OperationScope) Operation.getInstance()  
.getContext()).getCallerIdentity();  
}  
Locale locale = I18NUtils.getLocale();  
sortFieldList.add(new LabelValueBean(messageSource.getMessage(  
"trd.sort.displayvalue.tradedate", null, locale),  
"tradeDate"));  
sortFieldList.add(new LabelValueBean(messageSource.getMessage(  
"trd.sort.displayvalue.valuedate", null, locale),  
"valueDate"));  
sortFieldList.add(new LabelValueBean(messageSource.getMessage(  
"trd.sort.displayvalue.instrumenttype", null, locale),  
"instrumentType"));  
}
```

```

sortFieldList.add(new LabelValueBean(messageSource.getMessage(
    "trd.sort.displayvalue.trdrefno", null, locale),
    "referenceNo"));
sortFieldList.add(new LabelValueBean(messageSource.getMessage(
    "trd.sort.displayvalue.tradetype", null, locale),
    "tradeType"));
sortFieldList.add(new LabelValueBean(messageSource.getMessage(
    "trd.sort.displayvalue.actbaltype", null, locale),
    "accountBalanceType"));
sortFieldList.add(new LabelValueBean(messageSource.getMessage(
    "trd.sort.displayvalue.buySellFlag", null, locale),
    "buySellFlag"));
sortFieldList.add(new LabelValueBean(messageSource.getMessage(
    "trd.sort.displayvalue.tradestatus", null, locale),
    "status"));
initCommandForm.setTradeTypeList(principal.getUserConstraint().get(
    "TRADE_TYPE"));
initCommandForm.setAccountBalanceTypeList(principal
    .getUserConstraint().get("ACCOUNT_BALANCE_TYPE"));
initCommandForm.setCounterPartyTypeList(principal
    .getUserConstraint().get("COUNTER_PARTY_TYPE"));
initCommandForm.setCcyMap(principal.getUserConstraint().get(
    "CURRENCY"));
initCommandForm.setInstrumentTypeMap(principal.getUserConstraint()
    .get("INSTRUMENT_TYPE"));
initCommandForm.setBuySellFlagMap(principal.getUserConstraint()
    .get("BUY_SELL_FLAG"));
initCommandForm.setPrincipalAgentFlagMap(principal
    .getUserConstraint().get("PRINCIPAL_AGENT_FLAG"));
initCommandForm.setTradeStatusList(principal.getUserConstraint()
    .get("TRADE_STATUS"));
initCommandForm.setDataSourceList(principal.getUserConstraint()
    .get("DATA_SOURCE"));
initCommandForm.setSortFieldList(sortFieldList);
ScreenService screenService = Operation.getInstance().getReference(
    ScreenService.class);
initCommandForm.setScreenName(screenService.getScreenName(
    initCommandForm.getScreenId(), locale.toString()));
if (!initCommandForm.isSavedQuery()) {
    initCommandForm.setHeaderInfo(WebUtils.getHeaderForScreen(
        initCommandForm.getScreenId(), getComponent()));
}
log.info("commandForm, uiDateFormat and forwardPath has been added to ModelMap");
} catch (FinchException e) {
    log.error("Failed to init TrdQueryController:", e);
    modelMap.addError(e);
}

```

```
}
```

Once initialization method is created, override the `getQueryFormViewName()` and `getQueryResultViewName()` methods, where the tiles view name which is defined in the `views.xml` file for defining the JSPX files, is specified.

TrdQueryController.java

» [Expand](#)

[source](#)

```
@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

.....
@Override
protected String getQueryFormViewName() {
    return "tradeQueryCriteria";
}

@Override
protected String getQueryResultViewName() {
    return "tradeQueryResult";
}

.....
}
```

After the `getQueryFormViewName()` and `getQueryResultViewName()` is overridden and appropriate entries are made in the `views.xml` file, override certain additional methods.

TrdQueryController.java

› Expand

```
@Controller  
@RequestMapping(value = "/trd/query")  
@SessionAttributes({"commandForm"})  
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements  
MessageSourceAware {  
  
....  
  
@Override  
protected String getComponent() {  
    // Specify the component under which the query ui is being created  
    return "TRD";  
}  
  
@Override  
public void setMessageSource(MessageSource messageSource) {  
    this.messageSource=messageSource;  
}  
  
@Override  
protected Class<?> getResultViewClass() {  
    return null;  
}  
  
....  
}
```

The date parameters used for the query criteria has to be overridden by the application in order to set the query parameters.

TrdQueryController.java

› Expand

```
@Controller  
@RequestMapping(value = "/trd/query")  
@SessionAttributes({"commandForm"})  
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements  
MessageSourceAware {  
  
....  
  
@Override  
protected String[] getDateParams() throws FinchException {  
    return new String[]{  
        "tradeDateFrom",  
        "tradeDateTo"};  
}  
  
....  
}
```

Finally, in controller class the following methods need to be specified so as to implement the reports (PDF,XLSX) for the query result. To implement, override the getReportName() and getReportParameters() methods.

TrdQueryController.java

```
@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

.....
@Override
protected String getReportName(){
    // Specify the report name. This report name with the formaat type of the report is specified in the
jasper-views.xml
    return "trdquery";

}

@Override
protected Map<String, Object> getReportParameters(@ModelAttribute("commandForm")
TrdQueryCommandForm cmdForm) throws FinchException {

    Map<String, Object> params = Maps.newHashMap();
    Map<String, String> criteriaMap = Maps.newLinkedHashMap();
    // Specify the report parameters....
    .....
    addCriteria(criteriaMap,"Trade Type", cmdForm.getTradeType());
    .....
    params.put("criteriaMap", criteriaMap);

    return params;
}
.....
}
```

To get the support multiple values of a form field, anew method on AbstractQueryController has been introduced.To use the functionality the prepareQueryInput(..) method is to be overridden, as shown below:

TrdQueryController.java

» [Expand](#)

```
protected Map<String, Object> prepareQueryInput(TrdQueryCommandForm queryForm) throws Exception {
    BeanWrapper bw = new BeanWrapperImpl(queryForm);
    Map<String, Object> map = Maps.newHashMap();
    for(PropertyDescriptor pd : bw.getPropertyDescriptors()) {
        map.put(pd.getName(), bw.getPropertyValue(pd.getName()));
    }
    return map;
}
```

4.4 Create the Result View

The Result View class is a POJO class which stores the value of the result of the performed query. Sample result view class is shown below:

TrdQueryResultView.java

[Expand](#)

```
public class TrdQueryResultView implements Serializable{
    private static final long serialVersionUID = 1L;
    private long tradePk;
    private String tradeType;
    private String tradeReferenceNo;
    private String tradeCcy;
    private Date tradeDate;
    private Date valueDate;
    private String buySellFlag;
    private long accountBalanceType;
    private String inventoryAccountNo;
    private String settlementCcy;
    private String instrumentType;
    private String externalReferenceNo;
    private String status;

    public TrdQueryResultView(long tradePk, String tradeType, String tradeReferenceNo,
        String tradeCcy, Date tradeDate, Date valueDate, String buySellFlag, long accountBalanceType, String
        inventoryAccountNo, String settlementCcy, String instrumentType,
        String externalReferenceNo, String status) {
        super();
        this.tradeType = tradeType;
        this.tradeReferenceNo = tradeReferenceNo;
        this.tradeCcy = tradeCcy;
        this.tradeDate = tradeDate;
        this.valueDate = valueDate;
        this.buySellFlag = buySellFlag;
        this.accountBalanceType = accountBalanceType;
        this.inventoryAccountNo = inventoryAccountNo;
        this.settlementCcy = settlementCcy;
        this.instrumentType = instrumentType;
        this.externalReferenceNo = externalReferenceNo;
        this.status = status;
        this.setId(tradeReferenceNo);
        this.tradePk = tradePk;
    }

    public String getTradeType() {
        return tradeType;
    }

    public void setTradeType(String tradeType) {
        this.tradeType = tradeType;
    }

    public String getTradeReferenceNo() {
        return tradeReferenceNo;
    }

    public void setTradeReferenceNo(String tradeReferenceNo) {
        this.tradeReferenceNo = tradeReferenceNo;
    }

    public String getTradeCcy() {
        return tradeCcy;
    }

    public void setTradeCcy(String tradeCcy) {
        this.tradeCcy = tradeCcy;
    }

    public Date getTradeDate() {
```

[source](#)

```
    return tradeDate;
}
public void setTradeDate(Date tradeDate) {
    this.tradeDate = tradeDate;
}

public String getBuySellFlag() {
    return buySellFlag;
}
public void setBuySellFlag(String buySellFlag) {
    this.buySellFlag = buySellFlag;
}
...
@Beta
// for grid rendering
private String id;
public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public long getTradePk() {
    return tradePk;
}
public void setTradePk(long tradePk) {
    this.tradePk = tradePk;
```

```
    }  
}
```

id field in the result view class needs to be set for the pagination to work. The value of the id field must be unique in nature

For pagination implementation, please refer to [Pagination Support](#)

4.5 Create the Service

As mentioned in the Controller section, to get the query result, access the database through different layers. Access the service layer which in turn access the repository layer. This section describes what needs to be created in the service layer and repository layer to get the results for the query.

In the controller, overwrite the getQueryService() method, as shown below:

TrdQueryController.java

» [Expand](#)

```
@Override  
protected TradeService getQueryService() {  
    if (tradeService == null) {  
        tradeService = Operation.getInstance().getReference(TradeService.class);  
    }  
    return tradeService;  
}
```

[source](#)

Create a service interface by extending the QuerySupport (Deprecated) or QuerySupportEx of finch framework, as shown below:

TradeService.java

```
public interface TradeService extends QuerySupportEx<TrdQueryResultView> {  
    ..  
}
```

The getQueryResult() method is defined in the QuerySupportEx interface

Create the corresponding implementation class for the service, as shown below:

TradeServiceImpl.java

› Expand

```
@Service
public class TradeServiceImpl implements TradeService{
.....
@Autowired
private TradeRepository trdRepo;

@Override
@Transactional(value = "sample-tm", readOnly = true)
public List<TrdQueryResultView> getQueryResult(Map<String, Object> queryInput,int startIndex,int
maxResult) throws FinchException {
try {
    TrdQueryCriteriaBuilder criteriaBuilder = new TrdQueryCriteriaBuilder(queryInput);
    JPAQueryExecutionParams<TrdQueryResultView> queryExParams = new
JPAQueryExecutionParams.Builder<TrdQueryResultView>()
        .withQueryCriteria(criteriaBuilder.build(trdRepo.getEntityManager()))
        .withstartIndex(startIndex)
        .withMaxResult(maxResult).build();
    return trdRepo.getQueryResult(queryExParams);
} catch (Exception ex) {
    log.error("Failed to execute Trade Query", ex);
    throw new TrdException(ex);
}
}
.....
}
```

[source](#)

While creating the implementation class, ensure about the followings:

- Use the @Service annotation to define it as Service class.
- Autowire the repository bean
- Specify the transactional manager using the @Transactional annotation. Do note the value for the transaction manager is either global-tm or sample-tm. In case of INF related objects, specify it as global-tm and in case of sample app objects, specify it as sample-tm.
- Use the appropriate criteria builder class object to be used with JPAQueryExecutionParams

4.5 Create the Repository

While creating the repository class, ensure about the followings:

- The class must be annotated as @Repository
- While defining the EntityManager we must use the @PersistenceContext annotation and specify the context. It can be either global-em or sample-em. global-em is used for INF entities and sample-em is used for sample app entities.

Create the repository class, as shown below:

TradeRepository.java

› Expand

```
@Repository
public class TradeRepository extends RepositoryBase<Trade, Long> {

    private static final Logger Log = LoggerFactory.getLogger(TradeRepository.class);

    @PersistenceContext(unitName="sample-em")
    private EntityManager em;
    @Override
    public EntityManager getEntityManager() {
        return em;
    }
    ...
}
```

[source](#)

4.6 Create Query Criteria Builder

The criteria builder class is one of the main class where we write the actual query for retrieving the data. Write the code for getting the results in the build() method using JPA API, as shown in the following code snippet.

The build method under QueryCriteriaBuilder has been deprecated. Use instead AbstractQueryCrietriaBuilder and its related

TrdQueryCriteriaBuilder.java

› Expand

```
public class TrdQueryCriteriaBuilder extends AbstractQueryCriteriaBuilder<TrdQueryResultView, Trade> {
    private static final Logger LOGGER = LoggerFactory.getLogger(TrdQueryCriteriaBuilder.class);
    source

    public TrdQueryCriteriaBuilder(Map<String, Object> queryInput) {
        super(queryInput);
    }

    @Override
    protected void processSelectClause(CriteriaBuilder builder, CriteriaQuery<TrdQueryResultView> cq,
    Root<Trade> trdRoot) {
        cq.multiselect(trdRoot.get("tradePk"),
            trdRoot.get("tradeType"),
            trdRoot.get("referenceNo"),
            trdRoot.get("tradeCurrency").get("securityId"),
            trdRoot.get("tradeDate"),
            ...
            trdRoot.get("status"));
    }

    @Override
    protected List<Predicate> getPredicateList(CriteriaBuilder builder, CriteriaQuery<?> cq, Root<Trade>
    trdRoot) throws FinchException {
        AccountService accountService = Operation.getInstance().getReference(AccountService.class);
        InstrumentService instrumentService = Operation.getInstance().getReference(InstrumentService.class);
        FinInstRoleService firService = Operation.getInstance().getReference(FinInstRoleService.class);
        List<Predicate> predList = Lists.newArrayList();

        // Criteria
        processQueryStringField(predList, builder, trdRoot, "tradeType", "tradeType");
        processQueryStringField(predList, builder, trdRoot, "referenceNo", "referenceNo");
    }
}
```

```

        processQueryMultipleStringValueField(predList, builder, trdRoot, "tradeCcy", new String[]{"tradeCurrency",
"securityId"});
        processQueryStringField(predList, builder, trdRoot, "buySellFlag", "buySellFlag");
        processQueryStringField(predList, builder, trdRoot, "accountBalanceType", "accountBalanceType");
        // inventory account no
        if (!Strings.isNullOrEmpty(getInputParam("inventoryAccountNo", String.class))) {
            Pair<String, String> codePair = parseCode(getInputParam("inventoryAccountNo",
String.class).trim().toUpperCase());
            Account account;
            try {
                account = accountService.findAccount(codePair.left(),
                codePair.right());
            } catch (RefException e) {
                LOGGER.error("", e);
                throw new TrdException("error.field.invalid", "Inventory Account");
            }
            predList.add(builder.equal(trdRoot.get("inventoryAccount").get("accountPk"), account.getAccountPk()));
        }

        // account no
        if (!Strings.isNullOrEmpty(getInputParam("accountNo", String.class))) {
            Pair<String, String> codePair = parseCode(getInputParam("accountNo", String.class).trim().toUpperCase());
            Account account;
            try {
                account = accountService.findAccount(codePair.left(), codePair.right());
            } catch (RefException e) {
                LOGGER.error("", e);
                throw new TrdException("error.field.invalid", "Account");
            }
            predList.add(builder.equal(trdRoot.get("account").get("accountPk"), account.getAccountPk()));
        }
    }

    ...
    // Trade Date From
    try {
        processQueryDateField(predList, builder, trdRoot, "tradeDateFrom", "tradeDate", DateComparator.GE);
    } catch (Exception e) {
        LOGGER.error("", e);
        throw new TrdException("error.field.invalid", "Trade Date From");
    }
    // Trade Date To
    try {
        processQueryDateField(predList, builder, trdRoot, "tradeDateTo", "tradeDate", DateComparator.LE);
    } catch (Exception e) {
        LOGGER.error("", e);
        throw new TrdException("error.field.invalid", "Trade Date To");
    }
}

...
processQueryStringField(predList, builder, trdRoot, "status", "status");
Subquery<Short> subQry = cq.subquery(Short.class);
Root<Trade> subRoot = subQry.from(Trade.class);
Path<Short> verNoExp = subRoot.get("versionNo");
subQry.select(builder.max(verNoExp));
Path<Short> varExp1 = trdRoot.get("versionNo");
subQry.where(builder.equal(subRoot.get("referenceNo"), trdRoot.get("referenceNo")));
predList.add(builder.equal(varExp1, subQry));
return predList;
}

```

```
@Override
protected Map<String, Path<?>> preparePathMap(Root<Trade> trdRoot) {
    Map<String, Path<?>> pathMap = new HashMap<>();
    pathMap.put("tradePk", trdRoot.get("tradePk"));
    pathMap.put("tradeType", trdRoot.get("tradeType"));
    pathMap.put("referenceNo", trdRoot.get("referenceNo"));
    pathMap.put("tradeCurrency", trdRoot.get("tradeCurrency").get("securityId"));
    pathMap.put("tradeDate", trdRoot.get("tradeDate"));
    pathMap.put("valueDate", trdRoot.get("valueDate"));
    pathMap.put("buySellFlag", trdRoot.get("buySellFlag"));
    pathMap.put("accountBalanceType", trdRoot.get("accountBalanceType"));
    pathMap.put("inventoryAccount", trdRoot.get("inventoryAccount").get("accountNo"));
    pathMap.put("settlementCurrency", trdRoot.get("settlementCurrency").get("securityId"));
    pathMap.put("instrumentType", trdRoot.get("instrument").get("instrumentType"));
    return pathMap;
}

private Pair<String, String> parseCode(String code) {
    String[] parts = code.split("/");
    if(parts.length ==1) {
        return Pair.pair(code, DEFAULT_CODE_TYPE);
    }
    return Pair.pair(parts[0], parts[1]);
}

@Override
protected Class<TrdQueryResultView> getResultViewClass() {
    return TrdQueryResultView.class;
}

@Override
protected Class<Trade> getRootEntityClass() {
    return Trade.class;
}
```

```
}
```

4.7 Create Scree Database Setup

Screen data setup for the criteria screen and result screen needs to be setup in the INF_UI_SCREEN table. The following database fields need to be populated:

- SCREEN_KEY
- SCREEN_ID
- SCREEN_NAME
- SHORT_NAME
- DESCRIPTION
- VERSION_NUMBER

Once the data setup for the criteria screen and result screen is done, then update the record for the criteria screen and set the value for the column NEXT_SCREEN_PK to with the SCREEN_KEY value of the result screen.

4.8 Pre Fetching Data

In finch, while performing query, data (query result) can be pre-fetched i.e. 'n' of query result pages can be pre fetched so that during pagination, database call will not be made for each pagination request. Database request will be made only when 'n' no pages of data has been traversed. This pre-fetching of data will only work with the new pagination model i.e. where getQueryService() method is implemented in the controller. For details see the Service section of this document. This pre fetching of data would work in both normal query pages and wizard based query pages. Do note, that once data is pre fetched, the data is not re-loaded.

Out of the box, finch pre-fetches data for 5 pages and on every 3rd page next set of data is loaded. However this configuration can be configured by the application. The configuration needs to be done in the keys.properties and app-pref.xml file.

keys.properties

```
pref.key.application.prefetchpagesize=PRE_FETCH_PAGE_SIZE  
pref.key.application.dataloadpageinterval=DATA_LOAD_PAGE_INTERVAL  
pref.key.application.recordsperpage=RECORDS_PER_PAGE
```

app-pref.xml

```
<ROOT>  
...  
<PRE_FETCH_PAGE_SIZE>5</PRE_FETCH_PAGE_SIZE>  
<DATA_LOAD_PAGE_INTERVAL>3</DATA_LOAD_PAGE_INTERVAL>  
<RECORDS_PER_PAGE_TYPE_AHEAD>20</RECORDS_PER_PAGE_TYPE_AHEAD>  
...  
</ROOT>
```

The tag <PRE_FETCH_PAGE_SIZE> would contain the value for the no of pages to be pre-fetched and <DATA_LOAD_PAGE_INTERVAL> tag would contain the value for loading the next set of data after every nth page.

Also when a user changes the page size value from the preference when the user is in the result screen, the new preference value for the page size would be honored only when the user re submits the query.

5. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

6. Document History

Version	Date	Description of changes	Author/Reviewer
0.1	20-02-2015	Initial Version	Rahul c
0.2	24-02-2015	Reviewd	Suvasish Samaddar

3.3.4 Domain Layer

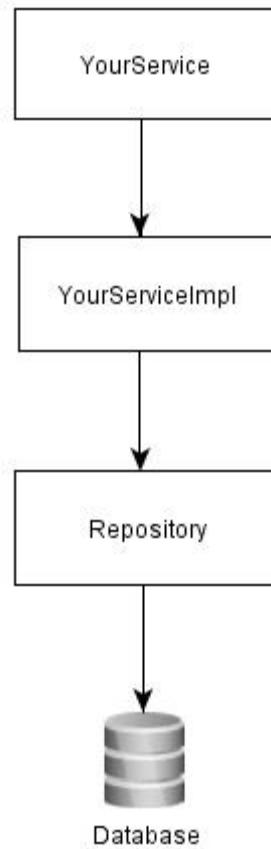
3.3.4.1 Service

Introduction

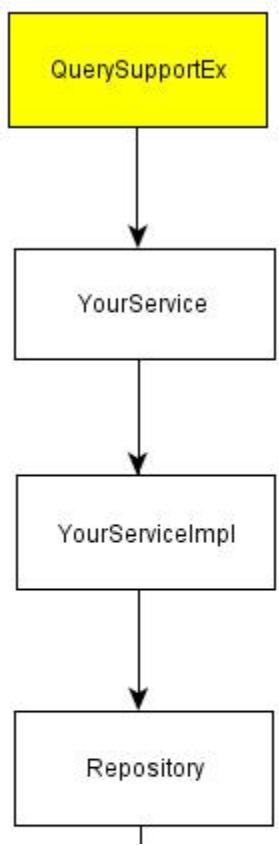
On top of the persistence layer sits the service layer of the application. Its role is to co-ordinate the domain objects and repositories in order to perform higher level operations. This is also the layer which provides transaction demarcation. One consideration for which transaction demarcation should be done at service level is to ensure that the changes made by service operations are atomic. Otherwise, concurrent operations may leave the application data in an inconsistent state.

Writing a Service

There are two flows in finch



Flow-1





Database

Flow-2

Note:-Yellow color blocks are finch provide.

We will go for the Flow-2 if we want to take advantage of [pre-fetching](#). Generally we follow Flow-1.

First write an Interface and declare your methods.

Next implement that Interface and annotate service class with `@Service`. Write your Business logic there.

```
@Service  
public class CompanyServiceImpl implements CompanyService {  
...  
}
```

For spring to process annotations, add the following lines in your application-context.xml file.

```
<context:annotation-config />  
<context:component-scan base-package="...specify your package name..." />
```

Transaction Management

The `@Transactional` annotation is metadata that specifies that an interface, class, or method must have transactional semantics; for example, “start a brand new read-only transaction when this method is invoked, suspending any existing transaction”. The default `@Transactional` settings are:

- The propagation setting is `PROPAGATION_REQUIRED`.
- The isolation level is `ISOLATION_DEFAULT`.
- The transaction is read/write.
- The transaction timeout defaults to the default timeout of the underlying transaction system, or `none` if timeouts are not supported.
- Any `RuntimeException` will trigger rollback, and any `checkedException` will not.

For enabling Transactional annotation following lines should be added in datasource-context file

```
<tx:annotation-driven transaction-manager="transactionManager"/>
```

Revision History

Version	Date	Description of Change	JIRA Issue ID	Author
1.0	05.08.2014	Created initial version		sanjayv

3.3.4.2 Repository

Introduction

A repository basically works as a mediator between our business logic layer and our data access layer of the application. Sometimes, it would be troublesome to expose the data access mechanism directly to business logic layer, it may result in redundant code for accessing data for similar entities or it may result in a code that is hard to test or understand. To overcome these kinds of issues, and to write an Interface driven and test driven code to access data, we use Repository. The repository makes queries to the data source for the data, thereafter maps the data from the data source to a business entity/domain object, finally persists the changes in the business entity to the data source. A repository separates the business logic from the interactions with the underlying data source or Web service. The separation between the data and business tiers has three benefits:

- It centralizes the data logic.
- It provides a substitution point for the unit tests.

Writing a Repository

Note:-Yellow color blocks are finch provide.

In finch, repository should extend RepositoryBase. In RepositoryBase there are some methods which are very useful -

write - Writes an entity into the persistent store.

update - Updates an entity in the persistent store.

find - Finds an entity by its primary key.

delete - Deletes an entity from the persistent store.

lock - Locks an entity in the persistent storage.

Annote your repository with @Repository.

```
@Service  
public class CompanyRepository extends RepositoryBase {  
    ...  
}
```

For spring to process annotations, add the following lines in your application-context.xml file.

```
<context:annotation-config />  
<context:component-scan base-package="...specify your package name..." />
```

Inject entity-manager using @PersistenceContext annotation

```
@PersistenceContext(unitName = "sample-em")  
private EntityManager em;
```

where sample-em is your entity-manager defined in datasource-context.xml file.

Revision History

Version	Date	Description of Change	JIRA Issue ID	Author
1.0	07.08.2014	Created initial version		sanjayv

3.3.5 Concepts

3.3.5.1 Application Date

Application Date

Date which is assumed to be the current date to the application. This might be different from the actual system date. For example, when the system date is 15th December, 2013 the application date might still be 14th December, 2013. So application assumes the current date to be 14th December, 2013 and not 15th December, 2013.

Requirement

Back-office processing for all the transactions for a day might not be completed by the time change of day takes place (midnight). Remaining transactions, which complete after the change of day, are considered as the transactions of the next business day, which is not desired. To take care of this type of scenario, the concept of Application Date has been introduced. Application Date is changed to the next business date only when the process execution of all the transactions for a particular day complete.

Storing Policy

Application date is stored based on the combination of Enterprise ID, Branch ID, Component ID and Application Mode in INF_APPLICATION_DATE.

ENTERPRISE_ID	BRANCH_PK	COMPONENT_ID	APP_MODE	DESCRIPTION	STATUS	APPLICATION_DATE
FIN	2	INF	OL	FIN_INF_OL	Y	25-07-13
FIN	2	INF	BT	FIN_INF_BT	Y	20-07-13
FIN	2	INF	SV	FIN_INF_SV	Y	18-07-13

The APP_MODE mentioned in the above table are as follows:

- OL : Online
- BT : Batch
- SV : Service

INF_APPLICATION_DATE table has a unique constraint that consists of four columns ENTERPRISE_ID, BRANCH_PK, COMPONENT_ID and APP_MODE. BRANCH_PK column is nullable but all other three columns are mandatory. So application can set BRANCH_PK optionally.

How it works

Please note that finch sets the default branch id in the Operation context when user is authenticated in web and console. So if application needs to set different branch id then application has to set it properly using Spring Security filter etc. in case of web.

Now application can have application date in various combinations like

Case	Set for every branch	Set for some of branches	Set for enterprise
1	No	No	Yes
2	No	Yes	Yes
3	Yes	No	Yes
4	Yes	No	No
5	No	No	No

For case #1, no row entry will be present in INF_APPLICATION_DATE table with BRANCH_PK set. finch will always resolves the enterprise level application date.

For case #2, finch will resolve application date of particular branch if branch id is present in Operation context otherwise it will resolve the enterprise level application date. And if no application date is found for branch provided then also it falls back to enterprise level application date.

For case #3, the behavior is same as case #2. Only difference is that for a given branch it can always resolve branch level application date.

For case #4, branch id should be present in Operation context always, otherwise it will fail to get the application date.

For case #5, this is the case where application does not use application date concept at all. Otherwise finch will always fail to get the application date.

In notification server, finch internally sets the default branch in Operation context.

If branch level application date is set then during running console batch, user id must be provided or default branch id should be configured in console.properties as batch.default.branchId. Otherwise console batch run as usual, these mention configuration is NOT required.

Use of Application Date is optional in finch.

Setting Application Date

Steps for using the Application date functionality have been stated below:

1. Create a file named global-config.properties and place it under classes\META-INF\config\global-config.properties
2. Set property as finch.global.appdate_supported=true in global-config.properties file
3. Need to make not null APP_REGI_DATE and APP_UPD_DATE fields of each table.

If the user don't want to use Application date functionality, need to follow the steps mentioned below:

1. Create a file named global-config.properties and place it under classes\META-INF\config\global-config.properties
2. Set property as finch.global.appdate_supported=false in global-config.properties file.
3. Need to make null APP_REGI_DATE and APP_UPD_DATE fields of each table.

Fetch Application date from DB

```
ApplicationDateService appDateService = Operation.getInstance().getReference(ApplicationDateService.class);
Date appDate = appDateService.getApplicationDate(getComponent());
```

Assumptions

finch does not validate any enterprise or branch passed as parameter for getting application date. finch directly searches with the given parameters; if application date not found with the given parameter then it fall back to enterprise specific application date setting.

Fall back

finch provides two fall back for resolving application date

1. If no branch id is provided (null or empty branch id) then it searches for application date configuration in which BRANCH_ID is NULL.
2. If branch id is provided (neither NULL nor empty) then it first searches with provided parameters; if no application is found then it again searches for application date configuration in which BRANCH_ID is NULL.

3.3.6 Batch/Console Service

Introduction

In software many task gets done using console service mainly those feature where interaction with user almost not required except

per-configured some input data. e.g. nightly some consolidation work, end user report creation , process to create data for enterprise for next day decision, over all status of data etc...

So finch infrastructure provides infrastructure for same so application will use the service as general programming required to fetch and calculate the data with configuration, rest infrastructure will take care.

And target this confluence and linked is to provide information to design application utilising the console services provided by finch.

Console Service

Provides following feature to facilitate the application:

[Console Service](#) => Continuous Console service e.g. service controller, Message routing service etc.

Linked page includes following:

1. Role of finch
2. Camel Overview
3. Development steps using Camel
4. Development Steps using finch
5. Database Setup
6. Common Configurations

Some services provided by finch are ServiceController, ListRegistry, StopService, ServiceLauncher, SchedulerService etc.

[Batch Service](#) => As and when required Console Service. In the linked page have look on following

1. Common Setup to develop a Batch Process for developing application specific service
2. Common Configurations to Run Batch Job for configuration
 - a. Creating a Batch Job
 - b. Data Setup for Batch Job for data setup in Database
 - c. Steps to Use Application provided Settings for Batch process
3. Run Batch Runner both for Console and UI
 - a. Running the Batch Job Using Console
 - b. Steps to Run Batch Runner both for Console and UI

[Report](#) => Report Creation service and linked page provides following information:

1. Console Based Report Development Steps
2. Run Console Report
3. Use of Tables for Report : Database table where report configuration e.g file format (pdf/xlf.csv/etc.) , file name pattern., report id etc are stores which report service utilizes at run-time.
4. Relationship Between Tables : shows following quick look
 - a. Information directory where report get stored for further processing,
 - b. Enterprise/branch for which report
 - c. participant of report
 - d. etc

[3.4.6.1.3 Scheduler](#) => Scheduling a job service and linked page provides following information:

1. Configuration
2. Create Job to Schedule
3. Schedule the Job
4. Schedule a Routine Job
5. Un-schedule a Job -> For removing scheduled job.
6. API: Scheduler Methods
7. Console Usage
 - a. Configuration
 - b. Scheduler Console Service
 - c. Scheduler Job Manager
 - d. Using the Console Process Executor Job
 - e. Specifying Base Directory Path for Console Executable

Revision History

Version	Date	Description of Change	JIRA Issue Id	Author
1.0	30-Jul-2014	Initial Version	FCHII-730	Ashok Kumar Jha

3.3.6.1 Console Service Process

1. Introduction

This document describe how to write a batch as service using finch framework.

2. Background

finch provides an abstraction for developing console based Service or daemon processes. finch abstraction takes care of life cycle management of service. When a service is started, it first register itself with service registry. The service registry is maintained by a console process called ServiceController. When a service process is stopped it deregister itself from the service registry. The registration and deregistration functions are implemented by finch service abstraction.

3. Steps to write and run a batch as a service

3.1. Extends AbstractConsoleService class

First application need to define class that will inherit from com.nrft.finck.inf.console.service.AbstractConsoleService class. Then within the newly created class application need to override service() method. Inside the overridden service method application can write its own functionality.

3.2. Add entry in console-executable-mapping.properties

Next application need to add entry in console-executable-mapping.properties file with fully qualified name. Following example shows such declaration

```
DemoBatchService=<fully-qualified-package-name>.<class-name>
```

3.3. Run the batch

Next step is to run the batch as service. For this, first start ServiceController using launch.bat file, then run bat using following command (DemoBatchService name has been used here as example, name will depends on application)

```
launch.bat DemoBatchService
```

4. Reference

More information about service process can be found [here](#)

5. Document History

Version	Date	Description of Changes	Author
1.0	13-02-2015	Initial document	Prasun Pal

3.4 Implementation Phase

3.4.1 Getting started

Introduction

Application developers can use the finch framework to develop different features for their application. The subsequent sections (as listed below) contains the detailed information which are relevant to the Application developers in order to use the finch framework for application development:

- Prerequisite - Software that need to be installed on any operating system
- Environment Setup - Steps to setup application development environment.
- Blank Application Setup - Application setup
- Initial Data Setup for finch - Describes the detailed information about initial data setup which is required to use the finch framework.
- Menu Setup - Describes the detailed information about initial data setup which is required to use the finch framework.
- CRUD Generation Tool - The tool which provides a convenient way to generate several components related to the CRUD (Create, Read, Update & Delete) operation.

3.4.1.1 Prerequisite

Introduction

This section details the primary and additional Software requirement which are required for using the finch framework for Application development.

Software Requirement for Using finch Framework

Refer to [finch Software Requirement](#) section which details the software dependencies of using finch framework in Application development.

finch recommends Apache Maven for building application based on finch. Apache Maven downloads all the dependent libraries from Maven repository. Developers need not install or download the libraries manually.

Additional Software Requirement for Using finch Framework

This section describes the additional software which are required for development environment in order to use the finch framework for Application development.

JDK

finch server side code or back-end code is based on Java programming language. Application developer must install Java Standard Edition 1.6.x in the development environment.

Ensure about the followings:

- JAVA_HOME environment variable is properly set. It should point to the base directory where Java Standard Edition (JDK) is installed.
- '%JAVA_HOME%'\bin included in the PATH environment variable.

Integrated Development Environment (IDE)

In the Application development, developers can use any standard IDE for Java like Eclipse, Netbeans or IntelliJ. In this manual, the guideline is provided for Eclipse environment. It is recommended to use Eclipse Kepler version for Application development.

Build Tool

finch recommends Apache Maven for application building. finch uses 3.x version of Apache Maven.

Ensure that M2_HOME environment variable is properly set. It should point to base directory where Apache Maven is installed.

Application Server

finch recommends Apache Tomcat as Application Server for developer testing. finch has been tested successfully on Apache Tomcat version 6.x and 7.x.

Database Server

finch has been tested successfully with MySQL 5.6.x and Oracle 10g/11g. Application developers can use any of the mentioned Database Server for development.

Messaging Product

finch recommends Apache ActiveMQ as the messaging solutions for applications those required Message Service for development environment. For production environment, the Application should decide about the messaging solution based on its requirements.

Apache ActiveMQ is a free software and very lightweight. .

3.4.1.2 Set-up Blank Application using finch

Introduction

This page describes how to set up an empty or blank web application based on finch framework. The application will have a default Dashboard and a set of Screens provided by finch framework out of the box.

Prerequisite

Following software must be installed in the machine where the blank application will be built:

- Java SE 6 or above
- Apache Maven 3.x.x
- Apache Tomcat application server 6.x or above

Make sure JDK bin and Maven bin directories are present in your PATH environment variable. Also, HTTP Proxy and non proxy hosts are properly set-up in your Maven settings.xml file.

Steps to Build Blank Application Using finch

To build a blank application using finch framework, follow the steps mentioned below:

1. Open a command prompt and go the directory where you want to build the application.
2. Execute following command to generate folder structure based on finch provided maven archetype for blank application:
`mvn archetype:generate -DarchetypeGroupId=com.nrft.finck -DarchetypeArtifactId=finch-blank-archetype -DarchetypeVersion=[finch-version] -DgroupId=[Java root package for Application] -DartifactId=[Application Name] -Dversion=[Application version] -DarchetypeRepository=http://mavenrepo.nrifintech.com/finch/release`

For example:

```
mvn archetype:generate -DarchetypeGroupId=com.nrft.finck -DarchetypeArtifactId=finch-blank-archetype -DarchetypeVersion=2.1.2.0 -DgroupId=com.nri.app -DartifactId=finch-app -Dversion=1.0-SNAPSHOT -DarchetypeRepository=http://mavenrepo.nrifintech.com/finch/release
```

3. The last command creates a directory whose name is same as the value of artifactId in the above command (finch-app in the example above). Go to this directory.
4. Please follow "Configuration for running finch using MySQL database" OR "Configuration for running finch using Oracle database" section (as per your requirement), to configure required files.
5. Execute mvn package command in the directory created after execution of step 2. After successful execution of this command a war file will be created under target folder of the application distribution directory (For the above example command it would be finch-app-dist¥target¥finch-blank.war)
6. Deploy the war in the Tomcat application server. To deploy the war copy the war file under '[TOMCAT-HOME]'¥webapps' folder.

Import Project in Eclipse

To import the project in Eclipse do the following steps

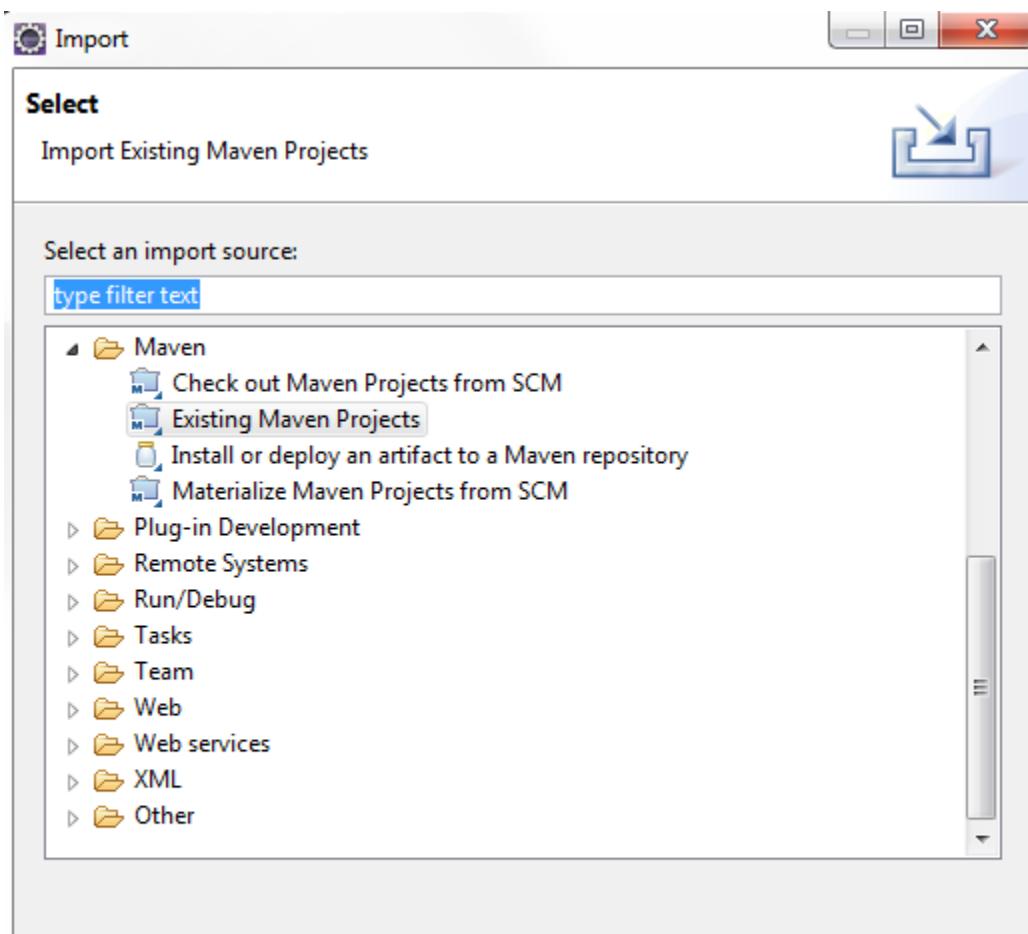
Eclipse

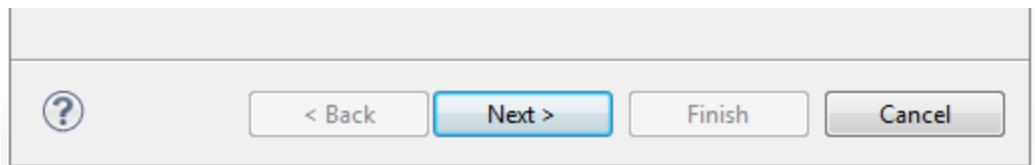
Assuming that you have properly installed Eclipse Kepler in your machine

Step 1: Open command prompt and go to the directory where you downloaded the blank application using mvn command. (For the above example <dir>/finch-app)

Step 2: Execute the command `mvn eclipse:eclipse`

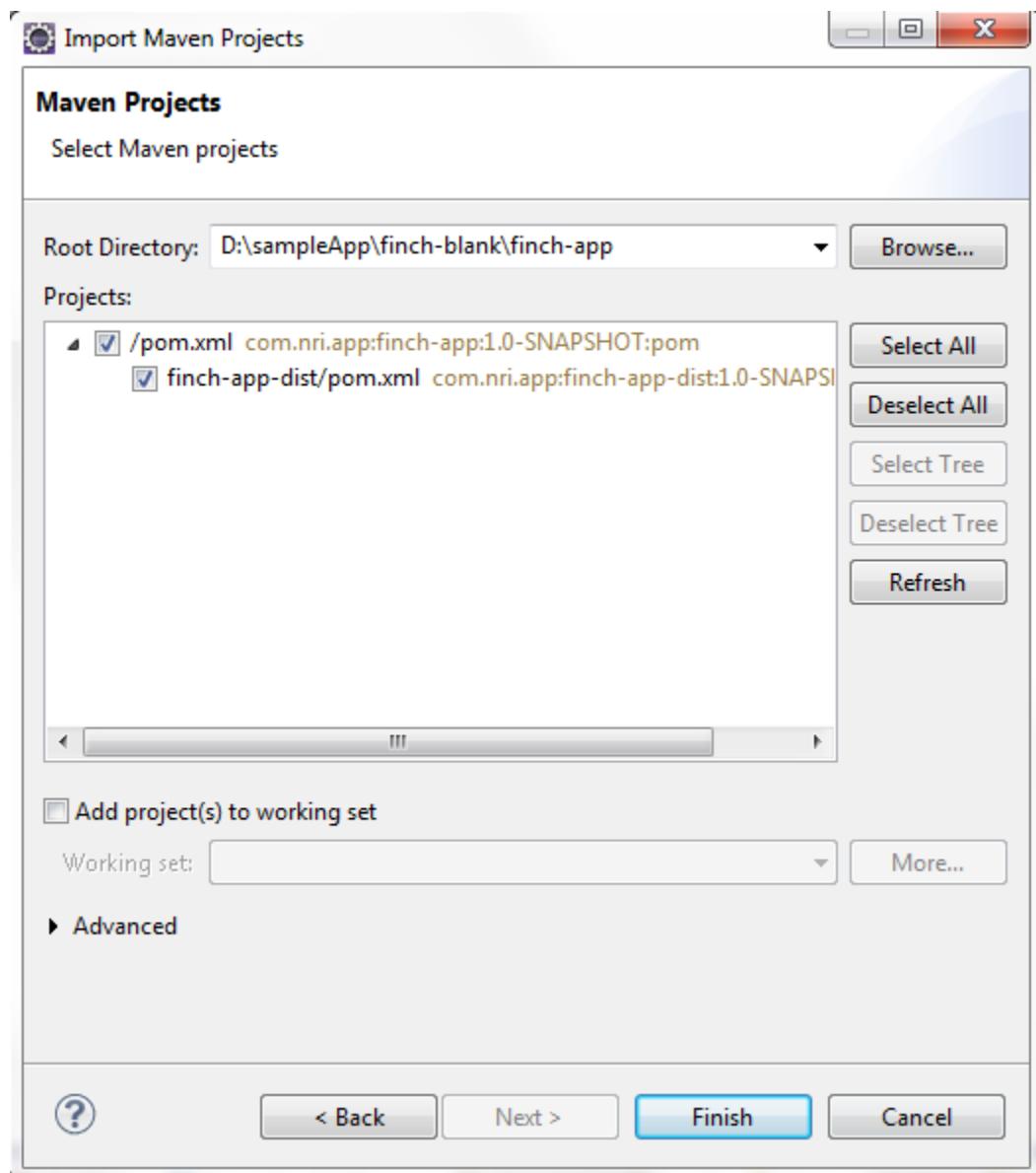
Step 3: Now open Eclipse and go to File -> Import





Step 4: Select Existing Maven Project and Click Next button.

Step 5: Click the Browse button and select the directory (e.g. <dir>\finch-app)



Step 6: select all item(s) and click Finish button

Some Useful Maven Commands

Following are the maven command

1) mvn eclipse:eclipse

This command build the project into eclipse project like structure. Used before importing project in eclipse.

2) mvn eclipse:clean

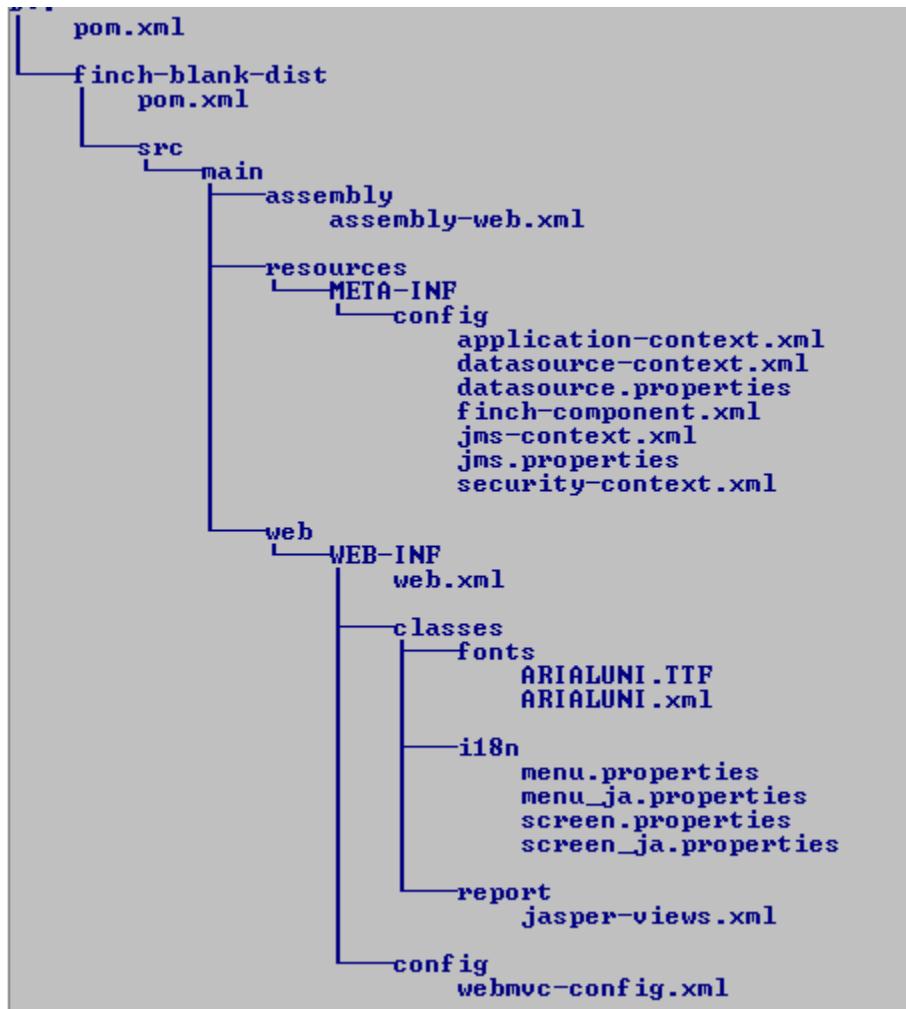
This command deletes the .project, .classpath, files used by Eclipse. Used when you need to clean those files.

3) mvn package

Compile the code and also package it. For blank application it will create a war file and store it in target folder . for example <dir>/finch-app/finch-app-dist/target.

Used after any change in blank application.After any code modification run this command.

Folder/Structure of the Blank Application



Create a directory finch-blank. The above folder structure should be created under this newly created directory.

Maven Files

Root POM File Under finch-blank¥finch-app folder

The POM should define repository where finch artifacts are delivered.

finch Maven Repository

```
<repository>
<id>nrifintech.finchrepo</id>
<name>finch Repository</name>
<url>http://mavenrepo.nrifintech.com/finch/release</url>
</repository>
```

The POM should define dependency on finch modules and other libraries like JDBC driver, JMS implementation etc.

Maven Dependencies

```
<dependency>
<groupId>com.nrft.finch</groupId>
<artifactId>finch-core</artifactId>
<version>${finch.version}</version>
</dependency>
```

Make sure that the value of finch.version in the root POM is the same as the value of -DarchetypeVersion given in the command (2.1.2.0 in the above example). If it is not same then change it. This value determines which public release of finch your blank application will depend on.

```
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<finch.version>2.1.2.0</finch.version>
<!-- other properties -->
</properties>
```

The root POM should include all sub-modules. For the finch blank application we need only distribution module to build the web archive(war)

modules

```
<modules>
<module>finch-blank-dist</module>
</modules>
```

POM File Under Distribution (finch-blank-dist) Module

The distribution module POM defines two main tasks, discussed below:

1. Unpacks or unzip finch web artifact under build directory

Unpack finch web artifact

```
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-dependency-plugin</artifactId>
<version>2.5.1</version>
<executions>
<execution>
<id>unpack-finch-blank-dist</id>
<phase>package</phase>
<goals>
<goal>unpack</goal>
</goals>
<configuration>
<artifactItems>
<artifactItem>
<groupId>com.nrift.finch</groupId>
<artifactId>finch-dist</artifactId>
<version>${finch.version}</version>
<classifier>web</classifier>
<type>zip</type>
<overWrite>false</overWrite>
<outputDirectory>${project.build.directory}/finch-web</outputDirectory>
</artifactItem>
</artifactItems>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
```

2. Define a maven profile to build war file using maven assembly file.

Maven war profile

```
<profile>
<id>web</id>
<activation> <activeByDefault>true</activeByDefault> </activation>
<build>
<plugins>
<plugin>
<artifactId>maven-assembly-plugin</artifactId>
<version>${assembly-plugin.version}</version>
<executions>
<execution>
<phase>package</phase>
<goals> <goal>single</goal> </goals>
<configuration>
<finalName>finch-blank</finalName>
<appendAssemblyId>false</appendAssemblyId>
<descriptors>
<descriptor>src/main/assembly/assembly-web.xml</descriptor>
</descriptors>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</profile>
```

Maven Assembly File to Build War File

Path and name of this file is mentioned in the distribution POM file. This file copies files from unpacked finch web artifact and copies files from distribution module. It creates a war file including all copied files.

In the file finch-app\finch-app-dist\src\main\assembly\assembly-web.xml add the following

```
<dependencySets>
<dependencySet>
<useProjectArtifact>false</useProjectArtifact>
<scope>runtime</scope>
<outputDirectory>/WEB-INF/lib/</outputDirectory>
<includes>
<include>org.apache.activemq:activemq-camel</include>
<include>org.apache.activemq:activemq-pool</include>
<include>org.apache.activemq:activemq-core</include>
<include>org.apache.geronimo.specs:geronimo-j2ee-management_1.1_spec</include>
</includes>
</dependencySet>
</dependencySets>
```

Other Files

The distribution module also need to have few other files apart from maven related files. Files and path of the files can be found in the 'Folder/Directory Structure' section.

Database Setup

To setup the database:

1. Setup a Database schema using the script provided with the last [release note](#) of finch.

Currently, finch provides database scripts for following database types. Once database schema setup for finch is completed, refer to [Initial data set-up section](#) for more details about initial database set-up. In case of default sample scripts needed for database setup refer to the section sample database script of [Initial data set-up section](#)

- Oracle
- MySQL

The script zip file contains script for following folders.

- COMMON
- CONSOLE
- WEB
 - INF
 - DBD

The script running sequence are as follows:

- Application that uses only finch console part should first run scripts under COMMON folder and then CONSOLE folder.
- Application that uses only finch web part should first run scripts under COMMON folder and then WEB folder.
- Application that uses both finch console and web part should first run scripts under COMMON folder, then CONSOLE folder, then WEB/INF folder and at last the scripts under WEB/DBD folder.

Each of the above-mentioned folder contains two sub-folders:

- DELTA - this folder contains scripts for existing application having finch data model up to release prior to this release.
- TAR - this folder contains scripts for new application that has no existing data model setup for finch.

Note: Please follow "Database Setup for Sample App" section under [this link](#) for MySql Setup.

Configuration for running finch using MySQL database

1. Change the "Place Holder" values in "datasource.properties" file under the `finch-blank\finch-app\finch-app-dist\src\main\resources\META-INF\config` folder.

For example:

```
jdbc.components.url.GLOBAL = jdbc:mysql://localhost:3306/finch?characterEncoding=UTF-8
jdbc.components.userName.GLOBAL = finch_dev
jdbc.components.password.GLOBAL = finch_dev
```

[NOTE: It is not recommended to use 'finch_dev' schema]

2. Download configuration files(zipped) for MySQL : [mysql-configuration-files.zip](#)

Extract below mentioned files from the above zipped file and replace the same files in the following directory "`finch-blank\finch-app\finch-app-dist\src\main\resources\META-INF\config`"

`datasource-context.xml`

3. Extract MySQL Database Driver JAR file (`mysql-connector-java-5.1.27.jar`) from `MySQL-Database-Driver.zip` file attached below, and place `mysql-connector-java-5.1.27.jar` in the container's 'lib' folder.
For example: If application is deployed in tomcat, then JAR file need to be place in `<tomcat_home>\webapps\finch-blank\WEB-INF\lib` folder.

MySQL Connector JAR: [MySQL-Database-Driver.zip](#)

Tomcat Server Configuration for running finch using MySQL database

When finch is deployed in Tomcat 7 and it is used with MySQL database, an additional configuration needs to be done `server.xml` file of Tomcat

```
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener"  
driverManagerProtection="false"/>
```

Configuration for running finch using Oracle database

Please change the "place holder" values in "datasource.properties" file under the following folder : `finch-blank\finch-app\finch-app-dist\src\main\resources\META-INF\config` -

Do note that, a sample "datasource.properties" file is attached here:

Configuration file: [datasource.properties](#)

3.4.1.3 finch Deliverables

Introduction

This page provides information about finch deliverable.

Deliverable

For regular release, like sprint release, finch delivers following artifacts:

- Binary artifacts
- Database script

Binary Artifacts

finch binary artifact includes Java jar files for the finch modules listed in the following table:

Module	Description
finch-core	Parent artifact core functionality
finch-console-infra	Console infrastructure including service abstraction related classes
finch-batch	Batch related classes
finch-notification	Push framework
finch-scheduler	Scheduler
finch-sso	Single Sign On
finch-upload	Excel/CSV upload
finch-web-infra	All web infrastructure
finch-tools	CRUD tool
finch-test	finch integration testing framework
finch-dbd	Dashboard

Apart from the above-mentioned jar files, finch also provides a consolidated zipped file for web based application and console base application. These files are placed under finch-dist artifact and file names are.

- `finch-dist-<release-version>-web.zip`
- `finch-dist-<release-version>-console.zip`

finch Maven Repository

All finch binary artifacts are placed in a Maven repository. The URL of the repository is: <http://mavenrepo.nrifintech.com/finch/releases/>

Database Script

finch provides database script for all finch releases. The database scripts are provided along with release note for the release in form of zip file. Currently, finch provides database scripts for following database types.

- Oracle
- MySQL

The script zip file contains script for following folders.

- COMMON
- CONSOLE
- WEB

Script Running Sequence

The script running sequence are as follows:

- Application that uses only finch console part should first run scripts under COMMON folder and then CONSOLE folder.
- Application that uses only finch web part should first run scripts under COMMON folder and then WEB folder.
- Application that uses both finch console and web part should first run scripts under COMMON folder and then CONSOLE and at last the scripts under WEB folder.

Each of the above-mentioned folder contains two sub-folders:

- DELTA - this folder contains scripts for existing application having finch data model up to release prior to this release.
- TAR - this folder contains scripts for new application that has no existing data model setup for finch..

3.4.1.4 Initial data set up for finch

1. Introduction

This document describes initial data set up required to start an application based on finch framework.

2. Data Model

The following diagram displays finch tables required for initial data set-up and relationship among those tables.

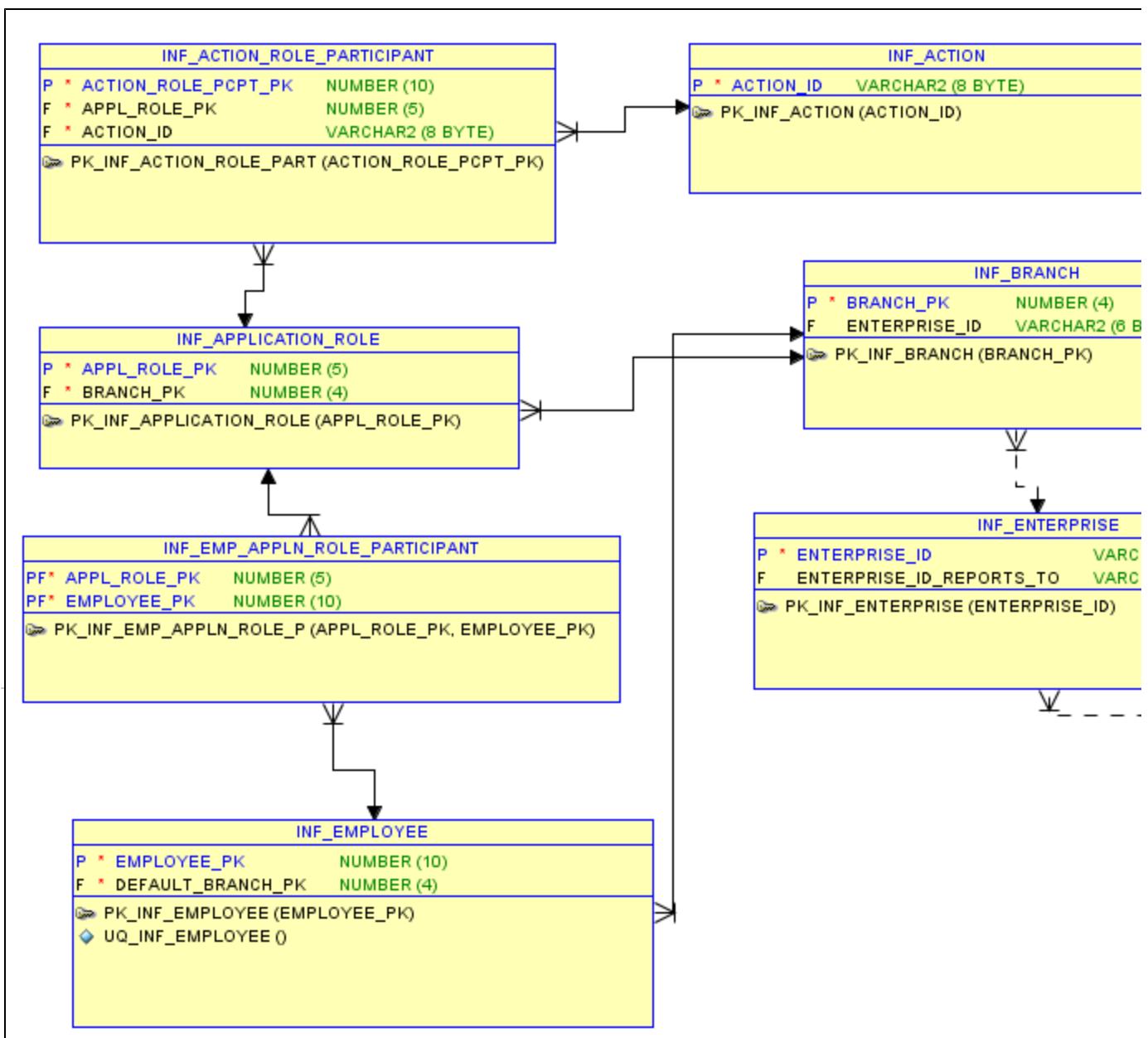


Table	Description
INF_COMPONENT	Provides data set-up for two components: <ul style="list-style-type: none"> INF (Infrastructure) DBD (Dashboard)
INF_CLOSING_STATUS	For each component, two records are created for two different application modes: <ul style="list-style-type: none"> OL (Online or Web Mode) BT (Batch or console mode).
INF_LANGUAGE	Records for JP (Japanese) and EN (English language).
INF_ACTION	Provides records for Finch provided actions like Dashboard actions, Employee Entry, Query, Amend cancel, Access Log etc.
INF_ACTION_CONSOLIDATION	Action consolidation records are provided for Employee Amend and Cancellation.

INF_UI_MENU	<p>finch provides menu set-up for the followings:</p> <ol style="list-style-type: none"> 1. Main Menu or Root Menu 2. Preference Data Import 3. Preference Data Export 4. Access Log Query 5. Employee Entry 6. Employee Query 7. Batch Execution 8. Batch Query <p>Parent of item 2 to 8 menus are set as item 1 i.e. main menu.</p>
INF_UI_SCREEN	Provides records for finch provided Screens like Employee Entry, Amend,Query etc.
INF_CONSTRAINT_LIST	Constraint indicates a logical name for which possible values are pre-defined. finch provides constraints for LANGUAGE, PAGE_SIZE, DATE_FORMAT etc.
INF_CONSTRAINT_VALUE	For each constraint, this table stores possible values for the constraint.

Dashboard

Table	Description
DBD_TASK_TYPE	Task types those appear in the task widget. Two task types are provided: <ul style="list-style-type: none"> • out-of-the-box - 'My Task' • 'Assigned By Me'
DBD_WIDGET	Provides 'Notification' widget as a system widget out-of-the-box.
DBD_WIDGET_TYPE	finch provides the following types of widgets: <ul style="list-style-type: none"> • Business Counter • Saved Query • Saved Template • Menu Shortcut • Notification • Chart • Task

4. Data Setup for Application Start-Up

In this section, we will be discussing a scenario where for an application we are going to set up the enterprises, branch offices, application role and user id, as listed in the following table:

Enterprise	Branch/Office	Application Role	User
NRI	NRI-HO	NRIADMIN	nriuser
CLIENT-A	A-TOKYO	ADMIN-A	admin-a
		NORMAL-TKY-A	tkyuser-a
CLIENT-B	B-TOKYO	ADMIN-B	admin-b
		NORMAL-TKY-B	tkyuser-b
	B-OSAKA	NORMAL-OSK-B	oskuser-b

The data setup, which is required in the application side for the above-mentioned scenario, are as follows:

1. Setup Enterprise – Insert the Enterprise information.
2. Setup Branch or Office for Enterprises – Insert the Branch or Office information.
3. Setup Application Role for Enterprises – Set up the application roles for the Enterprises.
4. Setup User – Insert the user related information.
5. Setup Menu Records – Set up the menu records.
6. Setup Screen Records – Set up the screen records.
7. Dashboard Setup – Setup the required access for the Widget types.

The subsequent sections describe the above-listed data setup steps in details.

Setup Enterprise

As per the mentioned scenario, 3 records should be inserted in the INF_ENTERPRISE table. Sample data for INF_ENTERPRISE table has been displayed in the table below:

Enterprise_ID	Mnemonic	Short_Name	Official_Name
NRI	NRI	NRI	NRI Enterprise
CLNTA	CLNTA	Client A	Client A
CLNTB	CLNTB	Client B	Client B

Note: It only shows the important columns, although the table contains more columns.

Setup Branch or Office for Enterprises

As per the mentioned scenario, 4 records should be inserted in the INF_BRANCH table. Sample data for INF_BRANCH table has been displayed in the table below:

BRANCH_PK	BRANCH_ID	ENTERPRISE_ID	MNEMONIC	SHORT_NAME	OFFICIAL_NAME	TIME_ZONE	STATUS
1	NRHO	NRI	NRIHO	NRI HO	NRI Head Office	Asia/Tokyo	NORMAL
2	TKYA	CLNTA	TKYA	Tokyo A	A Tokyo Office	Asia/Tokyo	NORMAL
3	TKYB	CLNTB	TKYB	Tokyo B	B Tokyo Office	Asia/Tokyo	NORMAL
4	OSKB	CLNTB	OSKB	Osaka B	B Osaka Office	Asia/Tokyo	NORMAL

Note: It only shows the important columns, although the table contains more columns.

Setup Application Role for Enterprises

As per the mentioned scenario, 6 records should be inserted in the INF_APPLICATION_ROLE table. Sample data for INF_APPLICATION_ROLE table has been displayed in the table below:

APPL_ROLE_PK	Application_Role_Name	Branch_PK	Status
1	NRIADMIN	1	NORMAL
2	ADMIN-A	2	NORMAL
3	NORMAL-TKY-A	2	NORMAL
4	ADMIN-B	3	NORMAL
5	NORMAL-TKY-B	3	NORMAL

6	NORMAL-OSK-B	4	NORMAL
---	--------------	---	--------

Note: It only shows the important columns, although the table contains more columns.

For each application role, we need to set-up data in the INF_ACTION_ROLE_PARTICIPANT. This table stores the relationship information between application role and actions. Depending on the requirement, a role should be given access to appropriate actions.

Setup User

In this step INF_EMPLOYEE and INF_EMP_APPLN_ROLE_PARTICIPANT to be updated.

INF_EMPLOYEE – Table Data Setup

Employee_PK	User_ID	Last_Name	First_Name	Title	Start_Date	Status	Mail
1	nriuser	Sato	Abc	Mr	09-OCT-12	NORMAL	nriuser@nri.co.jp
2	admin-a	Das	Amita	Ms	06-JAN-14	NORMAL	admin-a@clienta.com
3	tkyuser-a	Roy	Sumit	Mr	09-JAN-14	NORMAL	tkyuser-a@clienta.com
4	admin-b	De	Abani	Ms	14-JAN-14	NORMAL	admin-b@clientb.com
5	tkyuser-b	Kar	Bikash	Mr	10-JAN-14	NORMAL	tkyuser-b@clientb.com
6	oskuser-b	Sano	Kim	Ms	10-JAN-14	NORMAL	oskuser-b@clientb.com

INF_EMP_APPLN_ROLE_PARTICIPANT – Table Data Setup

Sample data set-up for INF_EMP_APPLN_ROLE_PARTICIPANT table.

APPL_ROLE_PK	EMPLOYEE_PK
1	1
2	2
3	3
4	4
5	5
6	6

Setup Menu Records

finch provides entries for menu in the INF_UI_MENU table. These menus are created with their parent menu as Main Menu. The Main Menu itself is a part of the menu provided by finch. It is the responsibility of the application to organize the menus in proper hierarchical manner by setting MENU_PARENT_PK of the finch provided menu records. Henceforth, the application will add application specific menus in this table, as per the requirement.

Setup Screen Records

finch provides data setup for finch specific screens in the INF_UI_SCREEN table. Henceforth, the application will add the application specific screens in this table, as per the requirement.

Dashboard Setup

finch provides table DBD_WIDGET_TYPE_APP_ROLE_PT for controlling access to the widget types (DBD_WIDGET_TYPE) based on application role (INF_APPLICATION_ROLE). Setup data in this table properly to give required access for widget type to an application role.

Sample Database Scripts

following is the db scripts to setup blank finch

DB	SCRIPT
Oracle	finch-blank-data-Oracle
MySQL	finch-blank-data-Mysql

This page contains:

- 1. Introduction
- 2. Data Model
- 3. DB Script
Provided
by finch
- 4. Data Setup
for Application
Start-Up

3.4.1.5 CRUD Generation Tool

Introduction

The CRUD generation tool of finch provides a convenient way to generate several components related to the CRUD (**C**reate, **R**ead, **U**pdate & **D**elete) operations. Components like JPA Entities, Service, Repositories, Controllers, JSNX can be generated by the user with minimal effort. The tool reads the database schema and generates the classes accordingly. User needs to provide a file in JSON (**J**ava**S**cript **O**bject **N**otation) format, containing several information, which the tool reads in order to generate the classes.

The files generated by CRUD only contains the skeleton of the codes. The application developer needs to incorporate the business logic and other implementation in the generated files.

JPA (Java Persistence API) Entities, Services and Repositories

The following snippet of code in the JSON file is used only for generating the JPA Entities, Services and Repositories.

CRUD_input.json

```
{  
  "superclass": "com.nrift.fin.ch.dbd.domain.entity.BaseDbdEntity",  
  "excludeColumn":  
    "APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK",  
  "componentList": "INF,DBD,TRD",  
  "defaultPackage": "com.nrift.sample",  
  "baseDirectory": "D:\$\$CRUD",  
  "dialect": "org.hibernate.dialect.Oracle10gDialect",  
  "driverClass": "oracle.jdbc.driver.OracleDriver",  
  "url": "jdbc:oracle:thin:@172.16.29.92:1521:d11gr21",  
  "userName": "finch_sampleapp",  
  "password": "finch_sampleapp",  
  "defaultSchema": "FINCH_SAMPLEAPP",  
  "defaultCatalog": "*",  
  "includeTable": "",  
  "excludeTable": "",  
  "aggregateRoot": [  
    {  
      "component": "TRD",  
      "name": "TRADE"  
    },  
    {  
      "component": "DBD",  
      "name": "WORKSPACE,WIDGET"  
    }  
  ],  
  "transactionalUnit": "sample-tm",  
  "entityManagerUnit": "sample-em",  
  "queryCriteria": [],  
  "entryCriteria": []  
}
```

The above snippet code is an example of the JSON file, which needs to be provided as an input for running the tool.

The following table lists the technical terms used in the above snippet of code and their description:

Term	Description
superclass	Defines the base class which each generated entity must extend. This property is optional. In case the entities does not extends any superclass, leave it blank. It generates the same class in every module under entity folder. The user needs to rename the class based on module name
excludeColumn	Defines the column names which need to be excluded from each entity. This property is optional. In case the entities does not extends any superclass, leave it blank.
componentList	Defines the list of components present in the application. This is used to generate the entities truncating the component part from the table name. This property is optional.
defaultPackage	Defines the default package which gets imported in each generated tool like entity,repository,command form etc. The component name and the literal "domain.entity" is appended for entity with the provided default package value. For example, the above value for default package property for component TRD will yield the package com.nrift.sample.trd.domain.entity declaration in generated entities. The default property is com.nrift.fin.ch.
baseDirectory	Defines the output directory where the tools will be generated. For example, the above value for base directory property for component TRD will yield entities under D:/entities/com/nrift/sample/trd/domain/entity. This property is mandatory.

dialect	Defines the database dialect.This property is mandatory. For MySql, it should be org.hibernate.dialect.MySQL5Dialect
driverClass	Defines the driver name for the database connection.This property is mandatory. For MySql, it should be com.mysql.jdbc.Driver
url	Defines the database URL.This property is mandatory.
userName	The username of the database connection.This property is mandatory.
password	The password of the database connection.This property is mandatory
defaultSchema	The schema name from which the entities will be generated.This property is mandatory.
defaultCatalog	The catalog for the database schema.This property is mandatory. For MySql, it should be same as the schema name
includeTable	The table for which the crud tool should be generated.
excludeTable	The table for which the crud tool should not be generated. <div style="border: 1px solid red; padding: 10px; margin-top: 10px;"> Note: <ul style="list-style-type: none"> The input for includeTable and excludeTable can vary from single value,comma separated values, wildcard character '*' etc. Both includeTable and excludeTable contents cannot co-exist at the same time,i.e, entry for includeTable and excludeTable should not be present in the json at the same time. </div>
aggregateRoot	This defines the root table name for each module. The controller, service, repository will be generated only for this root table name. It can contain comma separated values
transactionalUnit	The transactional value required for any transactional method.
entityManagerUnit	The persistence context value used in service. <div style="border: 1px solid red; padding: 10px; margin-top: 10px;"> Note <ul style="list-style-type: none"> The attribute 'transactionalUnit' and 'entityManager' are required for generating service and repositories. In case only JPA entites are required, the value for these attribute may be left blank. The property name in json must be in lower case. For example, if database column name is TRADE_CCY_PK, then the input property should be tradeCcy, for TRADE_TYPE, it must be tradeType. </div>
queryCriteria	The criteria specified during the generation of query templates. The empty array denotes that the query components will not be generated. For more please follow <ul style="list-style-type: none"> READ (Query Template)
entryCriteria	The criteria specified during the generation of entry templates.The empty array denotes that the entry components will not be generated.For more please follow <ul style="list-style-type: none"> CREATE (Entry Template)

Generated Entities:

The CRUD tool create Entities(.java files) for all associated tables found in the given schema. It is recommended that the table name must contain the module name as prefix to create a proper project structure. For example, table TRADE should be present in database as TRD_TRADE while TRD denotes the module.

Important

- 1) Since the tool generates entities for all related tables, entities similar to the existing ones may be generated. It is advised to ignore the existing entities and copy only the required entities. For example, if TRD_TRADE has a dependency on INF_EMPLOYEE, the tool will generate entity as Trade.java and Employee.java (already provided by finch). In this case we have to ignore the generated Employee.java as it is provided by finch.
- 2) The CRUD tool also generates corresponding .class files for entities only. Please ignore them and copy only the .java files.
- 3) Entities generated within the same package does not import other related entity. In case few selected entities needs to be moved to a different package, related entities must be imported manually.

Sample entity generated from the above json : [Trade.java](#), [TradeTaxFeeAmount.java](#), [TradeTaxFeeAmountId.java](#)

Generated Services

Sample service generated from the above json : [TradeService.java](#), [TradeServiceImpl.java](#)

Generated Repository

Sample repository generated from the above json : [TradeRepository.java](#)

To run the tool:

```
D:\Test\sample-console\bin>launch FinchGenerateCRUD -f "D:\\final.json"

=====Generating entites=====
=====Entites generation completed=====

=====Generating Query Result View=====
=====Query Result View generation completed=====

=====Generating Repository=====
=====Repository generation completed=====

=====Generating Query Criteria Builder=====
=====Query Criteria Builder generation completed=====
```

```
=====Generating Service=====
=====Service generation completed=====

=====Generating Query Command Form=====
=====Query Command Form generation completed=====

=====Generating messages.properties=====
=====message.properties generation completed=====

=====Generating Query Controller=====
=====Query Controller generation completed=====

=====Generating Entry Command Form=====
=====Entry Command Form generation completed=====

=====Generating Entry Controller=====
=====Entry Controller generation completed=====

=====Generating Amend Command Form=====
=====Amend Command Form generation completed=====

=====Generating Amend Controller=====
=====Amend Controller generation completed=====

=====Generating views.xml=====
=====views.xml generation completed=====

=====Generating Query JSPX=====
=====Query JSPX generation completed=====

=====Generating Entry JSPX=====
=====Entry JSPX generation completed=====

=====Generating Project structure=====
```

Code Snippets for Creating Templates

The subsequent pages describe the code snippet of creating the following templates:

- CREATE (Entry Template)
- READ (Query Template)
- UPDATE (Amend Template)

3.4.1.5.1 CREATE (Entry Template)

Introduction

The entry criteria should be defined in the following way to generate the controller, command form, jspx ,and views.xml.

Code Snippet

The following code snippet is for creating the CREATE (Entry Template).

CRUD_input.json

```
//other stuffs
"entryCriteria": [
{
  "component": "TRD",
  "baseEntryUrl": "/trd/entry",
  "baseAmendmentUrl": "/trd/amend",
  "baseCancelUrl": "/trd/cancel/*",
  "wizards": [
    {
      "modes": [
        {
          "name": "EDIT",
          "includeButtons": [
            "PREVIOUS", "NEXT", "RESET", "SUBMIT", "TEMPLATESAVE"
          ],
          "pages": [
            {
              "index": 1,
              "pageName": "tradeGeneralEntry",
              "excludeButtons": [
                "PREVIOUS", "SUBMIT"
              ],
              "entryField": [
                {
                  "type": "Dropdown",
                  "name": "tradeCcy",
                  "mandatory": true,
                  "constraintName": "CURRENCY",
                  "position": 1
                },
                {
                  "type": "Date",
                  "name": "tradeDate",
                  "mandatory": true,
                  "position": 2
                }
              ]
            },
            {
              "index": 2,
              "pageName": "tradeDetailsEntry",
              "excludeButtons": [
                "NEXT"
              ],
              "entryField": [
                {
                  "type": "Textbox",
                  "name": "netAmount",
                  "mandatory": true,
                  "position": 1
                },
                {
                  "type": "Textbox",
                  "name": "externalReferenceNo",
                  "mandatory": false,
                  "position": 2
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```
        }
    ],
},
{
  "name": "USER_CONFIRMATION",
  "includeButtons": [
    "BACK","CONFIRM"
  ],
  "pages": [
    {
      "index": 1,
      "pageName": "tradeGeneralDetail",
      "excludeButtons": [],
      "entryField": [
        {
          "name": "tradeCcy",
          "position": 1
        },
        {
          "name": "tradeDate",
          "position": 2
        }
      ]
    },
    {
      "index": 2,
      "pageName": "tradeDetailsView",
      "excludeButtons": [
      ],
      "entryField": [
        {
          "name": "tradeCcy",
          "position": 1
        },
        {
          "name": "tradeDate",
          "position": 2
        }
      ]
    }
  ]
},
{
  "name": "SYSTEM_CONFIRMATION",
  "includeButtons": [
  "OK"
  ],
  "pages": [
    {
      "index": 1,
      "pageName": "tradeGeneralDetail",
      "excludeButtons": [],
      "entryField": [
        {
          "name": "tradeCcy",
          "position": 1
        },
        {
          "name": "tradeDate",
          "position": 2
        }
      ]
    }
  ]
}
```

```
{
  "name": "tradeDate",
  "position": 2
}
],
},
{
  "index": 2,
  "pageName": "tradeDetailsView",
  "excludeButtons": [],
  "entryField": [
    {
      "name": "tradeCcy",
      "position": 1
    },
    {
      "name": "tradeDate",
      "position": 2
    }
  ]
}
]
```

```
    ]  
}  
]
```

Code Significance

The significance of the above json content for Entry template is described below:

entryCriteria : The criteria to be used during entry generation. It has the following child attributes. In case of blank entry criteria, it should be marked as [].

component : The component to which it belongs.

baseEntryURL : The base entry URL to be used in controller.

baseAmendURL : The base amend URL to be used in controller. This value should be left blank if the update template is not required.

wizards: The wizard configuration to be used in entry UI. It has the following child attributes.

modes: The available wizard modes. The CRUD expects the modes to be arranged in EDIT - USER CONFIRMATION - SYSTEM CONFIRMATION. It has the following child attributes:

name : The mode name.

includeButtons : The buttons to be included in the respective mode.

pages : The configuration settings for each entry pages. It has the following child attributes:

index : The page index in the respective mode.

pageName : The page name for the entry UI.

excludeButtons : The buttons to be excluded in the page for the mode.

entryField : The fields to be used in entry jspx. It has the following configuration parameters:

type : The type of entry field. Possible values are Textbox, Date, Dropdown , Popup

name : The name of the attribute to be mapped. It should be in lower camel case. For example, TRADE_CCY should be written as tradeCcy.

mandatory : Whether the field is mandatory or not.

constraintName : The constraint name associated with the attribute. This field is optional.

position : The field position in the entry JSPX.

Generated views.xml

The CRUD tool creates generated-views.xml which contains the tiles-definition for various pages. The application should copy the content from generated-views.xml and place it inside the existing views.xml.

Sample file generated from above json : [generated-views.xml](#)

Generated Entry Command Form

Sample command form generated from above json: [TradeEntryCommandForm.java](#)

Generated Entry Controller

Sample controller generated from the above json: [TradeEntryController.java](#)

Generated Entry JSPX

Sample jspx generated from the above json: tradeGeneralEntry.jspx

3.4.1.5.2 READ (Query Template)

Introduction

The query criteria should be defined in the following way to generate the controller, command form, jspx ,and views.xml.

Code Snippet

The following code snippet is for creating the READ (Query Template).

```
CRUD_input.json

// other stuffs
"queryCriteria": [
  {
    "component": "TRD",
    "baseQueryURL": "/trd/query",
    "options": [
      {
        "queryField": [
          {
            "type": "Dropdown",
            "sortable": false,
            "optional": true,
            "name": "settlementCcy",
            "position": 1,
            "constraintName": "CURRENCY",
            "entityMapping": "settlementCcy->securityId"
          },
          {
            "type": "Textbox",
            "sortable": false,
            "optional": false,
            "name": "status",
            "position": 2,
            "entityMapping": "tradeCcy->securityId"
          }
        ]
      }
    ],
    //other stuffs=
```

Code Significance

The significance of the above json content for query template is described below:

queryCriteria : The criteria to be used during query formation. It has the child attributes. In case of blank query criteria, it should be marked as [].

component : The component name to which it belongs.

baseQueryURL : The base query URL to be used in controller

options : The attributes to be used in each query field. It has one child attribute **queryField**.

queryField : This defines the type of each query field to be used in jspx. It has the following child attributes:

type : Defines the type of input. Valid values are Textbox, Popup, Date and Dropdown.

sortable : Defines whether the value should be used as sort criteria or not. Valid values are true and false.

optional : Whether the item should be used in more criteria. Valid values are true and false.

name : Property name to be used.

entityMapping : Mapping of this property with entity attribute. For child entity, please use -> notation to relate the child to parent. For example, accountNo from inventoryAccount will be

inventoryAccount->accountNo. For entity mapping, with the same value as name attribute, name attribute value should be used. (Mandatory)

constraintName : Constraint value to be used. Applicable for dropdown only. For other inputs, this attribute is not necessary.

position : the position where this field will appear in the jspx.

Generated Query Command Form

Sample query command form generated from the above json : [TradeQueryCommandForm.java](#)

Generated Query Controller

Sample query controller generated from the above json: [TradeQueryController.json](#)

Generated Query JSPX

Sample query jspx generated from the above json : [trdQueryCriteria.jspx](#)

3.4.1.5.3 UPDATE (Amend Template)

Introduction

The baseAmendURL must be defined in the following way to generate the controller, command form, jspx ,and views.xml for amend operations. Since most of the attributes of entry and amend template is similar, so CRUD tool itself generates the amend components.

To run the tool :

```
D:\Test\sample-console\bin>launch FinchGenerateCRUD -f "D:\\final.json"

=====Generating entites=====
=====Entites generation completed=====

=====Generating Query Result View=====
=====Query Result View generation completed=====

=====Generating Repository=====
=====Repository generation completed=====

=====Generating Query Criteria Builder=====
=====Query Criteria Builder generation completed=====

=====Generating Service=====
=====Service generation completed=====

=====Generating Query Command Form=====
=====Query Command Form generation completed=====

=====Generating messages.properties=====
=====message.properties generation completed=====

=====Generating Query Controller=====
=====Query Controller generation completed=====

=====Generating Entry Command Form=====
=====Entry Command Form generation completed=====

=====Generating Entry Controller=====
=====Entry Controller generation completed=====

=====Generating Amend Command Form=====
=====Amend Command Form generation completed=====

=====Generating Amend Controller=====
=====Amend Controller generation completed=====

=====Generating views.xml=====
=====views.xml generation completed=====

=====Generating Query JSPX=====
=====Query JSPX generation completed=====

=====Generating Entry JSPX=====
=====Entry JSPX generation completed=====

=====Generating Project structure=====
```

3.4.1.6 Set-up Finch Sample Application

Introduction

This page describes how to set up, build and deploy sample application.

Prerequisite

Following software must be installed in the machine where the sample application will be built:

- Java SE 6 or above
- Apache Maven 3.x.x
- Apache Tomcat application server 6.x or above
- Eclipse Kepler
- GIT. For GIT installation see [GIT installation](#)

Set-up Sample Application

To set-up sample application fast we require the code base to download from GIT. To download code base execute the following steps

- Create a folder in local directory. For example D:\sampleApp
- Open command prompt and go to the location of the folder created in Step 1
- Enter the command -> git clone <https://github.com/nrifintech/finch.git>

After executing this command you will be asked to enter user name and password for GIT authentication. Enter user name and password and press ENTER.



```
D:\sampleApp>git clone https://github.com/nrifintech/finch.git
Cloning into 'finch'...
Username for 'https://github.com': mithut-nrift
Password for 'https://mithut-nrift@github.com': _
```

After executing above steps one folder named 'finch' will be created under sampleApp folder. 'finch' folder will contain the following folders

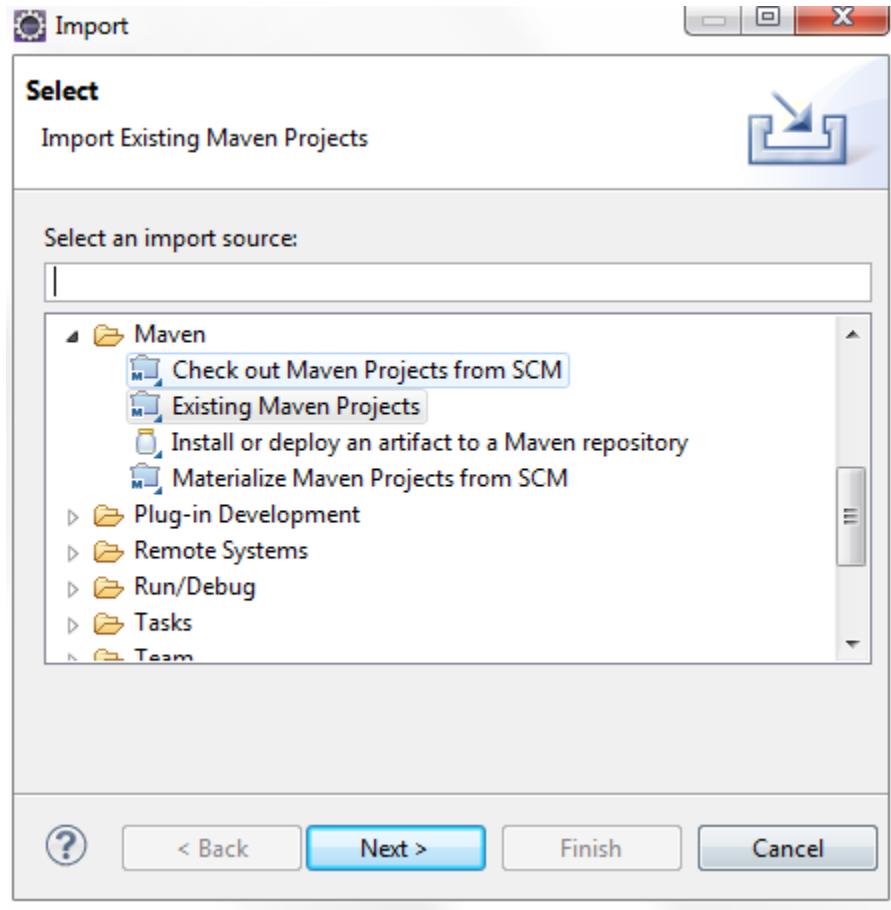
- devel : it contains all finch modules
- finch-blank :
- finch-minification :
- finch-uglification :
- sample-app : it contains all the sample application related modules. It contains another varia folder that contains sample application related database script
- varia : it contains all the finch related database scripts.

Import Sample Application in Eclipse

Assuming that Eclipse Kepler is installed in user machine do the following steps

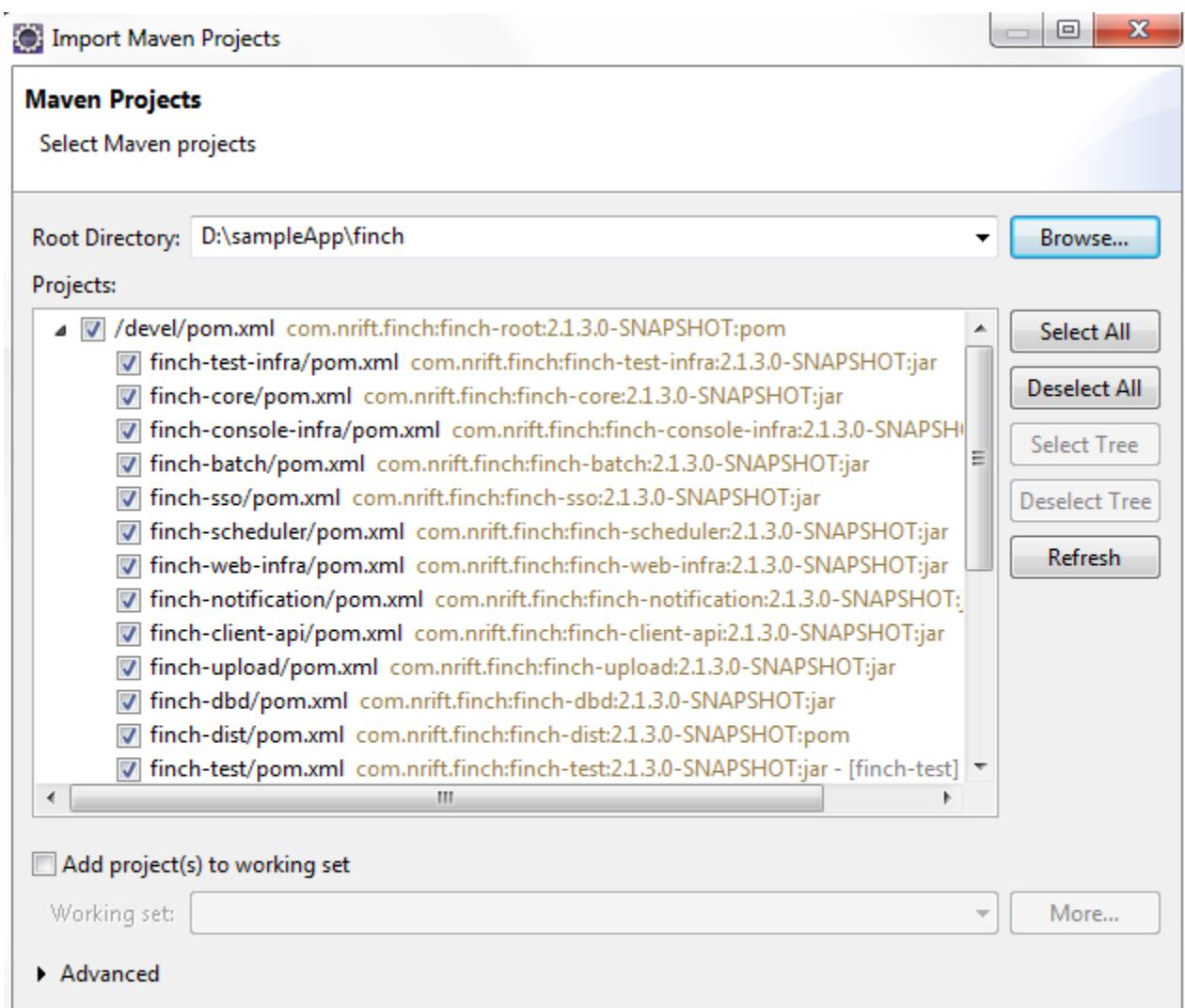
- 1) Open command prompt and go to <clone_dir>\finch\devel folder and run the command : mvn eclipse:eclipse
- 2) Go to <clone_dir>\finch\sample-app folder and run the command : mvn eclipse:eclipse

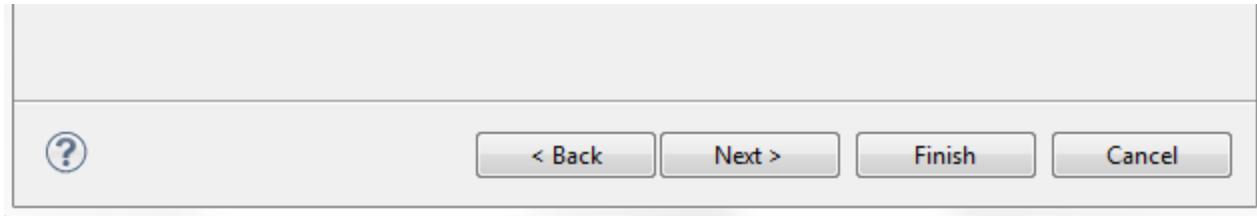
3) Now open Eclipse and go to File -> Import



4) Select Existing Maven Projects and Click Next button.

5) Click the Browse button and select the directory (e.g <clone_dir>\sampleApp\finch)





6) select all the items and click Finish button.

Database Set-up for Sample Application

finch provides db script for Oracle and MySQL in every release. This sample application runs using Oracle database.

To set up sample application to run with MySQL database following configuration needed to be performed

MySQL database set up

To run sample application in MySQL data base firstly MySQL server is to be installed. Assuming that MySQL server is installed in users machine do the following

- i) Add <MySQL_Installation_Location>\bin directory in PATH variable.
- ii) Open command prompt.
- iii) Type the command: mysql -u root -p and press Enter

```
D:\>mysql -u root -p
Enter password: _
```

Enter password(provided at the installation time) and press Enter

- iv) If the password is correct following prompt will be displayed

```
D:\>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 220
Server version: 5.6.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
mysql> _
```

v) Now you need to create a user for sample application schema. To create user enter following command one after another

```
CREATE USER '<userId>'@'localhost' IDENTIFIED BY '<password>';
```

For example: CREATE USER 'sample_app'@'localhost' IDENTIFIED BY 'sample_app_pwd';

```
GRANT ALL PRIVILEGES ON *.* TO '<userId>'@'localhost' WITH GRANT OPTION;
```

For example: GRANT ALL PRIVILEGES ON *.* TO 'sample_app'@'localhost' WITH GRANT OPTION;

```
CREATE USER '<userId>'@'<domain_userId>.user.nrifintech.com' IDENTIFIED BY '<password>';
```

For example: CREATE USER 'sample_app'@'mithut.user.nrifintech.com' IDENTIFIED BY 'sample_app_pwd';

```
GRANT ALL PRIVILEGES ON *.* TO '<userId>'@'<domain_userId>.user.nrifintech.com' WITH GRANT OPTION;
```

For example: GRANT ALL PRIVILEGES ON *.* TO 'sample_app'@'mithut.user.nrifintech.com' WITH GRANT OPTION;

Grant Privileges to other users

The above commands grants access to <userId> for example sample_app to the user mentioned in <domain_userId> .

If you want to make <userId> to be used by other user(s) then provide the following in place of <domain_userId>

Domain userId	If specific user id is given then the schema can be accessed from that user
%	All users under user.nrifintech.com domain can access the schema

For example:

If you want to grant 'prasunp' who belongs to user.nrifintech.com domain then execute the following commands

```
CREATE USER 'sample_app'@'prasunp.user.nrifintech.com' IDENTIFIED BY 'sample_app_pwd';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'sample_app'@'prasunp.user.nrifintech.com' WITH GRANT OPTION;
```

If you want to grant all users who belongs to user.nrifintech.com domain then execute the following commands

```
CREATE USER 'sample_app'@'%.user.nrifintech.com' IDENTIFIED BY 'sample_app_pwd';
```

```
GRANT ALL PRIVILEGES ON *.* TO 'sample_app'@'%.user.nrifintech.com' WITH GRANT OPTION;
```

vi) After user is created close the command prompt and again open command prompt and enter the command

```
mysql -u <userId> -p
```

For example: mysql -u sample_app -p

press enter

vii) Enter password <password> . For example: sample_app_pwd.

viii) Now you need to create schema for sample application. To create schema enter following command one after another:

1. create database <schema_name> charset utf8 collate utf8_bin;

For example: 1. create database finch_sample_dev charset utf8 collate utf8_bin;

2. use <schema_name>. For example: use finch_sample_dev

3. set collation_connection=utf8_bin;

4. set collation_database=utf8_bin;

5. set collation_server=utf8_bin;

6. show variables like '%col%';

ix) After executing the above commands sample application schema is ready. Now need to execute finch provided db scripts and sample app db script.

finch db script can be found [here](#) and sample app db scripts can be found in this location <clone_dir>¥finch¥sample-app¥varia

x) Download the finch db scripts and extract the zip file in your hard drive. In command prompt you need to execute the file using the following command

```
source <db_script_location_with_file_name>  
you need to provide the absolute path in <db_script_location_with_file_name>
```

```
D:\>mysql -u sample_app -p  
Enter password: *****  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 223  
Server version: 5.6.12 MySQL Community Server (GPL)  
  
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> use finch_sample_dev  
Database changed  
mysql> source C:\Users\mithut\Downloads\FINCH_CLEAN_2.1.2.0\COMMON\TARFILE\Consolidate_Script_Finch_Common_2.1.2.0_Tar.sql
```

fig : Executing the db script

Execute the script files in the following order. Assuming you are in the folder where you unzip the script files

- a) ¥FINCH_CLEAN_2.1.2.0¥COMMON¥TARFILE¥Consolidate_Script_Finch_Common_2.1.2.0_Tar.sql
- b) ¥FINCH_CLEAN_2.1.2.0¥CONSOLE¥TARFILE¥Consolidate_Script_Finch_Console_2.1.2.0_Tar.sql
- c) ¥FINCH_CLEAN_2.1.2.0¥WEB¥INF¥TARFILE¥Consolidate_Script_Finch_Web_INF_2.1.2.0_Tar.sql
- d) ¥FINCH_CLEAN_2.1.2.0¥WEB¥DBD¥TARFILE¥Consolidate_Script_Finch_Web_DB_2.1.2.0_Tar.sql

After executing all the above file you need to run sample app scripts as well

Execute the script files in the following order. Assuming you are in the folder <clone_dir>¥finc¥sample-app¥varia

- a) ¥MY-SQL¥sample-app-ref-create.sql
- b) ¥MY-SQL¥sample-app-trd-create.sql
- c) ¥MY-SQL¥common¥finc-inf-common-static-data.sql
- d) ¥MY-SQL¥console¥finc-inf-console-static-data.sql
- e) ¥MY-SQL¥web¥finc-inf-web-static-data.sql
- f) ¥MY-SQL¥web¥dbd-static-data.sql
- g) ¥MY-SQL¥sample-app-ref-static-data.sql

After executing all scripts you need to change the datasource.properties file located in <clone_dir>¥finc¥sample-app¥sample-dist¥src¥main¥resources¥META-INF¥config

```

# Setting for GLOBAL component
#
jdbc.components.url.GLOBAL = jdbc:<driver>://<host>:<port>/<schema_name>?characterEncoding=UTF-8
jdbc.components.userName.GLOBAL = <userId>
jdbc.components.password.GLOBAL = <password>
jdbc.components.minLimit.GLOBAL = <minLimit>
jdbc.components.maxLimit.GLOBAL = <maxLimit>
jdbc.components.initialLimit.GLOBAL = <initialLimit>
jdbc.components.queryTimeout.GLOBAL = <queryTimeout>

# Setting for SAMPLE component
jdbc.components.url.SAMPLE = jdbc:<driver>://<host>:<port>/<schema_name>?characterEncoding=UTF-8
jdbc.components.userName.SAMPLE = <userId>
jdbc.components.password.SAMPLE = <password>
jdbc.components.minLimit.SAMPLE = <minLimit>
jdbc.components.maxLimit.SAMPLE = <maxLimit>
jdbc.components.initialLimit.SAMPLE = <initialLimit>
jdbc.components.queryTimeout.SAMPLE = <queryTimeout>

# Datasource properties Oracle
jdbc.database.driverClass = com.mysql.jdbc.Driver
jpa.hibernate.dialectClass = org.hibernate.dialect.MySQL5Dialect

```

You need to replace datasource-context.xml file located at <clone_dir>¥finch¥sample-app¥sample-dist¥src¥main¥resources¥META-INF¥config . Get the file from [here](#) and replace the file.

Build Sample Application

Open command prompt move to <clone_dir>¥finch¥devel folder and execute the following command

```
mvn clean install -Dall
```

After successfully execution move to <clone_dir>¥finch¥sample-app folder and execute the following command

```
mvn clean package -Dall
```

After successfully execution the sample-web.war file and sample-console.zip file is stored in the following location

```
<clone_dir>¥sampleApp¥finch¥sample-app¥sample-dist¥target
```

Deploy Sample Application in Tomcat

MySQL Driver

Extract MySQL Database Driver JAR file (mysql-connector-java-5.1.27.jar) from MySQL-Database-Driver.zip file attached below, and place mysql-connector-java-5.1.27.jar in the container's 'lib' folder.

For example: If application is deployed in tomcat, then JAR file need to be place in <tomcat_home>¥webapps¥sample-web¥WEB-INF¥lib folder.

MySQL Connector JAR: [MySQL-Database-Driver.zip](#)

Tomcat Server Configuration for running finch using MySQL database

When sample application is depolyed in Tomcat 7 and it is used with MySQL database, an additional configuration needs to be done serve r.xml file of Tomcat

server.xml can be found <tomcat_home>¥config folder.

```
<Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener"  
driverManagerProtection="false"/>
```

After building the application sample-web.war file is created at location <clone_dir>\$finch\$sample-app\$sample-dist\$target

Copy the sample-web.war file and put it in <tomcat home>\$webapps folder.

Now start tomcat by double clicking the startup.bat file located in <tomcat home>\$bin folder.

Run Sample Application

After starting tomcat open any browser (e.g Internet Explorer) and type this url: <http://localhost:8080/sample-web>

8080 is the default port that Tomcat runs on. It has to be changed if user's tomcat runs on a different port

3.4.1.7 Set-up Sample Application

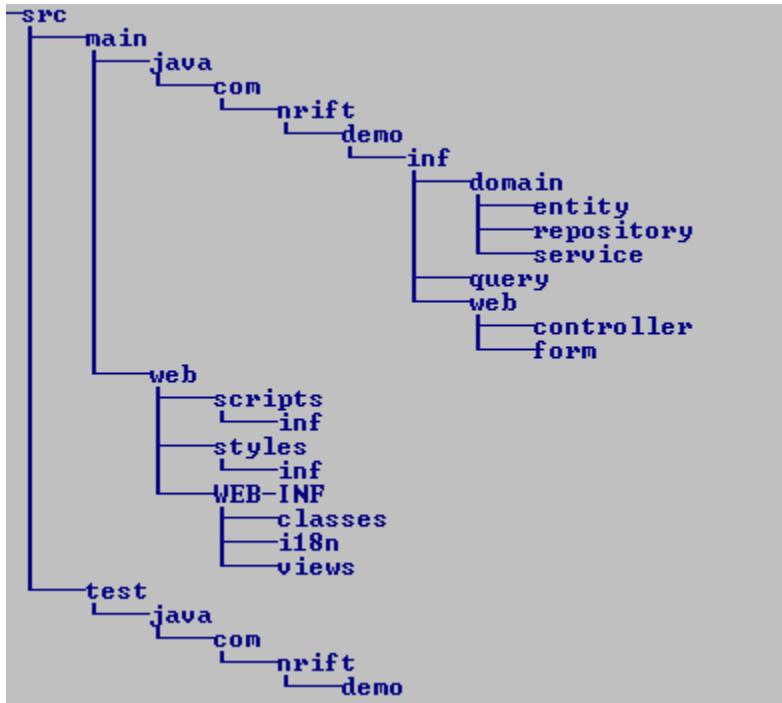
Introduction

This page describes how to setup and develop application using finch framework. The application will have a default Dashboard and a set of Screens provided by finch framework out of the box.

Steps to Build Application using finch

To build application using finch framework, follow the steps mentioned below:

1. First step is to setup a blank application. Please follow [this link](#) to setup a blank application.
2. Next step will be to create separate modules as per application needs. Steps to create separate module is mentioned below:
In our example we are creating an app called demo and every module in it will follow the following structure
Here inf is just a place holder, it will depends on application need.



To generate module with above directory structure, following steps need to be followed

- Open command prompt and navigate to finch-app (created using steps describe in Steps to Build Blank Application Using finch link) directory.
- Execute following command for each module.

```
mvn archetype:generate -DgroupId=[Java root package for Application] -DartifactId=[Application module name]
-Dversion=[Application Module version] -DarchetypeArtifactId=[Archetype Artifact Id] -DarchetypeGroupId=[Archetype Group Id]
```

Example:

```
mvn archetype:generate -DgroupId=com.nrift.demo -DartifactId=demo-trd -Dversion=1.0-SNAPSHOT -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart
```

ArchetypeArtifactId depends on application needs, therefor it may not be the same as specified here.

If pom.xml file in finch-app do not contain any entry for demo-trd (as in our example, it will be your module name in your case) module, then please add this module in the pom.xml file under <modules> tag.

```

Example:
<project>

//other stuffs
<modules>

<module>finch-app-dist</module>

<module>demo-trd</module>
</modules>

</project>

```

- c. Maven will also confirm groupId, artifactId information supplied to it, when it will ask for confirmation press 'Y'.
- d. If everything goes well, one directory will be created namely demo-trd.
- e. Now create static resources like .js, .css, image etc. files under directory of your choice. For example all the .js files for demo-trd, resides under src/main/web/scripts/inf directory and all the .css files resides under src/main/web/style/inf directory. For .js files create folders namely scripts and scripts/inf and for style sheet create folder styles/inf under src/main/web folder.
- f. Use [this link](#) (use finchro as user name and password) to download finch-console.zip file, then unzip this file and go to 'conf/META-INF/config' folder and change datasouce.properties file and also copy MySQL jar file into lib folder. Go to [this link](#) to generate Controller, Repository, Entity, JSPX etc files using CRUD tools, and copy files into respected directories.

During CRUD tools execution, user may face "too many connection" MySQL exception. If this exception is encountered then please open MySQL from command prompt and execute following command
set global max_connections=5000;

- g. Now configure pom.xml file of demo-trd directory

Add following dependencies

```

<dependency>
  <groupId>org.apache.activemq</groupId>
  <artifactId>activemq-camel</artifactId>
  <version>5.6.0</version>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.apache.activemq</groupId>
  <artifactId>activemq-pool</artifactId>
  <version>5.6.0</version>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.apache.activemq</groupId>
  <artifactId>activemq-core</artifactId>
  <version>5.6.0</version>
  <scope>runtime</scope>
</dependency>

```

Also include respective database driver dependency in the above file.
For example:

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.27</version>
<scope>runtime</scope>
</dependency>
```

- h. Now configure assembly-web.xml file which resides under finch-app-dist/src/main/assembly folder.

For static resources, add the following filesets tag in assembly-web.xml file

```
fileset tags
<fileSet>
<directory>..../demo-trd/src/main/web/scripts</directory>
<outputDirectory>scripts/</outputDirectory>
</fileSet>
<fileSet>
<directory>..../demo-trd/src/main/web/WEB-INF/i18n</directory>
<outputDirectory>/WEB-INF/classes/i18n</outputDirectory>
</fileSet>
<fileSet>
<directory>..../demo-trd/src/main/styles</directory>
<outputDirectory>styles/</outputDirectory>
</fileSet>
<fileSet>
<directory>..../demo-trd/src/main/web/WEB-INF/views</directory>
<outputDirectory>/WEB-INF/views</outputDirectory>
<excludes>
<exclude>scripts/**</exclude>
</excludes>
</fileSet>
<fileSet>
<directory>src/main/web/</directory>
<outputDirectory>/</outputDirectory>
</fileSet>
```

Also add the following dependencies for class files

Please add respective jar file for database driver in <include> tag. For example: <include>mysql:mysql-connector-java</include>

Dependency tag in assembly-web.xml file

```
<dependencySets>
  <dependencySet>
    <useProjectArtifact>false</useProjectArtifact>
    <scope>runtime</scope>
    <outputDirectory>/WEB-INF/lib/</outputDirectory>
    <includes>
      <include>com.nrift.demo:demo-trd</include>
      <include>org.apache.activemq:activemq-camel</include>
      <include>org.apache.activemq:activemq-pool</include>
      <include>org.apache.activemq:activemq-core</include>
      <include>org.apache.geronimo.specs:geronimo-j2ee-management_1.1_spec</include>
    </includes>
  </dependencySet>
</dependencySets>
```

- i. Add following dependency in pom.xml of finch-app-dist folder

Dependency tag in pom.xml of dist folder

```
<dependencies>
  <dependency>
    <groupId>com.nrift.demo</groupId>
    <artifactId>demo-trd</artifactId>
    <version>${project.version}</version>
  </dependency>
</dependencies>
```

- j. Change 'datasource.properties' and 'datasource-context.xml' file under 'finch-app-dist/src/main/resources/META-INF/config' directory.
Example: '[datasource.properties](#)' , '[datasource-context.xml](#)'
- k. Change package type of the module. Open pom.xml file of demo-trd , change package type to jar [<packaging>jar</packaging>].
- l. To change name of final war file, open pom.xml file of finch-app-dist folder and look for finalName tag under profiles tag and change it suitable one. Same name should be given in finch-app/pom.xml file under <name> tag.
- m. Now execute mvn package command to generate final distribution. After successful execution of the command a war file will be generated in finch-app-dist/target folder.
- n. Deploy the war in the tomcat application server. To deploy the war copy the war file under '[TOMCAT-HOME]/webapps' folder.

Add a new module

1. Introduction

1.1. Summery

finch framework built with multiple modules like finch-core, finch-batch, finch-infra, finch-dbd etc. Application built on top of finch can create multiple module as per their needs. Each module in any application serves a specific purpose. If for a requirement there does not exists any appropriate module then application need to create a new module.

1.2. Scope

This document describe how to create a new module - what the nomenclature of new module will be, folder structure, how configuration files are kept, etc.

2. Pre-requisites

To create a new module, user must have knowledge of Maven, Java and finch. This document assume that application has already being created using finch framework and application now wants to add a new module.

3. Sequence and Steps

3.1. Naming convention of modules

Naming of module is important since it helps us to understand the purpose of the module and at the same time identify it quickly. Confusion should not arise which feature this module supports by just looking at the name. Module name should be short possibly 3 or 4 alphabets and should not be alpha numeric.

3.2. Directory Structure

Directory structure of each module should follow similar pattern. It is up to application to define directory structure.

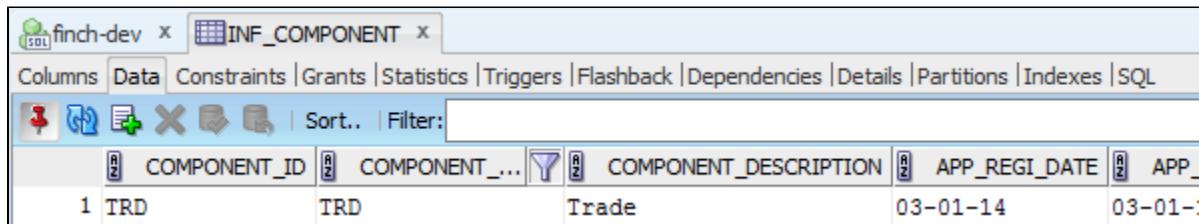
3.3. Steps

Please see [this link](#) to create a module. Please skip step 1 (create blank-app using finch), as this document expects that, user is adding new module into existing project.

3.4. Table

3.4.1. INF_COMPONENT

This table lists all components. A new records needs to be added here. Example for TRD module has been shown below.



COMPONENT_ID	COMPONENT_NAME	COMPONENT_DESCRIPTION	APP_REGI_DATE	APP_MODI_DATE
1	TRD	Trade	03-01-14	03-01-14

4. Document History

Version	Date	Description of changes	Author
1.0	12-02-2013	Initial Document	Prasun Pal

3.4.2 Setup Development Environment

3.4.2.1 Environment Setup

Required Software

- jdk-7.x
- apache-maven-3.X.X
- [Eclipse Kepler Release](#)
- [GIT](#)
- OracleXEClient
- sql-developer-3.X.X
- MySQL

Steps to Setup Finch Development Environment

Step 1:Install jdk

Installation of Jdk is simple. Get the Oracle JDK Installer and follow the instruction.

Step 2: GIT Setup

Detail procedure is mentioned in page written for GIT installation page.

Step3: Set up for Database

Install OracleXEClient and SQL Developer

- Installation of Oracle is normal just run the installer available and follow the instruction.
- To access DB we should install Sql Developer. Installation of Sql Developer in normal just follow the instruction
- After installation of SQL Developer configure the DB connections. Database Connection details can be found at Environment Details .

Step 4:Install Apache-Maven-3.0.3

- Get the apache-maven-3.0.x from software repository.
- It is a zipped file. Extract the zipped file at any of your preferred location.
- Setup the Environment Variables mentioned on below Note
- Verify Maven installation - Open Command Console and run 'mvn --version'
- Goto maven directory under User home directory. For example C:\Users\<Username>\.m2
- Open settings.xml and put your Username and Password under <proxy> tag as below.
- Open command prompt and go to the directory where your code downloaded
- Run the below command to build

For finch

```
mvn eclipse:clean  
mvn eclipse:eclipse -Ddownloadsources  
mvn eclipse:clean
```

For Sample-App

```
mvn eclipse:clean  
mvn eclipse:eclipse -Ddownloadsources  
mvn eclipse:clean
```

- Run eclipse.exe to run the eclipse
- Create the appropriate workspace
- Goto File -> Import -> Maven -> Existing Maven Projects
- in the import project window goto the project location and select the project folder and follow the instruction to import the project into your workspace
- you will get some warnings/errors for you build path set.
- To resolve the warnings/errors goto Java -> Build path -> Classpath Variables window.
- Press new button will get a popup window "New Variable Entry".
- **set Name: M2_REPO and Path: C:/Users/<username>/m2/repository** and press ok.

Set Environment Variables

Set following Environment Variables as per your installation and configuration. Set JAVA_HOME as \$(JDK Installation Dir)

1. Prepend %JAVA_HOME%\bin to PATH

2. Set M2_HOME as \$(Maven Installation Dir)
3. Set M2_REPO as \${user.home}/.m2/repository

3.4.2.2 Environment Details

GIT Details

URL - <https://github.com/nrifintech/finch.git>

GIT server - <https://github.com/>

For GIT access, please contact [Gautam Roy](#).

Issue tracking

JIRA URL : <http://jira.nrifintech.com/project/browse/FCHII>. For creating issue refer to the issue naming convention followed in finch: [2.1.5 Subject Tracking](#)

Database Information

Name	Description
Oracle SID	d11gr21
Server Name	
Server IP	172.16.29.92
Oracle Version	d11gr21

Schema Information

Name	Purpose	Creation Date	Update Date	Version
finch_dev	Development Schema	06 Sep 2012		
finch_unit	Unit Test	06 Sep 2012		

Nightly Build

We have set up a nightly build machine. It runs finch build process at mid-night everyday and deploy newly built sample application in the tomcat server.

URL to access the sample application is NA.

Name	Port	Purpose	Update Date	Version
172.16.29.85	8282	Jenkins Nightly Build Schedule:		

<http://finch-release:8282/jenkins/>

3.4.2.3 GIT Setup

GIT-CLIENT : Installation guide for windows

GIT is an efficient open source distributed version control system. It is compatible with operating systems like windows,linux,mac etc.

This document is a guideline for installing GIT(1.8.3) installer in windows based operating systems.

STEP-1:

[GIT-installer.zip](#)

Get the GIT installer file from the archive dump otherwise click the above link to download it and copy it to your local file system.

STEP-2:

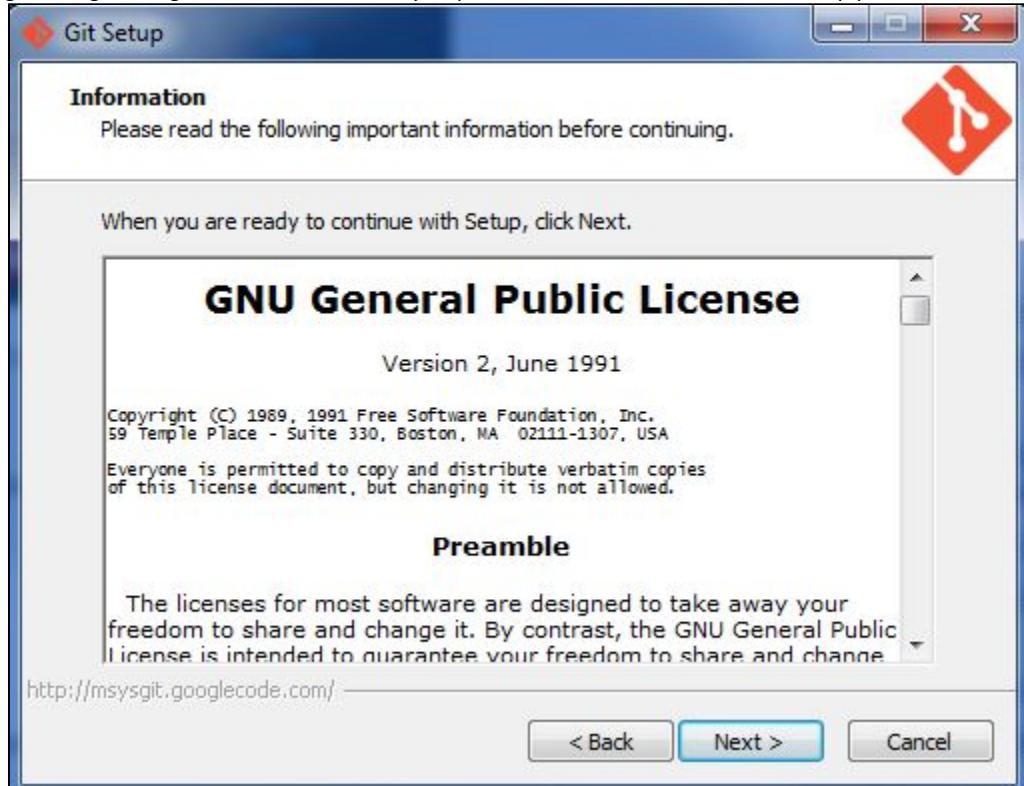
On starting the installer wizard the first screen that will come is the welcome screen.



Click the next button to go to the next screen.

Step-3:

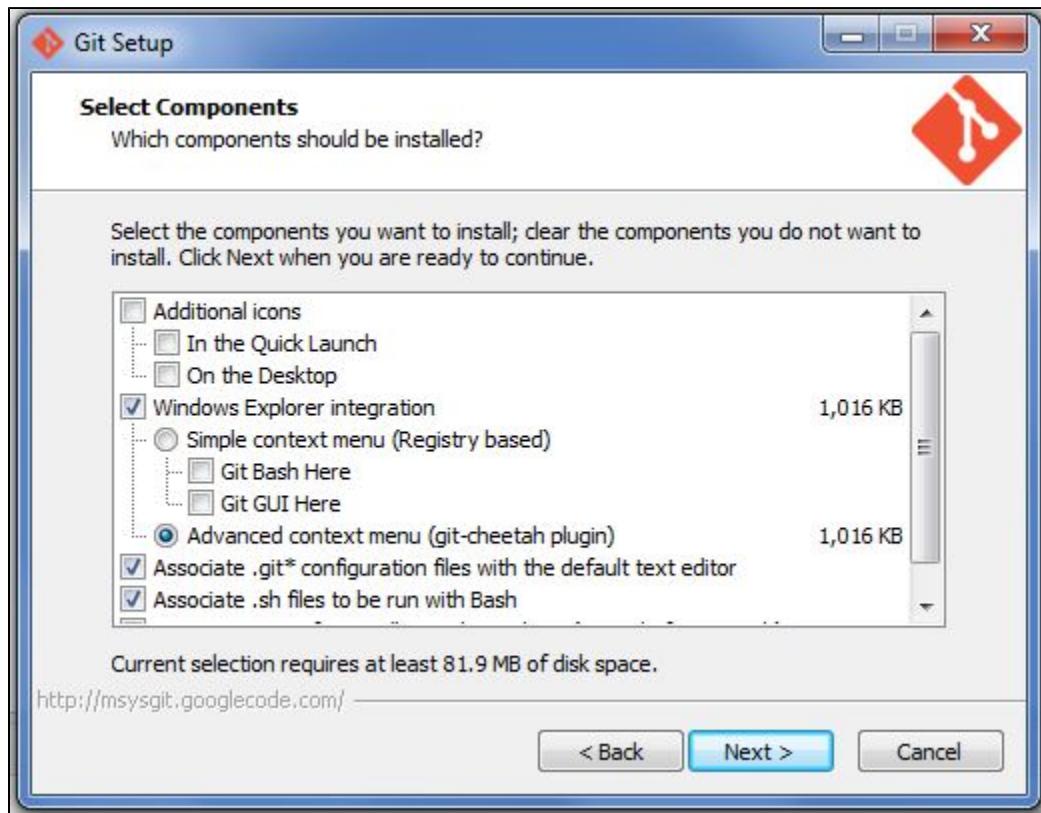
The next screen is the license agreement screen. It will let you know that it is freeware and if you are keen to know the license status then go through the agreement doc otherwise just press "Next" to continue with the set-up process.



Note: By default this will be installed under "C:\Program Files\Git".

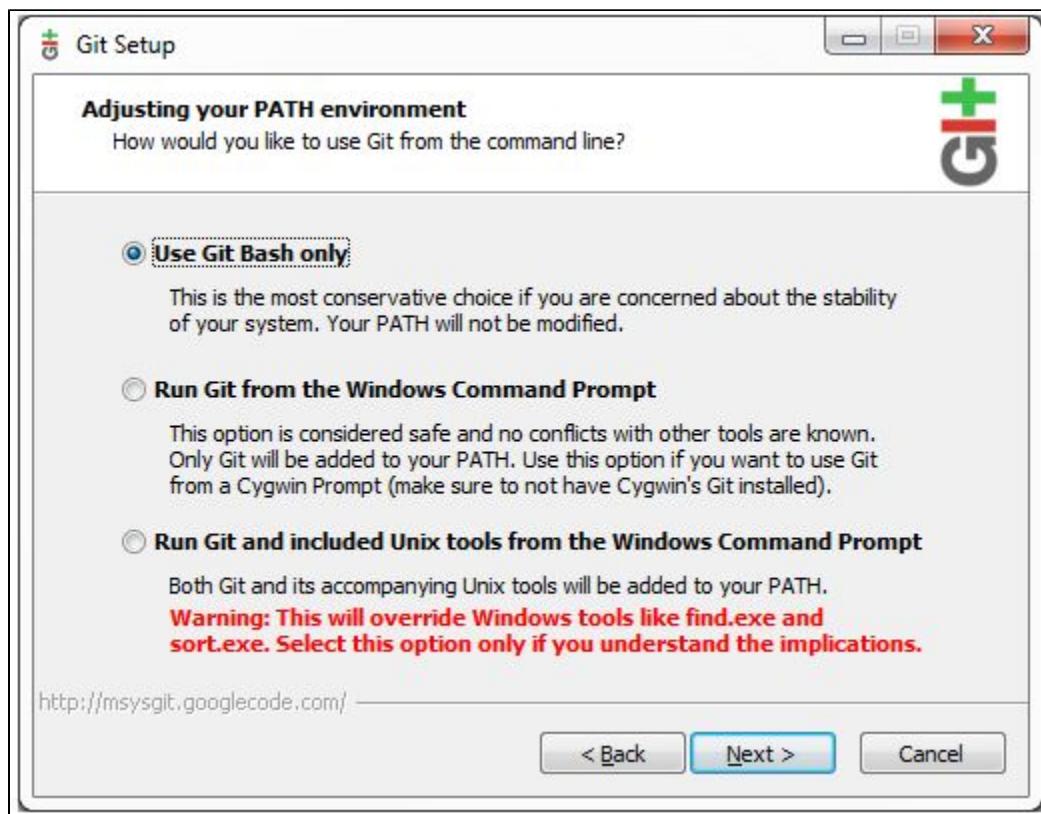
Step-4:

The very next screen is the component screen. It asks the user to select which components of GIT he/she wants to install. Use the default selections and click on "Next".



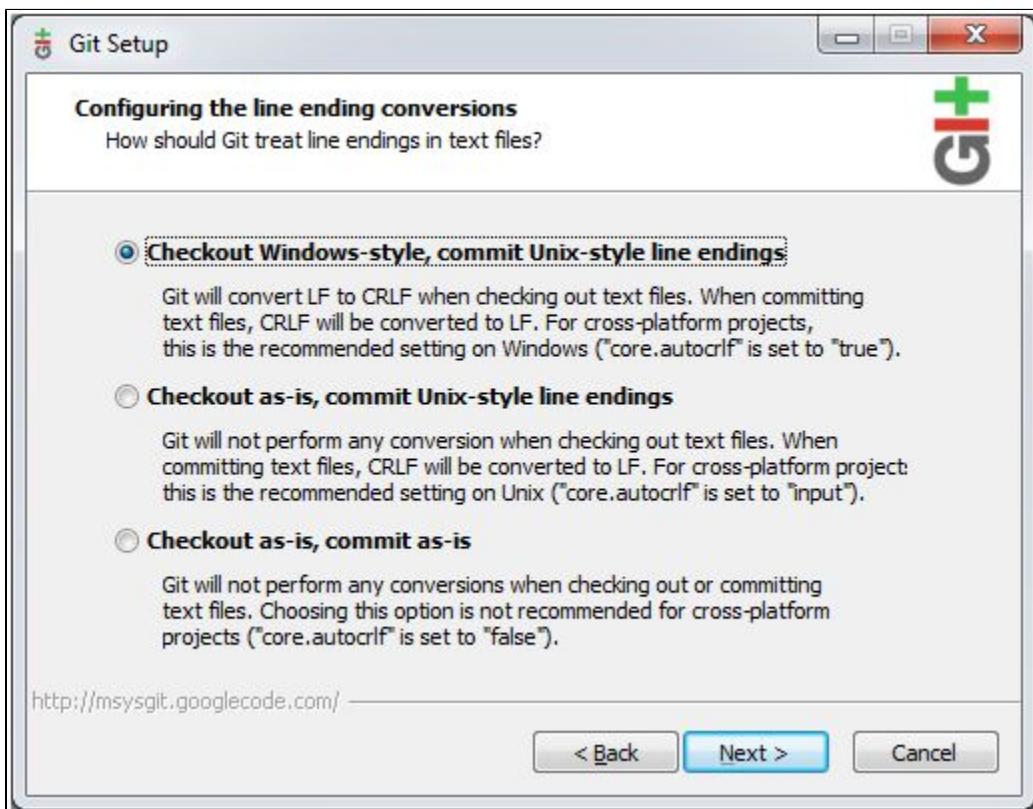
Step-5:

The screen3 is for adjusting your path environment. Use the default selection “Use Git Bash Only”.



Step-6:

This screen is for configuring the line-ending conversions. This indicates how Git will treat the line ending in the text files. If you are planning to checkout files from a remote Git repository that is running on Linux, use the default selection “Checkout Windows-style, commit Unix-style line endings”.



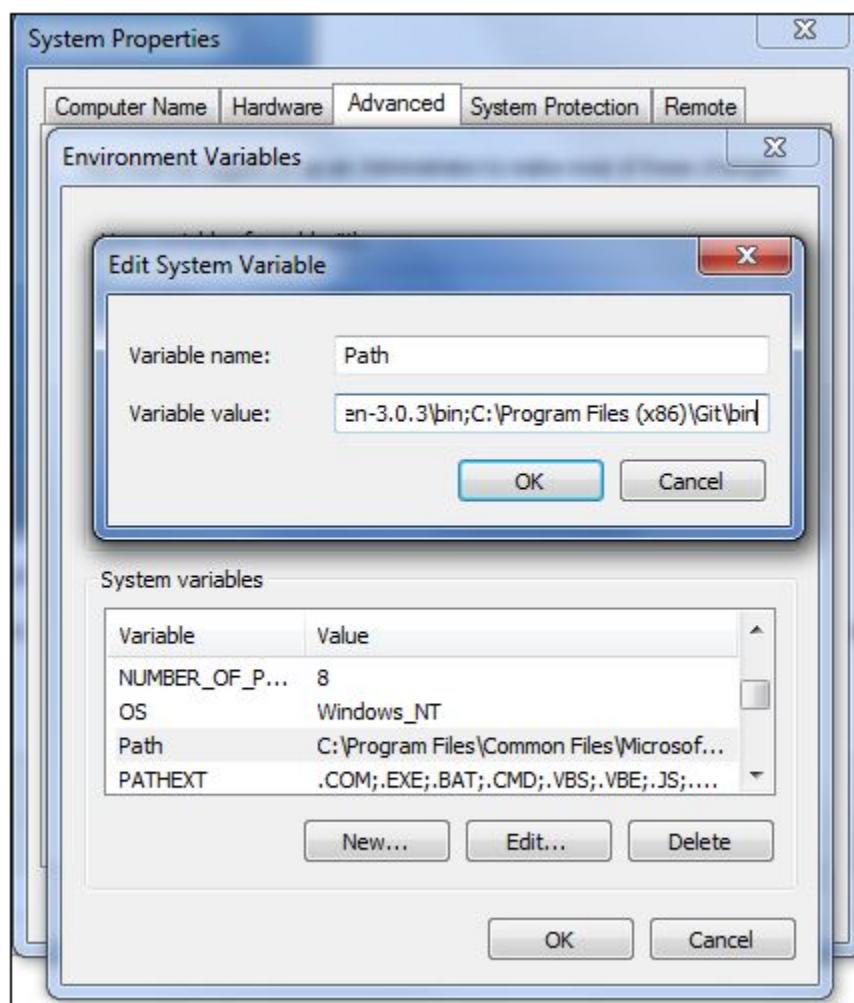
Step-7:

After clicking the "Next" button from the above configuration screen the installation will start and a progress bar will indicate the installation progress.

Step-8:

Click on finish to exist the set-up.

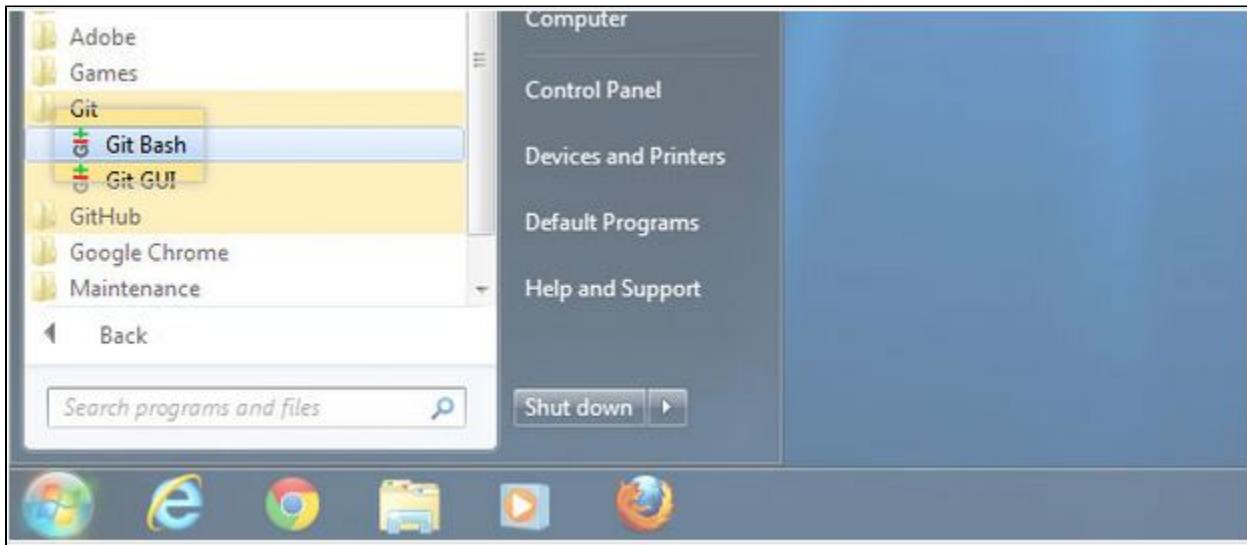
After the installation is complete you need put an entry for the GIT for environment variable key "Path". Go to "Control Panel\System and Security\System" and click on the "Advanced system setting" link from the left side panel. Now click on "Environment Variables" button to see all the system variables and click on the variable "Path" and append the bin path of GIT directory, i.e,"C:\Program Files (x86)\Git\bin" and don't forget to put the delimiter ";" before it , with the variable value.



Now run the following commands from the command console to configure your settings.

```
git config --global user.name "your username"  
git config --global user.email "your email"  
git config --global http.proxy http://username:password@proxy.nrifintech.com:3128
```

The same operation can be done from the GIT Bash. Open the Git Bash (not command console).



Enter the same lines of commands here, just prep-end a "\$" at the beginning of each command line.

GIT can be used from command line as well as Eclipse GIT plugin e-git.

Configuration

Proxy settings:

If git is already configured and running, then proxy needs to be configured:

```
git config --global http.proxy http://<userid>:<password>@proxy.nrifintech.com:3128
```

where

userid : user id created for git account

password: created during git account

Editor Setup:

```
git config --global core.editor "<Editor with full path>' -multiInst -notabbar -noSession -noPlugin"
```

e.g: to open files with notepad++,

```
git config --global core.editor "C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -noSession -noPlugin"
```

3.4.2.3.1 Command Line Interface (CLI)

First time only

```
# for the first time, clone the repo and
git clone <URL>
# check whether you have 'develop' branch or not
git branch -a -vv
# if you don't have 'develop' branch then create local branch 'develop' that tracks 'origin/develop', this requires
only once, otherwise following is NOT required
git checkout -b develop origin/develop
```

Daily commands for Sprint

```
# first of all make sure that you are in 'develop' branch.  
git status  
  
# if you are not in 'develop' branch then checkout to 'develop' branch.  
git checkout develop  
# if you don't have issue specific local branch then create it from 'develop' branch, otherwise skip the following command.  
# say you are creating a local branch for issue FCH-XXX from 'develop' branch.  
git checkout -b FCH-XXX develop  
  
# if you already have your issue specific local branch say 'FCH-XXX', do the following.  
# edit files, add and commit in 'FCH-XXX' as usual, make sure that you are in 'FCH-XXX'  
git add --all  
git commit -m "message for FCH-XXX"  
  
# work in your local branch till the issue is resolved, commit frequently as per you convenient  
# and when you feel that the issue is completely resolved then only it's time to push to remote 'develop' branch.  
# checkout to local 'develop' branch and update it.  
git checkout develop  
git pull  
  
# go to FCH-XXX again and rebase on the updated local 'develop' branch  
git checkout FCH-XXX  
# you need to consolidate the local commits comments and squash it using -i option  
git rebase develop -i  
  
# to consolidate to one comment take one 'pick' and make other 'squash' then in second VIM give your only one comment  
  
# if CONFLICT then manually merge that file only and add that file only (don't use add all)  
git add <specific file name>  
  
# make sure that no files remained conflicted  
git status  
# continue rebase  
git rebase --continue  
  
# go to local 'develop', merge and push it  
git checkout develop  
git merge FCH-XXX  
  
# in local 'develop' push to remote 'develop'  
git push origin develop  
  
# delete local 'FCH-XXX'  
git branch -d FCH-XXX
```

For every unit of work create a branch and merge that branch into 'develop' branch. Don't directly work on 'develop' branch. For every single task/issue, there should be only one push to the remote 'develop' branch.

```
# push new local branch to remote repo and track it  
git checkout FCH-XXX  
git push -u origin FCH-XXX  
  
# push to existing remote branch  
git push origin FCH-XXX
```

Daily Useful Commands

useful commands

```
# override all local changes with the updated remote origin  
git fetch --all  
git reset --hard origin/develop  
  
# delete remote branch  
git push origin --delete FCH-XXX
```

```
# file delete  
# to delete a file in local branch and remote, first provide '--all' option in add  
git add --all  
# then do normal commit  
git commit -m "deteted files"
```

```
# a better way is to use git clean  
# will do a dry-run to remove untracked files, including directories (-d) and files ignored by git (-x). Replace the -n  
argumanet with -f to perform the actual delete  
git clean -d -x -n
```

QA release

```
# make sure that you are in 'develop' branch.
git status

# if you are not in 'develop' branch then switch to it.
git checkout develop
# first freeze GIT remote push for all developers; make sure all developers already pushed their changes to
git pull origin develop

# checkout to qa release branch. The <version> will be same as public release version like 4.2.5.0.
git checkout -b <version> develop

# now you are in qa release branch. This branch is a place to clean up the release, test everything, update
# and do any other kind of preparation for the upcoming release.
# If any collaboration and code-review is required then this branch can be used.
# This branch will be used for qa testing.
# change pom.xmls to remove snapshot,
# change README.MD
# change datasource.properties of sample-app to point finch_qa
git add --all
# commit those changes
git commit -m "qa release <version>"

# push this branch to remote for qa testing
git push -u origin <version>
# for QA release we do not tag in GIT

# Now we will continue our development for next Sprint in 'develop' branch.
# go to 'develop' branch, change POMs version to next development version with SNAPSHOT
# change README.MD
git checkout develop

# change POMs and README.MD
git commit -am "development for <next version>"
git push
```

Public Release

```

# make sure that you are in QA branch like 4.2.5.0.
git status

# if you are not in QA branch then switch to it.
git checkout <version>
# first freeze GIT remote push for all developers
# make sure all developers already pushed their changes
# QA team already tested all the issues to be delivered in this release
# then pull it
git pull origin <version>

# DON'T change POMs at this stage
# if you already have local 'master' branch then checkout to it.
git checkout master
# pull the latest for 'master'. You should NOT get any conflict.
git pull
# merge release <version>.
git merge <version>

# at this stage you can get POM conflicts.
# check the conflicted files
git status

resolve it manually and add that POM file only (don't use add all)
git add <specific POM file name>
# change pom.xmls to remove snapshot,
# change README.MD
# change datasource.properties of sample-app pointing to finch_release
git add --all
# commit those changes
git commit -m "public release <version>"

# Build locally and ask all the developers to test the application before push
# Release Manager should build it locally and share his URL(http://<username>:port/sample-web) to all the developers for
pre release testing
# push to 'master'
git push origin master
# tag it
git tag -a <version> -m "public release <version>"
# psuh all the tags
git push --tags

# got to github and update the latest release

```

Patch Release

```

# make sure that you are in 'master' branch.
git status

# if you don't have local 'master' branch then make it with tracking remote 'master'; otherwise skip this
git checkout -b master origin/master

# if you have local 'master' already then checkout to it.
git checkout master

# first freeze GIT remote push for all developers; make sure all developers already pushed their changes for the patch to
# 'master' branch then pull it
git pull

# make local hot fix branch on top of local 'master' branch and checkout to it.
# this branch acts as temporary path release branch 4.2.2.1
git checkout -b <patch> master

# change any addition file if required and commit to branch <patch> otherwise SKIP this.
# DON'T change POM versions and README.MD in this stage.
git add --all
git commit -m <message>

# go to 'master' and merge it with <patch>. If you skipped the above step then it's probably up-to date.
git checkout master
git merge <patch>

# change README.md and all the POM project version number then commit
git commit -am <message>

# give single commit message like "public release 4.2.2.1"
# compile with all test cases, deploy and do smoke testing
# push to origin master
git push origin master

# tag it.
git checkout master
git tag -a <version> -m "public release <version>" 
git push --tags

# got to github and update the latest release

```

3.4.2.3.2 E-GIT

Assumption

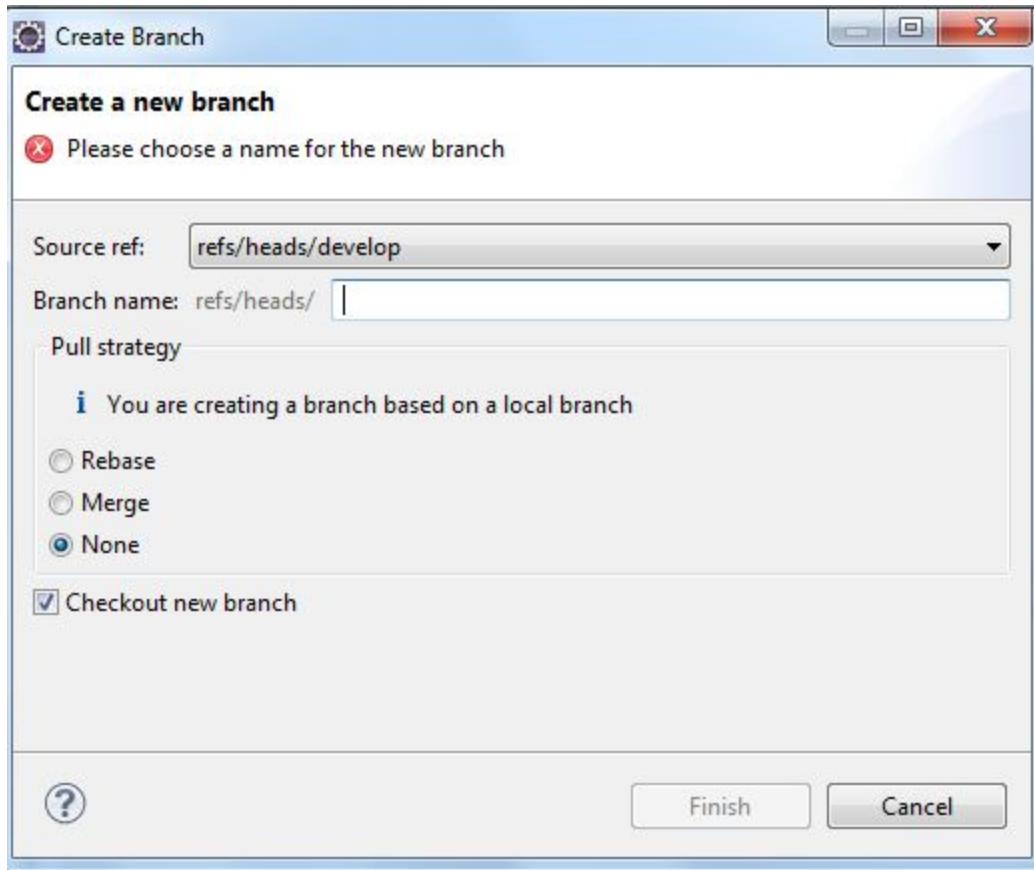
This page assume all software needed to finch development are installed like:

- All Software including Eclipse Java EE IDE for Web Developers, Version: Kepler Release with git and EGIT and mentioned in [Developer environment Setup page](#)
- And Developer is going to work with a particular issue say FCH-9999

Steps:

Step1: Fetch the develop version of from git

Step2: Create local branch for assigned issue



Step3: Do the changes and verification.

Step4: Commit changed files for the issue in the local branch with "Message".

Step5: Switch to local develop branch

Step5.a: Do the pull

Step6: Switch to local issue branch.

Step6.a: Rebase to local develop

Step7: Switch to local develop

Step7.a. Merge with Squash

Step7.b. Commit changed files for the issue in the develop with "Message".

Step7.c. push to upstream

Step7: Remove Issue Branch - This steps is optional

3.4.2.4 JRebel Configuration

JRebel Installation for Eclipse 3.3+

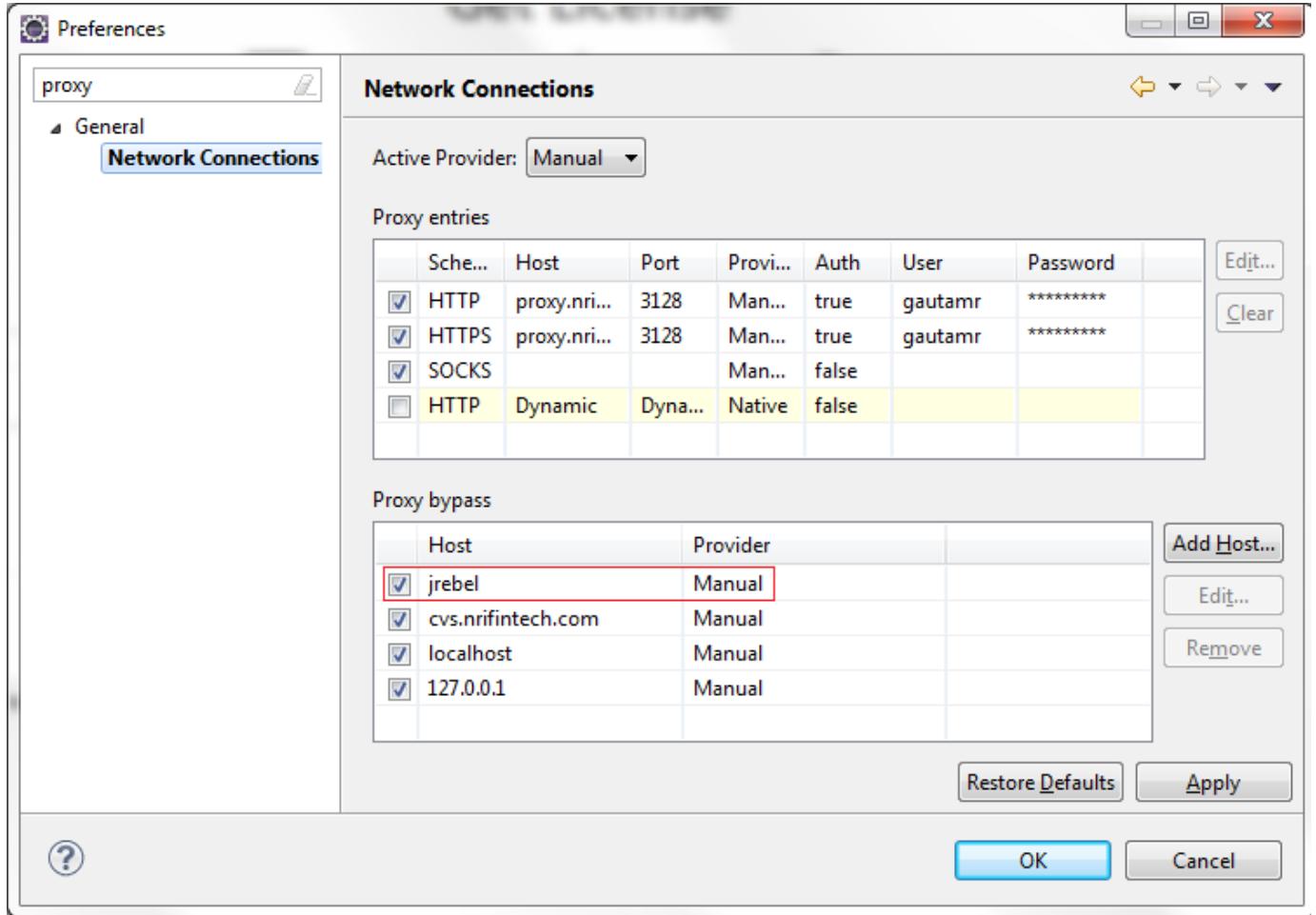
To use JRebel, developers need (a) Eclipse JRebel plugin (5.3.1.RELEASE) and (b) JRebel license.

Step 1

If any previous version of JRebel plugin (< 5.3.1.RELEASE) is installed in Eclipse then uninstall it.

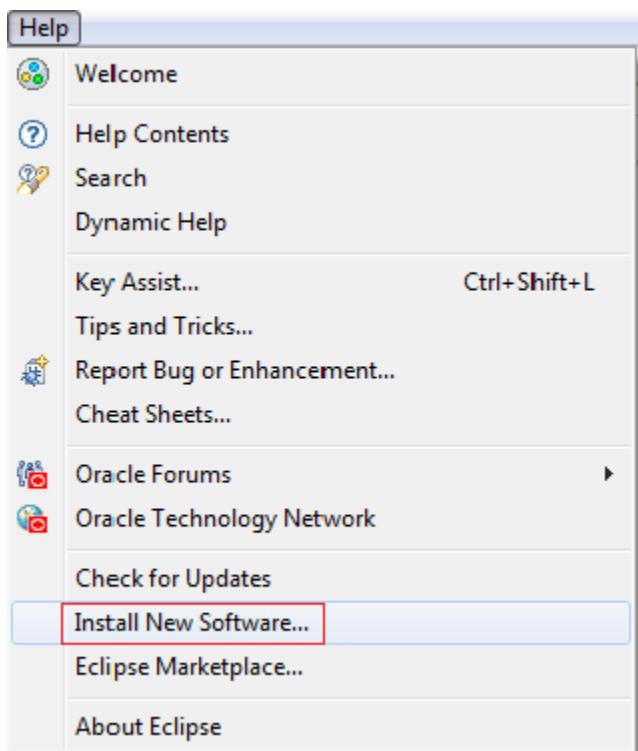
Step 2

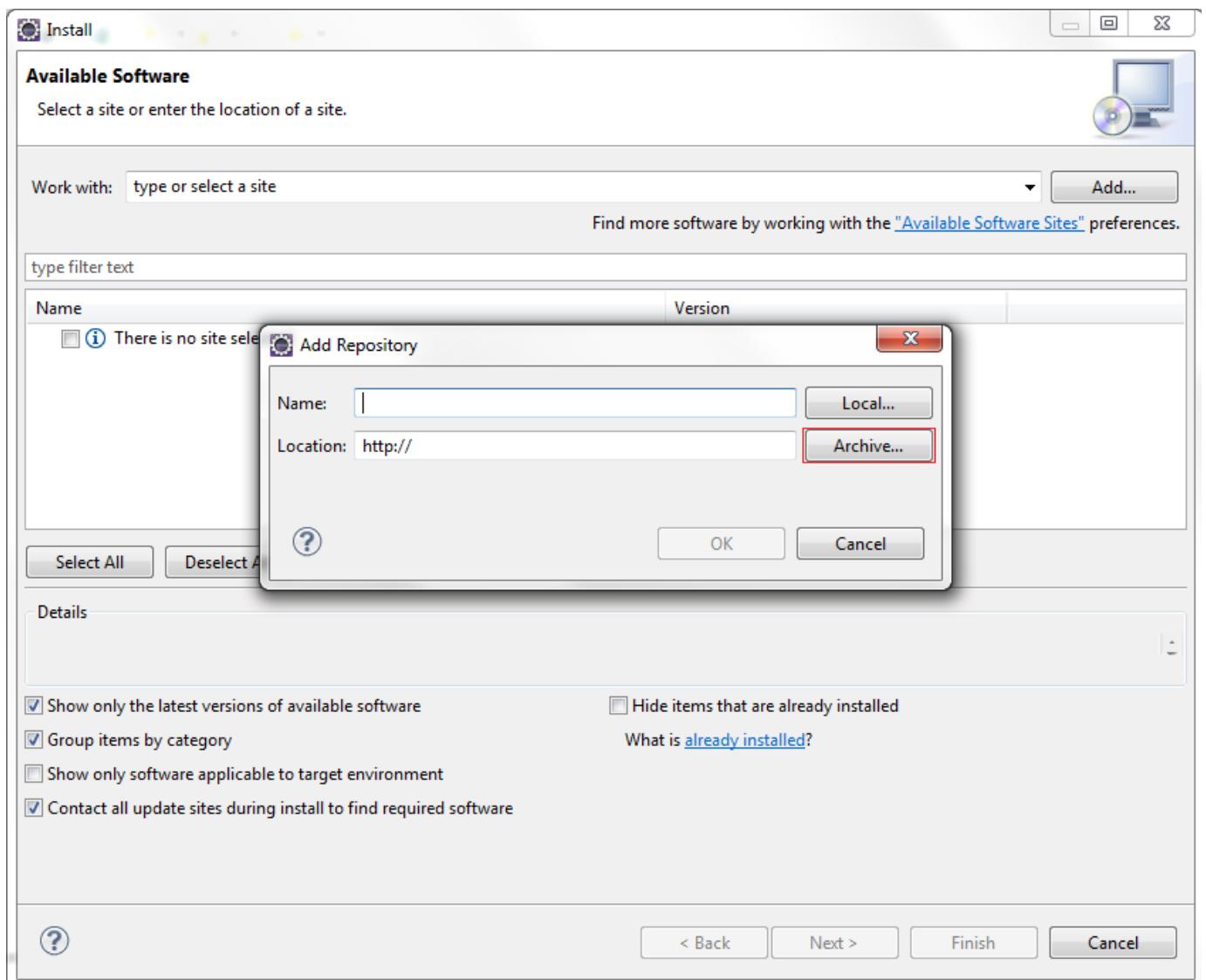
Add jrebel URL to "Proxy bypass" in Eclipse.



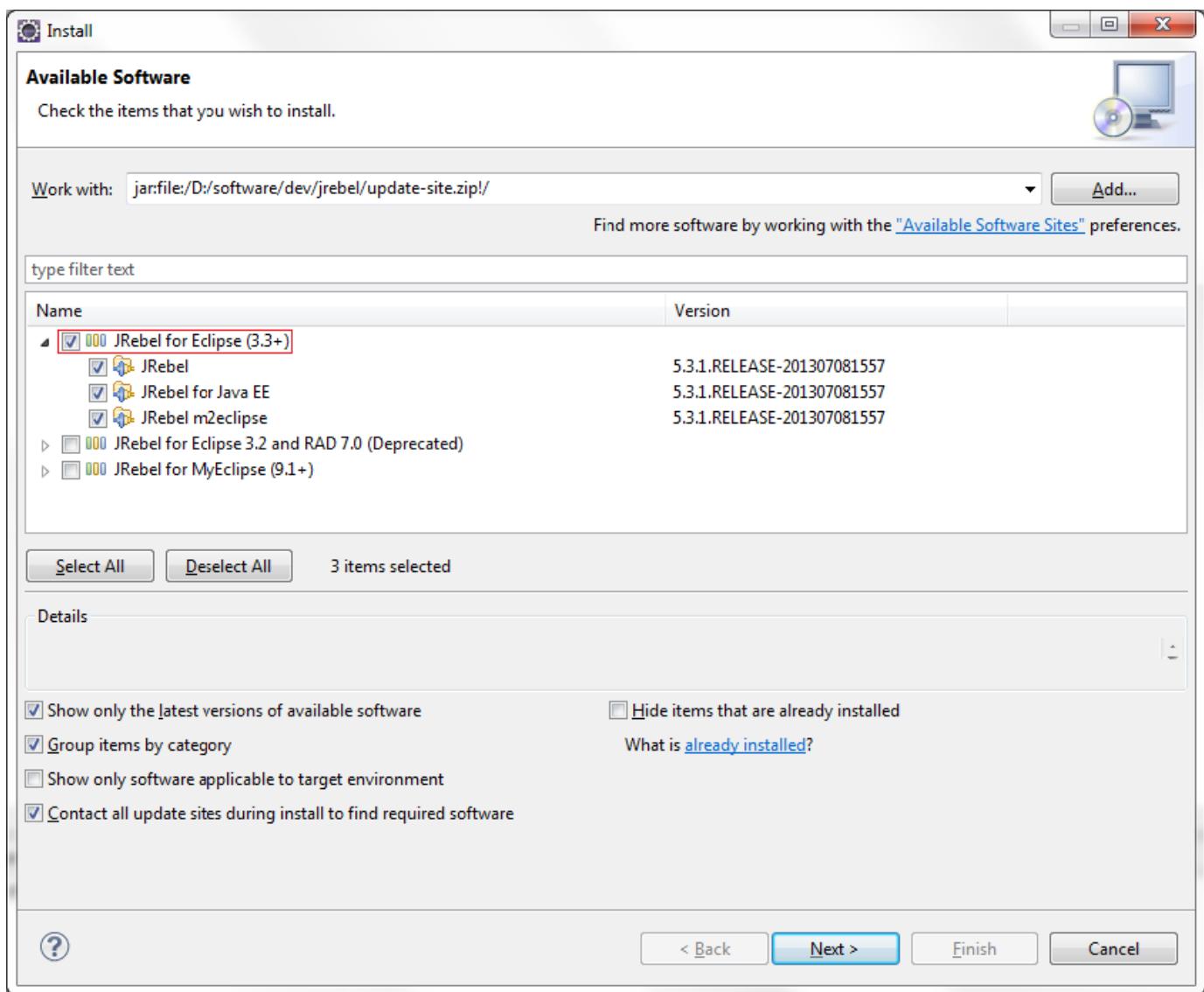
Step 3

Get Eclipse offline plugin from [JRebel-Plugin](#) or [Archive](#). The file name is update-site.zip. You can save this ZIP file in your local machine or it can be installed in Eclipse directly from this location. Please note that you should use Eclipse offline plugin installation for this.





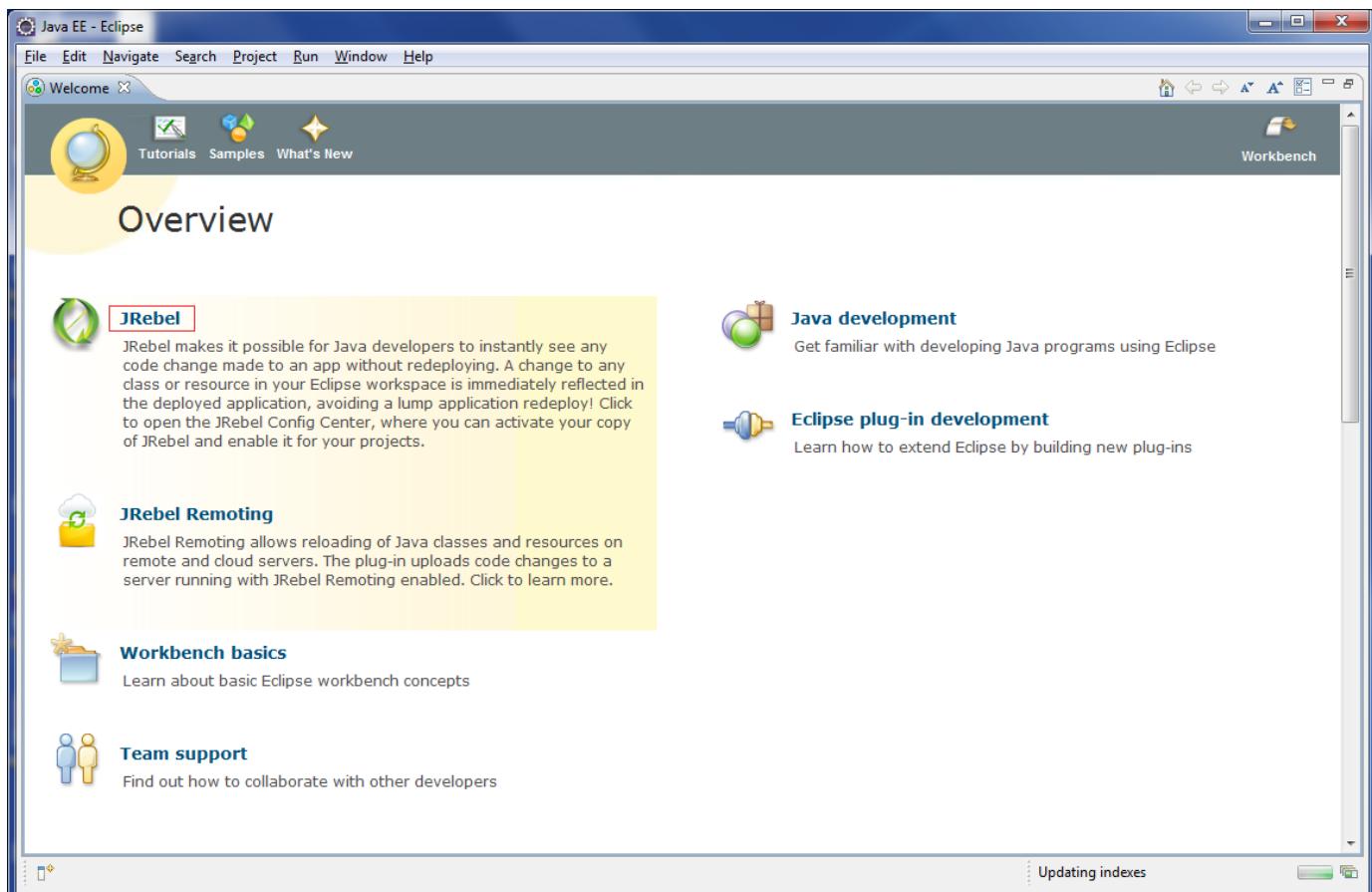
Please select the first option only for eclipse 3.3+. No other option should be selected for Eclipse only environment.



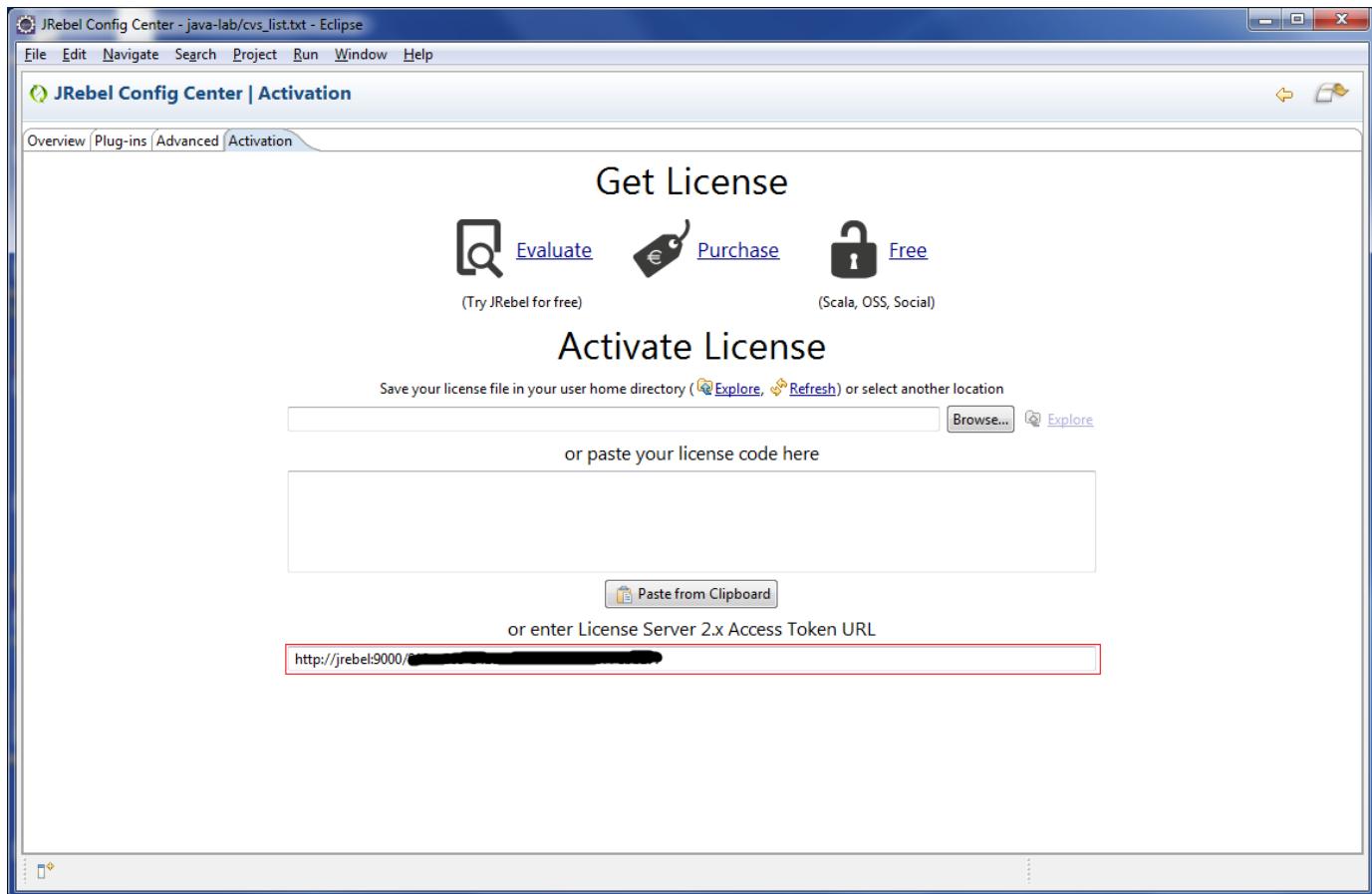
After successful installation, restart Eclipse.

Step 4

After restart you will see the Eclipse "Overview" page with JRebel. Click on JRebel as shown below:



If proper license is not installed for your Eclipse JRebel then you will be landed on JRebel license page as shown:



Paster the JRebel License Server 2.X Access Tokel URL provided to you (If you don't have the URL then ask JRebel admin for it; likely a designated finch team member). Eclipse JRebel plugin will be activated. It's now ready for development.

3.4.3 Setup Database

3.4.3.1 finch DB

3.4.3.2 Application DB

3.4.4 Bootstrap Application

Introduction

This page describes how to bootstrap the application development using finch infrastructure framework which also provides a set of Screens and Console Services out of the box.

Prerequisite

Following software must be installed in the machine where the blank application will be built:

- Java SE 6 or above
- Apache Maven 3.x.x (finch used), however other like gradle/SBT also can be used, however in this case dependency relation setup will be as per selected build framework
- Apache Tomcat application server 6.x or above
- Database (My SQL/Oracle/both)

Two example for Bootstrap Application Development available are:

- `blank` include Development IDE setup (eclipse) and new module
- `sample-app` with module.

Revision History

Version	Date	Description of Change	JIRA Issue Id	Author
1.0	30-Jul-2014	Initial Version	FCHII-731	Ashok Kumar Jha

3.4.4.1 Maven Archetypes

Introduction

This page describes how to create Maven Archetype finch framework. The application will have a default Dashboard and a set of Screens provided by finch framework out of the box.

Description

As this information is starting point to create Application using finch. So a separate page with required link is available [here](#).

And above link includes following:

1. Prerequisite
2. Steps to Build Blank Application Using finch(includes archetype)
3. Import Project in IDE (Eclipse)
4. Some Useful Maven Commands
5. Folder/Structure of the Blank Application
6. Maven Files
7. Maven Assembly File to Build War File
8. Other Files
9. Database Setup
10. Web Server setup with MY-SQL and Oracle

Reference:

External [link](#) for maven archetype

3.4.4.2 Add Module

Introduction

This page describes how to add a new module in application using finch framework.

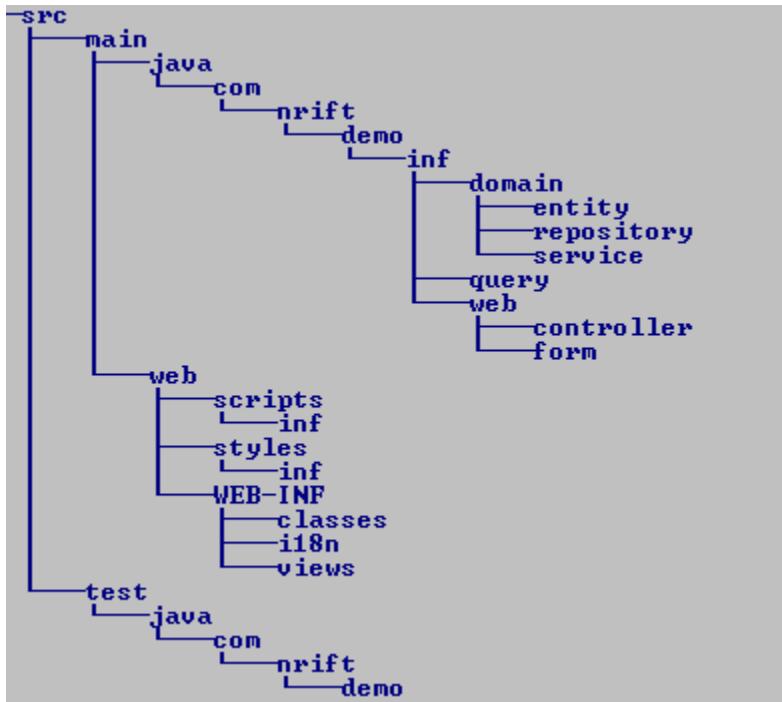
Assumption:

Application has been setup using [this link](#).

Steps to Add Module in Application using finch

To build application using finch framework, follow the steps mentioned below:

1. to create separate modules as per application needs. Steps to create separate module is mentioned below:
In our example we are creating an app called demo and every module in it will follow the following structure
Here inf is just a place holder, it will depends on application need.



To generate module with above directory structure, following steps need to be followed

- a. Open command prompt and navigate to finch-app (created using steps describe in Steps to Build Blank Application Using finch link) directory.
- b. Execute following command for each module.

```
mvn archetype:generate -DgroupId=[Java root package for Application] -DartifactId=[Application module name]  
-Dversion=[Application Module version] -DarchetypeArtifactId=[Archetype Artifact Id] -DarchetypeGroupId=[Archetype Group Id]
```

Example:

```
mvn archetype:generate -DgroupId=com.nrift.demo -DartifactId=demo-nnn -Dversion=1.0-SNAPSHOT -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart
```

ArchetypeArtifactId depends on application needs, therefor it may not be the same as specified here.

If pom.xml file in finch-app do not contain any entry for demo-nnn (as in our example, it will be your module name in your case) module, then please add this module in the pom.xml file under <modules> tag.

```

Example:
<project>

//other stuffs

<modules>

<module>finch-app-dist</module>

<module>demo-nnn</module>
</modules>

</project>

```

- c. Maven will also confirm groupId, artifactId information supplied to it, when it will ask for confirmation press 'Y'.
- d. If everything goes well, one directory will be created namely demo-nnn.
- e. Now create static resources like .js, .css, image etc. files under directory of your choice. For example all the .js files for demo-nnn, resides under src/main/web/scripts/inf directory and all the .css files resides under src/main/web/style/inf directory. For .js files create folders namely scripts and scripts/inf and for style sheet create folder styles/inf under src/main/web folder.
- f. Use [this link](#) (use finchro as user name and password) to download finch-console.zip file, then unzip this file and go to 'conf/META-INF/config' folder and change datasoucese.properties file and also copy MySql jar file into lib folder. Go to [this link](#) to generate Controller, Repository, Entity, JSNX etc files using CRUD tools, and copy files into respected directories.

During CRUD tools execution, user may face "too many connection" MySql exception. If this exception is encounter then please open MySql from command prompt and execute following command

```
set global max_connections=5000;
```

- g. Now configure pom.xml file of demo-nnn directory

Add following dependencies

```

<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-camel</artifactId>
    <version>5.6.0</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-pool</artifactId>
    <version>5.6.0</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-core</artifactId>
    <version>5.6.0</version>
    <scope>runtime</scope>
</dependency>

```

Also include respective database driver dependency in the above file.
For example:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.27</version>
    <scope>runtime</scope>
</dependency>
```

- h. Now configure assembly-web.xml file which resides under finch-app-dist/src/main/assembly folder.

For static resources, add the following filesets tag in assembly-web.xml file

fileset tags

```
<fileSet>
    <directory>../demo-nnn/src/main/web/scripts</directory>
    <outputDirectory>scripts/</outputDirectory>
</fileSet>
<fileSet>
    <directory>../demo-nnn/src/main/web/WEB-INF/i18n</directory>
    <outputDirectory>/WEB-INF/classes/i18n</outputDirectory>
</fileSet>
<fileSet>
    <directory>../demo-nnn/src/main/styles</directory>
    <outputDirectory>styles/</outputDirectory>
</fileSet>
<fileSet>
    <directory>../demo-nnn/src/main/web/WEB-INF/views</directory>
    <outputDirectory>/WEB-INF/views</outputDirectory>
    <excludes>
        <exclude>scripts/**</exclude>
    </excludes>
</fileSet>
<fileSet>
    <directory>src/main/web/</directory>
    <outputDirectory>/</outputDirectory>
</fileSet>
```

Also add the following dependencies for class files

Please add respective jar file for database driver in <include> tag. For example: <include>mysql:mysql-connector-java</include>

Dependency tag in assembly-web.xml file

```
<dependencySets>
    <dependencySet>
        <useProjectArtifact>false</useProjectArtifact>
        <scope>runtime</scope>
        <outputDirectory>/WEB-INF/lib/</outputDirectory>
        <includes>
            <include>com.nrift.demo:demo-nnn</include>
            <include>org.apache.activemq:activemq-camel</include>
            <include>org.apache.activemq:activemq-pool</include>
            <include>org.apache.activemq:activemq-core</include>
            <include>org.apache.geronimo.specs:geronimo-j2ee-management_1.1_spec</include>
        </includes>
    </dependencySet>
</dependencySets>
```

- i. Add following dependency in pom.xml of finch-app-dist folder

Dependency tag in pom.xml of dist folder

```

<dependencies>
    <dependency>
        <groupId>com.nrift.demo</groupId>
        <artifactId>demo-nnn</artifactId>
        <version>${project.version}</version>
    </dependency>
</dependencies>

```

- j. Change 'datasource.properties' and 'datasource-context.xml' file under 'finch-app-dist\src\main\resources\META-INF\config' directory.
Example: '[datasource.properties](#)' , '[datasource-context.xml](#)'
- k. Change package type of the module. Open pom.xml file of demo-nnn , change package type to jar [<packaging>jar</packaging>].
- l. To change name of final war file, open pom.xml file of finch-app-dist folder and look for finalName tag under profiles tag and change it suitable one. Same name should be given in finch-app/pom.xml file under <name> tag.
- m. Now execute mvn package command to generate final distribution. After successful execution of the command a war file will be generated in finch-app-dist/target folder.
- n. Deploy the war in the tomcat application server. To deploy the war copy the war file under '[TOMCAT-HOME]\webapps' folder.

3.4.5 Code Generation Using Tool

Introduction

In order to develop web application using finch, following required:

- Model (JPA Entities)
- CRUD operation => With persistence layer Create , Read,Update, Delete operation.
- Repository => For Interacting with Repository e.g. Database
- Service => To provide service to user
- Controller => To control the request from users
- jspx (view) => View part

So the finch infrastructure provide tool with infrastructure itself to expedite the project development project cycle.

Tool input in JSON (javascript object notation) format of all information required by the tool to generate automatic code base for already mentioned above.

Detail information for this tool available here.

Revision History

Version	Date	Description of Change	JIRA Issue Id	Author
1.0	30-Jul-2014	Initial Version	FCHII-732	Ashok Kumar Jha

3.4.6 Feature Wise Implementation

3.4.6.1 General Feature

Introduction

This section describes about the following general features:

- 3.4.6.1.1 Data Persistence Abstraction
- 3.4.6.1.2 Rule Engine
- 3.4.6.1.3 Scheduler
- 3.4.6.1.4 Application Access Restriction
- 3.4.6.1.5 Preference
- 3.4.6.1.6 Notification

- 3.4.6.1.7 Exception Handling Framework
- 3.4.6.1.8 Add a new module
- 3.4.6.1.9 Configure an Online/Batch Report
- 3.4.6.1.10 Enterprise specific db schema management
- 3.4.6.1.11 Enterprise specific implementation/customization
- 3.4.6.1.12 Handle of Concurrent operation
- 3.4.6.1.13 Manage Transaction
- 3.4.6.1.14 Multilingual i18n Support
- 3.4.6.1.15 New Batch Process

3.4.6.1.1 Data Persistence Abstraction

Introduction

finch provides basic abstraction for data persistence for storing and retrieving data into/from relational database. finch uses Java Persistence API (JPA) for data persistence. Application developers need to know JPA in order to use finch provided abstraction.

Abstraction for Data Persistence

The following steps need to be followed in order to opt for abstraction for data persistence.

Step-1 : Generate Entities

Developer designs the application data model as per the business requirement. Once the data model is designed, database tables are created with appropriate relationships among the tables. Java JPA entities can be generated for the tables using standard JPA generation tools. Eclipse provides a plug-in to generate JPA entities from database schema. finch recommends the following audit fields for each table:

Audit Field	Description
CREATED_BY	ID of the user who created this record
CREATION_DATE	Date when the record was created
UPDATED_BY	ID of the user who has updated the record last
UPDATE_DATE	Date when the record was last updated
APP_REGI_DATE	Application Date when record was created
APP_UPD_DATE	Application date when the record was last updated
CC_CHECK	A field to track version of the record. It is required for JPA based Optimistic concurrency control.

finch provides following base classes which are required to map these audit fields. These classes also manage the audit fields for entity creation and update. New entity classes can extends one of these classes, instead of mapping audit fields in their class.

Class Name	Description
FinchEntityBaseNoVersion	Extend this class when table contains audit fields, like CREATED_BY, CREATION_TIME, UPDATED_BY, UPDATE_DATE, APP_REGI_DATE and APP_UPD_DATE
FinchEntityBase	Extend this class when table contains all audit fields, like CREATED_BY, CREATION_TIME, UPDATED_BY, UPDATE_DATE, APP_REGI_DATE, APP_UPD_DATE and CC_CHECK
EntityBase	Extend this class when table contains audit fields, like CREATED_BY, CREATION_TIME, UPDATED_BY and UPDATE_DATE
EntityBaseWithVersion	Extend this class when table contains audit fields, like CREATED_BY, CREATION_TIME, UPDATED_BY, UPDATE_DATE and CC_CHECK

FinchEntityBaseNoVersion and FinchEntityBase classes contain an abstract method getComponent. Entity classes those extend one of these base classes must return correct component while implementing this method. It is used to retrieve application date.

A sample code snippet is as follows:

```
/**  
 * The persistent class for the TRD_TRADE database table.  
 */  
@Entity  
@Table(name="TRD_TRADE")  
public class Trade extends FinchEntityBase implements Serializable {  
    ...  
}
```

Step-2 : Develop Repository Classes

- Aggregate - Collection of entities that belong together. Each such aggregate has a common root called Aggregate Root. Clients can access non-root entities through the Aggregate Root entity. For example, in security trade business module, Trade entity will be the aggregate root and other entities like TaxFee, Commission etc will be part of Trade aggregate.
- Repository - Isolates domain objects from details of the database access code by providing an interface to access domain objects or entities. Repository contains method to add, delete, and find entities based on some criteria. Application should create repository for each aggregate root. finch provides an abstract base class RepositoryBase. It provides APIs for common DB operations. Developer should create repository classes by extending the RepositoryBase class.

A sample code snippet is as follows:

TradeRepository.java

```
@Repository  
public class TradeRepository extends RepositoryBase<Trade, Long> {  
    /**  
     * The <code>LOGGER</code> instance for this class.  
     */  
    private static final Log LOGGER = LogFactory.getLog(TradeRepository.class);  
  
    @PersistenceContext(unitName="sample-em")  
    private EntityManager em;  
  
    @Override  
    public EntityManager getEntityManager(){  
        return em;  
    }  
  
    public void saveTrade(Trade trade) throws FinchException{  
        ...  
    }  
}
```

Note

- finch supports multiple JPA entity managers. Each entity manager can point to different database schema. Entity managers like sample-em is defined in the datasource-context.xml.
- All finch repository classes use global-em entity manager. The global-em must be defined in the application Spring configuration file, like datasource-context.xml. Application can define their own entity manager and use that in their repository classes. They can use the global-em entity manager if finch tables and application tables are defined in the same database schema.

3.4.6.1.2 Rule Engine

Introduction

The behavior of any application developed using finch can be controlled by Rule Engine. It works on large number of attributes which have been externalized in the form of database tables and property files and gives a system behavior. Appropriate data configuration can alter the system output in finch.

Every financial transaction need to be processed in different applicable modules. Rule engine has been written to propagate the event in different applicable modules. It provides the best matching output with respect to given input criteria. For example, message flow for any event like Trade, Settlement, Corporate Action etc.

The output of the rule engine can be

- The best matching rule
- All the matching rules

To incorporate business intelligence in the system, so that the system can intelligently take decisions at run time, the rules are used. Rules can be defined as the set of explicit regulations or predefined conditions within a particular area of activity. The advantages of using Rule Engine in the system are as follows:

- System can dynamically determines the best rule applicable
- Input to the Rule Engine is dynamically created based on the attributes of a transaction

Rule consists of two parts:

- Input Criteria – attributes of a transaction defines the input of a rule
- Rule Output – the output is the best rule applicable which is selected based on the input criteria

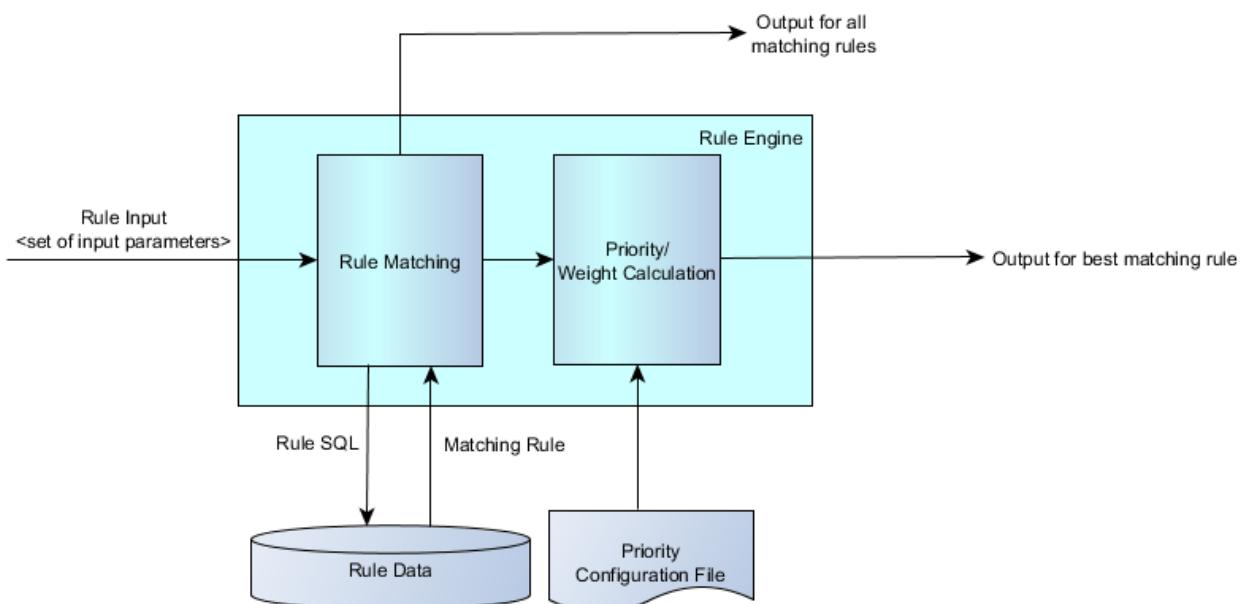
Attributes having weightage in a particular property file.

RuleFinder Interface used to get the list of Rules in ordered by their weight and get the Best Rule having the best weight.

Rule Engine Design

The Rule Engine is a software system that selects the best applicable business rule from the available set of rules based on the input criteria and applies the rule in run time production environment. The following points define how the Rule Engine works in a production environment:

- set of input attributes are defined based on which the rule is selected
- output of the rule is defined which is the best applicable rule based on the inputs
- rules are evaluated against the input attributes
- rule having the maximum number of matches with the input attributes is selected as the best applicable rule
- in order to resolve conflicts, relative weights are assigned to the rule input attributes
- Calculate weights for each matching rules and sort them based on calculated weights



Rule Usage

The rule can be used for the following reasons:

- Tax, Fee, Commission
- Settlement standing Instructions
- Posting rules in all accounting components etc.
- Message Routing Rule:
 - It is a rule engine that also plays the role of a workflow engine.
 - Some message attributes are processed in real time and the message destination component & queue is determined
 - Attributes used to identify the appropriate rule (Rule Input)
 - Enterprise
 - Source Component & Queue
 - Message Type
 - Transaction Type
 - Counterparty Type
 - Settlement Mode
 - Expected Rule Output
 - Destination Component & Queue

Rule Processor

Weight of a rule attribute (classifier) can be of the following types:

- Simple – Presence or absence of value indicates whether this classifier will have weight or not.
- Value Dependent – Actual value of the attribute determines the weight. Example of this category of attributes is the tree type properties like instrument-type. An instrument-type 'ST' has some static weight, but the dynamic weight depends on which node of the branch the type appears on.
Dynamic weight is maximum if the node is a leaf. General strategy adopted by the processor to assign weight is as follows:
 - If the property is specified in the configuration file, then the static weight associated with the property as specified in the configuration is applied (if the property value is non null).
 - If the property is specified as a multivalued property, then the processor proceeds as described below:
 - Strategy adopted to assign dynamic weight for an attribute is based on the fact that the weight depends on the value of an attribute against a set of possible values.
 - This class assumes that caller passes the name of the attribute and set of possible values by calling some method. Further, this processor assumes that the set of possible values of an attribute is ordered based on weight, such that most significant value in the context is at the beginning of the list and the least significant value is at the end.
 - The dynamic weight of an attribute (a) having a value (v) is then computed as follows:
Let $(v_1, v_2, v_3, \dots, v_n)$ be the set of possible values that the attribute (a) can assume. By first understanding, v_1 has most weight and v_n has the least weight. Find out the index of value v against the set of valid values. Let this index be i. Let s be the cardinality of the value set. Thus it follows that dynamic weight $d = s - i$

Use finch Rule Engine API for Sample Application

The following section describes how to use finch Rule Engine API to develop the sample application rule feature. Let's take the example of Message Router Rule.

Message Routing Rule

It is a rule engine that also plays the role of a workflow engine. Some message attributes are processed in realtime and the message destination component & queue is determined. Attributes used to identify the appropriate rule (Rule Input)* Enterprise are as follows:

- Source Component & Queue
- Message Type
- Transaction Type
- Counterparty Type
- Settlement Mode
- Expected Rule Output
- Destination Component & Queue

Steps to construct Message Router Rule using finch Rule Engine API:

1. Create REF_MESSAGE_ROUTING_RULE table

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 MESSAGE_ROUTER_RULE_PK	NUMBER(10,0)	No	(null)	1	(null)
2 ENTERPRISE_ID	VARCHAR2(6 BYTE)	No	(null)	2	(null)
3 MESSAGE_TYPE	VARCHAR2(10 BYTE)	Yes	(null)	3	(null)
4 SOURCE_COMPONENT	VARCHAR2(6 BYTE)	Yes	(null)	4	(null)
5 SOURCE_QUEUE_ID	VARCHAR2(10 BYTE)	Yes	(null)	5	(null)
6 COUNTER_PARTY_TYPE	VARCHAR2(10 BYTE)	Yes	(null)	6	(null)
7 SETTLEMENT_MODE	VARCHAR2(20 BYTE)	Yes	(null)	7	(null)
8 TRANSACTION_TYPE	VARCHAR2(50 BYTE)	Yes	(null)	8	(null)
9 DESTINATION_COMPONENT	VARCHAR2(10 BYTE)	Yes	(null)	9	(null)
10 DESTINATION_QUEUE_ID	VARCHAR2(50 BYTE)	Yes	(null)	10	(null)
11 STATUS	VARCHAR2(6 BYTE)	Yes	(null)	11	(null)
12 APP_REGI_DATE	DATE	No	(null)	12	(null)
13 APP_UPD_DATE	DATE	No	(null)	13	(null)
14 CREATED_BY	VARCHAR2(20 BYTE)	No	(null)	14	(null)
15 CREATION_DATE	DATE	No	(null)	15	(null)
16 UPDATED_BY	VARCHAR2(20 BYTE)	No	(null)	16	(null)
17 UPDATE_DATE	DATE	No	(null)	17	(null)

2. Create class Class Name:<e.g. MessageRouterRuleFinder.java. >

3. Create properties file <e.g. MessageRouterRuleFinder.properties> which contains the relative weight of classifiers of table < REF_MESSAGE_ROUTING_RULE>. MessageRouterRuleFinder is concrete implementation of message routing rule look-up.

To use the MessageRouterRuleFinder, first instantiate one bean as per Java Beans spec or using DynaBean which has the following properties:

- a. settlement_mode (value is of type String or may be null)
- b. counter_party_type (value is of type String or may be null)
- c. source_component (value is of type String or may be null)
- d. message_type (value is of type String or may be null)
- e. source_queue_id (value is of type String or may be null)
- f. transaction_type (value is of type String or may be null)

This bean instance serves as the data source which is used to build the query which will fetch rules from database.

MessageRouterRuleFinder extends abstract Class [com.nrft.finch.inf.rule.BasicRuleFinder](#) from finch.

```
public class MessageRouterRuleFinder extends BasicRuleFinder {
//Codes .....
//Codes .....
}
```

4. Set of properties expected by finder in source bean.(we are using [org.apache.commons.beanutils.DynaProperty](#) to set the properties).

```

public static final DynaProperty[] RULE_CLASSIFIERS = new DynaProperty[] {
    new DynaProperty("message_type"),
    new DynaProperty("counter_party_type"),
    new DynaProperty("source_component"),
    new DynaProperty("settlement_mode"),
    new DynaProperty("source_queue_id"),
    new DynaProperty("transaction_type");
}

```

The DynaClass of the bean to be used as source.(we are using [org.apache.commons.beanutils.BasicDynaClass](#) to set the properties).

```

public static final BasicDynaClass RULE_CLASSIFIER_CLASS = new BasicDynaClass(
    null, null, RULE_CLASSIFIERS);

```

5. Create the source bean which is used to build the criteria to select rule and also create two constructors for the following uses:
 - a. Construct an instance where source object to build query is null.
 - b. Construct an instance treating the passed object as source of building SQL query.

```

private Object source;

//Constructor
public MessageRouterRuleFinder() {
    super(MessageRouterRuleFinder.class);
}

//Constructor
public MessageRouterRuleFinder(Object src) {
    super(MessageRouterRuleFinder.class);
    if (src == null)
        throw new IllegalArgumentException("source object is null");
    source = src;
}

```

6. Clear the source object for building query.

```

public void reset() {
    source = null;
}

```

7. Set the source object for building query.

```

public void setSource(Object src) {
    source = src;
}

```

8. Declare method getSourceRules() which is the abstract method of [com.nrft.fin.ch.inf.rule.BasicRuleFinder](#) class. The getSourceRules() returns a list of [com.nrft.fin.ch.inf.rule.RuleEntry](#) objects which identify the source rules.

We are using [com.nrft.fin.ch.inf.rule.RuleRowMapper](#) class from finch which implements org.springframework.jdbc.core.RowMapper interface.[RuleRowMapper](#) class for mapping each row to a result object.
[RuleRowMapper](#) calls its parameterized constructor by passing the two arguments(RULE_CLASSIFIER_CLASS,

RouterRuleEntry.class)

```
protected List<? extends RuleEntry> getSourceRules() throws RuleException {  
  
    try {  
        String sql = getQuery();  
        return getJdbcTemplate()  
            .query(sql,  
                new RuleRowMapper(  
                    RULE_CLASSIFIER_CLASS,  
                    RouterRuleEntry.class));  
    } catch (Exception e) {  
        log.error("Exception fetching rules from db", e);  
        RuleException.throwException(e);  
        return null; // Required to satisfy Java compiler.  
    }  
  
}
```

We are using [JdbcTemplate](#) of Spring to interact with database. Following is the code snippet of how to use the jdbc connection using [JdbcTemplate](#). Use ReferenceName as "SAMPLE" passed as argument in [getReference\(\)](#)

```
private DataSource getDataSource() {  
    return Operation.getInstance().getReference("SAMPLE", DataSource.class);  
}  
  
private JdbcTemplate getJdbcTemplate() {  
    return new JdbcTemplate(getDataSource());  
}
```

9. Declare [getQuery\(\)](#) method which has called from [getSourceRules\(\)](#) method. The purpose of this method is to get sql query which returns the complete sql query to execute. Here we are using Java Object Oriented Querying/jooQ([org.jooq](#)) for constructing the SQL.

Note:

Explore <http://www.jooq.org/> to learn jooQ.

```

private String getQuery() throws Exception {

    if (source == null)
        throw new IllegalArgumentException("source object is null");
    DefaultQueryCriteria criteria = new RuleQueryCriteria(source, getRuleProcessor()
        .getPropertyFilter());

    String sql = DSL.using(SQLDialect.ORACLE,
        new Settings().withStatementType(StatementType.STATIC_STATEMENT))
        .select()
        .from("ref_message_routing_rules")
        .where(DSL.field("enterprise_id")
            .eq(Operation.getInstance()
                .getContext().getEnterpriseId()))
        .and(DSL.field("status").eq("NORMAL"))
        .and(criteria.getCriteria())
        .getSQL();

    if (log.isDebugEnabled())
        log.debug("Rule Query: " + sql);
    return sql;
}

```

10. Create the inner class RouterRuleEntry which extends [com.nrft.fin.ch.inf.rule.RuleEntry](#) which also extends BasicDynaBean of org.apache.commons.beanutils for accessing rule outputs which passed as argument in new RuleRowMapper(RULE_CLASSIFIER_CLASSES,RouterRuleEntry.class) on the method getSourceRules().

```

public static class RouterRuleEntry extends RuleEntry {

    private static final long serialVersionUID = 1L;

    /**
     * Standard Constructor for DynaBean interface.
     */
    public RouterRuleEntry(DynaClass c) {
        super(c);
    }

    /**
     * Access the primary key of the rule.
     *
     * @return the rule primary key
     */
    public long getRulePk() {
        return (Long)super.get("message_router_rule_pk");
    }

    /**
     * Access destination queue id as output of the rule.
     *
     * @return destination queue id as output of the rule
     */
    public String getDestinationQueueId() {
        return (String)super.get("destination_queue_id");
    }

    /**
     * Access destination component.
     *
     * @return destination component
     */
    public String getDestinationComponent() {
        return (String)super.get("destination_component");
    }
}

```

11. To construct the query criteria we need to use [com.nrft.fin.ch.inf.db.RuleQueryCriteria](#) class of finch which again extends [com.nrft.fin.ch.inf.db.DefaultQueryCriteria](#) of finch. In `getQuery()` method we instantiate `DefaultQueryCriteria` class and called the parameterized constructor of `RuleQueryCriteria`.

```
DefaultQueryCriteria criteria = new RuleQueryCriteria(source, getRuleProcessor().getPropertyFilter());
```

The arguments are:

- source -- the source bean which is used to build the criteria to select rule.
- other argument is to read `MessageRouterRuleFinder.properties` using another class from finch rule engine API is `RuleProcessor`([com.nrft.fin.ch.inf.rule.RuleProcessor](#)).

12. Then we call `getCriteria()` method of [com.nrft.fin.ch.inf.db.RuleQueryCriteria](#) to get the criteria for construction of full SQL.

```
criteria.getCriteria()
```

3.4.6.1.3 Scheduler

Introduction

finch infrastructure supports the feature of job scheduling. The scheduler can be of two type, periodical or one time and they are based on Quartz 2.2.0 scheduler. Scheduling activity is handled by the infrastructure with use of set API's and configuration, as mentioned below.

Step 1. Configuration

To use the finch scheduler, in application-context.xml and scheduler-context.xml file the scheduler to be configured. In application-context.xml file, add the scheduler package in the component scan and include the scheduler-context.xml file. In scheduler-context.xml file, define the SchedulerFactoryBean class and configure it's properties, as shown in the following code snippets:

application-context.xml

```
<context:component-scan base-package="... , com.nrft.fin.ch.inf.scheduler" />  
<import resource="classpath: META-INF/config/scheduler-context.xml"/>
```

scheduler-context.xml

```
<bean id="schedulerFactoryBean" class="org.springframework.scheduling.quartz.SchedulerFactoryBean"  
destroy-method="destroy">  
    <property name="quartzProperties">  
        <props>  
            <prop key="org.quartz.jobStore.class">org.quartz.impl.jdbcjobstore.JobStoreTX</prop>  
            <prop key="org.quartz.jobStore.driverDelegateClass">org.quartz.impl.jdbcjobstore.StdJDBCDelegate</prop>  
            <prop key="org.quartz.threadPool.class">org.quartz.simpl.SimpleThreadPool</prop>  
            <prop key="org.quartz.threadPool.threadCount">5</prop>  
            <prop key="org.quartz.threadPool.makeThreadsDaemons">true</prop>  
            <prop key="org.quartz.scheduler.makeSchedulerThreadDaemon">true</prop>  
            <prop key="org.quartz.scheduler.instanceName">schedulerFactoryBean</prop>  
            <prop key="org.quartz.jobStore.misfireThreshold">600000</prop>  
            <prop key="org.quartz.scheduler.instanceId">AUTO</prop>  
            <prop key="org.quartz.jobStore.isClustered">true</prop>  
            <prop key="org.quartz.jobStore.clusterCheckinInterval">20000</prop>  
            <prop key="org.quartz.jobStore.dataSource">ds</prop>  
            <prop key="org.quartz.jobStore.tablePrefix">INF_QRTZ_</prop>  
            <prop key="org.quartz.dataSource.ds.driver">${jdbc.database.driverClass}</prop>  
            <prop key="org.quartz.dataSource.ds.URL">${jdbc.components.url.GLOBAL}</prop>  
            <prop key="org.quartz.dataSource.ds.user">${jdbc.components.userName.GLOBAL}</prop>  
            <prop key="org.quartz.dataSource.ds.password">${jdbc.components.password.GLOBAL}</prop>  
        </props>  
    </property>  
    <property name="autoStartup" value="false"/>  
    <property name="waitForJobsToCompleteOnShutdown" value = "true" />  
  </bean>  
</beans>
```

The following table lists the various configuration properties, which have been defined in the above code snippets:

Property	Value	Description
org.quartz.jobStore.class	org.quartz.impl.jdbcjobstore.JobStoreTX	The Job Store Class, which the quartz scheduler is going to use. JobStoreTX is used for persisting the jobs in database.
org.quartz.jobStore.driverDelegateClass	org.quartz.impl.jdbcjobstore.StdJDBCDelegate	The delegation class. StdJDBCDelegate is compatible with all major databases.
org.quartz.threadPool.class	org.quartz.simpl.SimpleThreadPool	Thread pool class used.
org.quartz.threadPool.threadCount	50	Number of threads available for scheduler job.
org.quartz.scheduler.instanceName	schedulerFactoryBean	Instance name of the scheduler.
org.quartz.jobStore.misfireThreshold	600000	Job misfire threshold value.
org.quartz.scheduler.instanceId	AUTO	Instance ID of the scheduler. The application automatically assigns the Instance ID.
org.quartz.jobStore.isClustered	true	It ensures that the Scheduler works in clustered mode. When the scheduler is configured in cluster mode then the machines in the cluster should have the identical time.
org.quartz.jobStore.clusterCheckinInterval	20000	Cluster scan checks the interval.
org.quartz.jobStore.dataSource	ds	The data source name.
org.quartz.jobStore.tablePrefix	INF_QRTZ_	The prefix for the table name for the quartz related database tables.
org.quartz.dataSource.ds.driver	jdbc.database.driverClass	The database driver class value, as mentioned in the datasource.properties file.
org.quartz.dataSource.ds.URL	jdbc.components.url.GLOBAL	URL for the database connection value as, mentioned in the datasource.properties file.
org.quartz.dataSource.ds.user	jdbc.components.userName.GLOBAL	User name of the database schema as, mentioned in the datasource.properties file.
org.quartz.dataSource.ds.password	jdbc.components.password.GLOBAL	Password of the database schema, as mentioned in the datasource.properties file.

Note:

The configuration details are available at: <http://quartz-scheduler.org/documentation/quartz-2.x/configuration/ConfigJobStoreTX>

Step 2 Create Job to Schedule

Once the above-mentioned configuration is done, the user needs to create a Job class which would act as the entry class. This class needs to implement an interface called Job. This interface has a method called execute(JobExecutionContext arg0), which needs to be implemented in the Job class. This method acts as the entry point for the Job execution. The following code snippets display an example of a Sample Job class:

```

import org.quartz.Job;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;

public class SampleJob implements Job {

    @Override
    public void execute(JobExecutionContext arg0) throws JobExecutionException {
        System.out.println("**** Sample Quartz Job ");
    }
}

```

Step 3 Schedule the Job

Currently either One-time Job or Routine Job can be scheduled. The following steps show how to schedule a Job using the finch Scheduler.

Schedule an One-time Job

The following code snippets show how to schedule an one-time Job:

```
FinchScheduler scheduler = Operation.getInstance().getReference(FinchScheduler.class);
```

API:

```
FinchScheduler:initOneTimeJob(Class<? extends Job> jobClass, String jobId, String jobGroupId, Map<String, Object> jobParam, Date jobRunDate, boolean doNothingOnJobMisfire)
```

Parameter	Description
jobClass	The Class for the Job which is going to be scheduled. This class should implement the Job interface.
jobId	The Job Id for the scheduled Job.
jobGroupId	The Job Group Id for the scheduled Job.
jobParam	Job parameter map.
jobRunDate	The Job execution date.
doNothingOnJobMisfire	Whether the misfired job to be re-executed or not. To re-execute teh job, select value as false and select true not to re-execute.

Example:

```

Date jobRunDate = ...

scheduler.initOneTimeJob(SampleJob1.class, "job1", "group1",null, jobRunDate, false);

scheduler.doStart(); // Start the scheduler

```

Schedule a Routine Job

The following code snippets show how to schedule a Routine job:

```
FinchScheduler scheduler = Operation.getInstance().getReference(FinchScheduler.class);
```

API:

```
FinchScheduler:initRoutineJob(Class<? extends Job> jobClass, String jobId, String jobGroupId, Map<String, Object> jobParam, String cronExpression, boolean doNothingOnJobMissfire, String calendarName)
```

Parameter	Description
jobClass	The Class for the Job which is going to be scheduled. This class should implement the Job interface.
jobId	The Job Id for the scheduled Job.
jobGroupId	The Job Group Id for the scheduled Job.
jobParam	Job parameter map.
cronExpression	The Cron Expression for the Routine Job. finch Scheduler uses Cron Expression for execution schedule for the Routine Job. To build the cron expression, go to: http://www.cronmaker.com/
doNothingOnJobMissfire	This parameter takes a boolean value. If the value is set as true, then all the miss-fired Job will not get triggered once the scheduler is re-activated. However the same will get triggered as per their execution schedule. If the value is false, then all the miss-fired Job will get triggered while the scheduler is re-activated
calendarName	This parameter contains the name of the calendar which the user has setup. In case of calendar not being used, specify the value as null.

Example (executing Job at one minute interval):

```
scheduler.initRoutineJob(SampleJob.class, "job1", "group1", null, "0 0/1 * 1/1 * ? *", true, null);
scheduler.doStart(); // Start the scheduler
```

Use Calendar

The calendar is setup with the scheduler to notify that on the given dates the job will not be executed. To use this functionality, create a calendar class by extending the AbstractBaseCalendar class and then use the setupCalendar(String calendarName, Calendar calendar) method of FinchScheduler class. The following code snippets if for creating the calendar class:

```

import java.util.Date;
import java.util.Set;
import java.util.TreeSet;

public class SampleHolidayCalendar extends AbstractBaseCalendar {
    private static final long serialVersionUID = 4757582188930859546L;
    private Set<Date> excludedDates = new TreeSet<>();

    protected Set<Date> getExcludedDates() {
        return excludedDates;
    }

    public void setExcludedDates(Date[] excludedDates) {
        for(Date dt : excludedDates){
            addExcludedDate(dt);
        }
    }

    protected void addExcludedDate(Date dt) {
        Date date = getStartOfDayJavaCalendar(dt.getTime()).getTime();
        this.excludedDates.add(date);
    }

    @Override
    protected boolean isDateIncluded(Date date) {
        return !excludedDates.contains(date);
    }
}

```

API:

`FinchScheduler:setupCalendar(String calendarName, Calendar calendar)`

Example using Sample Holiday Calendar

```

Date[] excludedDates = ...
SampleHolidayCalendar sampleHolidayCalendar = new SampleHolidayCalendar();
sampleHolidayCalendar.setExcludedDates(excludedDates);

scheduler.initRoutineJob(SampleJob.class, "job1", "group1", null, "0 0/1 * 1/1 * ? *", true, "myCalendar");

scheduler.doStart(); // Start the scheduler

```

Un-Schedule a Job

finch scheduler provides API by which a scheduled job can be un-scheduled. Once a job is un-scheduled the same wont execute further. To un-schedule a job, method called `doUnscheduleJob(String jobId, String jobGroupId)` of `FinchScheduler` class is executed. The API and example of un-scheduling job has been displayed below.

API:

```
FinchScheduler:doUnscheduleJob(String jobId, String jobGroupId)
```

Example:

```
finchScheduler.doUnscheduleJob("job1","group1")
```

API: Scheduler Methods

The following table lists the various methods provided by the finch scheduler.

Method	Example	Description
FinchScheduler:initRoutineJob(Class<? extends Job> clazz, String jobName, String group, Map<String, Object> jobParam, String cronExpression, boolean doNothingOnJobMissfire, String calendarName)	scheduler.initRoutineJob(SampleJob.class, "job1", "group1",null,"0 0/1 * 1/1 * ? *", true, "myCalendar")	Initialize and setup routine job. In case of calendar not being used, set the value of calendarName as null.
FinchScheduler:initOneTimeJob(Class<? extends Job> clazz, String jobName, String group, Map<String, Object> jobParam, Date jobRunDate, boolean doNothingOnJobMisfire)	scheduler.initOneTimeJob(SampleJob1.class, "job1", "group1",null, jobRunDate, false);	Initialize and setup one time job.
FinchScheduler:doUnscheduleJob(String jobId, String jobGroupId)	scheduler.doUnscheduleJob("job1","group1")	Un-schedule a job from the scheduler.
FinchScheduler:doStart()	scheduler.doStart()	Starts the scheduler.
FinchScheduler.startDelayed(int seconds)	scheduler.startDelayed(60)	Delay the start of the scheduler by n seconds.
FinchScheduler:doStop()	scheduler.doStop()	Stops the scheduler and destroys all the Threads including the Scheduler Factory Bean. Note: In case do Stop() is called and Job scheduling needs to be done, the server needs to be restarted to update the Scheduler Factory Bean.

FinchScheduler:doStop(boolean waitForJobsToComplete)	scheduler.doStop(true)	<p>Stops the scheduler and destroys all the threads including the Scheduler Factory Bean. If the value of the boolean is true, then scheduler would shutdown after all the jobs are executed. If the value is false, then the scheduler will shutdown immediately.</p> <p>Note: In case do Stop() is called and Job scheduling needs to be done, the server needs to be restarted to up the Scheduler Factory Bean.</p>
FinchScheduler:doInterrupt(String jobId, String jobGroupId)	scheduler.doInterrupt("job1","group1")	<p>Interrupts a running Job.</p> <p>Note: To use this the job class must implement the InterruptableJob interface.</p>
FinchScheduler:doPause(String jobId, String jobGroupId)	scheduler.doPause("job1","group1")	Pause a running job.
FinchScheduler:doResume(String jobId, String jobGroupId)	scheduler.doResume("job1","group1")	Resume a paused job.
FinchScheduler:doExecuteNow(String jobId, String jobGroupId)	scheduler.doResume("job1","group1")	Execute a scheduled job immediately.
FinchScheduler:setupCalendar(String calendarName, Calendar calendar)	<pre>Date[] excludedDates = ... SampleHolidayCalendar sampleHolidayCalendar = new SampleHolidayCalendar(); sampleHolidayCalendar.setExcludedDates(excludedDates); scheduler.setupCalendar("myCalendar", sampleHolidayCalendar)</pre>	Setup a calendar so that Jobs don't get executed on the specified dates in the calendar.
FinchScheduler:getJobTriggerStatus(String jobId, String jobGroupId)	scheduler.getJobTriggerStatus("job1","group1")	<p>Gets the trigger status for the job. The status values are as follows:</p> <ul style="list-style-type: none"> • BLOCKED • COMPLETE • ERROR • NONE • NORMAL • PAUSED

Step 4 Console Usage

Configuration

In order to use the SchedulerJobManager and SchedulerService, the following configuration needs to be done in console-executable-mapping.properties file.

```
console-executable-mapping.properties
SchedulerService = com.nrft.fin.ch.inf.scheduler.SchedulerConsoleService
SchedulerJobManager = com.nrft.fin.ch.inf.scheduler.SchedulerJobManager
```

Scheduler Console Service

The services and associated commands have been listed in the following table:

Service	Command
Service Controller	launch ServiceController
Start Scheduler Service	launch SchedulerService
Stop Scheduler Service	launch StopService -s SchedulerService

Scheduler Job Manager

finch scheduler can be used from the console application. To use the scheduler from console, the following activity needs to be performed:

1. Start the Scheduler Service
2. Use the Scheduler Job Manager to schedule the job. Job can be of two types, one-time and routine job.

The SchedulerJobManager comes with number of command line arguments by which the job can be scheduled. The following table lists the command line arguments:

Argument Key	Description
-o	Valid options are: <ul style="list-style-type: none">• SCHEDULE• UNSCHEDULE• HELP
-jn	To specify the job name.
-gn	To specify the job group name.
-cn	To specify the job class.
-ce	To specify the cron expression for routine jobs.
-cal	To specify the calendar name for the scheduler.
-rt	To specify the run time for the job. This is going to be used for one time job.
-jp	To specify the job parameter.

-mf	When this option is used, all the misfired jobs won't get triggered. However when the option is not used then the misfired job would get triggered immediately when the scheduler service is up. Example: If a job is scheduled to run at 9:00 pm and at that point of time, the job was not triggered then the scheduler will wait for the job to get triggered till the time set in the org.quartz.jobStore.misfireThreshold property of the configuration. Once this time gets elapsed, the job is treated as misfired job. The misfired job would get ignored, and the same would get triggered in the next scheduled time (in case of routine job). However if the option is not used then the misfired job would get automatically triggered once the scheduler service is up and will not wait for the next schedule time for firing the job.
-----	---

Example: Scheduling a Job - Execute Job on 20th second of each minute

```
launch SchedulerJobManager -o SCHEDULE -jn pjob1 -gn pgroup1 -cn
com.nrift.finch.inf.scheduler.ConsoleProcessExecutorJob -ce "20 * * * ?" -jp "notepad.exe"
```

Example: Scheduling a Job - Execute Job instantly and execute it every minute

```
launch SchedulerJobManager -o SCHEDULE -jn pjob1 -gn pgroup1 -cn
com.nrift.finch.inf.scheduler.ConsoleProcessExecutorJob -ce "0 0/1 * 1/1 * ?" -jp "notepad.exe"
```

Example: Schedule One Time Job

```
launch SchedulerJobManager -o SCHEDULE -jn pjob1 -gn pgroup1 -cn
com.nrift.finch.inf.scheduler.ConsoleProcessExecutorJob -rt "09-12-2013 13:05:00" -jp "notepad.exe"
```

Example: Un - Schedule a Job

```
launch SchedulerJobManager -o UNSCHEDULE -jn pjob1 -gn pgroup1
```

Example: Display Help

```
launch SchedulerJobManager -o HELP
```

Note:

Scenario	Result
<ul style="list-style-type: none"> a job is scheduled without the -mf option to run the job on a particular time at the moment when the job needs to run, the scheduler service is down the scheduler service is up but the job scheduled time is already over 	The job which was earlier scheduled to run would be triggered immediately.

When a job is scheduled without the -mf option to run the job on a particular time, and at that moment the scheduler service is down, but at a latter stage (after the job scheduled time) the scheduler service was up, then the job which was earlier scheduled to run would be triggered immediately.

Using the Console Process Executor Job

The following snippet of code is for using the console process executor job:

```
FinchScheduler scheduler = Operation.getInstance().getReference(FinchScheduler.class);
Map<String, Object> paramMap = new HashMap<String, Object>();
paramMap.put("jobParamKey", "notepad.exe");
scheduler.initRoutineJob(ConsoleProcessExecutorJob.class, "job1", "group1", paramMap, "20 * * * * ?", false, null);
```

Specifying Base Directory Path for Console Executable

To specify the base directory for the console executable job, mention the path value in the finch.console.baseDirPath system property. -D option of java.exe from command line is used to set the mentioned system property.

Example:

```
java -Dfinch.console.baseDirPath=d:\custom_path\ ....
```

3.4.6.1.4 Application Access Restriction

Introduction

Applications might require to impose temporary restriction based on user interface or console processes business logic. For example, in case of financial application, during daily closing, application might want to restrict entry and update operation. This restriction is additionally imposed on top of the normal access control (role based access control and record level access control).

Implementation

The implementation of Access Restriction can be web or console based.

Web

In finch, UI screen navigation can be started from following points:

- UI Menu
- Dashboard
- Query Result - Action Consolidation
- Bulk Action Wizard

In finch framework, abstract controllers exist for all the actions like Entry, Amend, Cancel, Query and Bulk. finch enhances these abstract controllers to check if access to the operation should be permitted or not for the given context. The context includes user principal, component, action type (QUERY, EDIT etc) and screen ID etc. finch provides a bean AccessRestrictionCheckService and AccessRestrictionHandler interface. AccessRestrictionCheckService' is a collection of AccessRestrictionHandler objects. Application provides implementation for AccessRestrictionHandler. Refer to the following code snippet in this regard:

```

public interface AccessRestrictionHandler{
    /**
     * Throw an exception with proper error message in case access is not
     * permitted for given context.
     */
    public void handleAccess(AccessContext accessContext) throws FinchException;
}

```

While finch decides to check access, it calls handleAccess of all objects of AccessRestrictionHandler. It might fail in the fast invocation. If any of the AccessRestrictionHandler objects throws exception then access is restricted and an error message is displayed as per the thrown exception.

If no handler is configured or AccessRestrictionCheckService bean is not provided in Spring context, access is not restricted.

Console

Batch process calls AccessRestrictionCheckService during start-up. If application throws any exception in the configured AccessRestrictionHandler, the batch is terminated with specific error code.

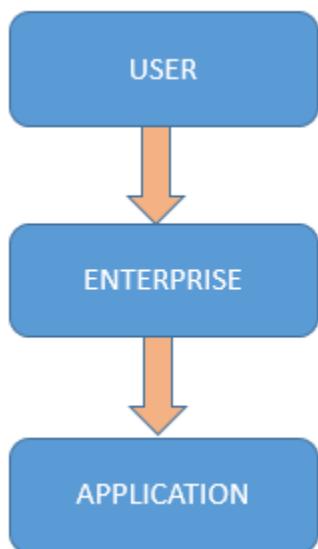
3.4.6.1.5 Preference

Introduction

Preference or personalization feature allows the user or client organization (enterprise) of an application to override default behavior or configuration of the application. For example, English is defined as default language for an application. User like to see the application in Japanese. The language can be changed using the preference feature. For the user, the application will be displayed in Japanese language and for the other users, who have not change the language, the application will be displayed in English. finch provides preference features out-of-the-box. But it is possible for the application to define new set of configurations for preference.

Preference Levels

finch preference has following three levels:



Generally, preference value is first searched in the user level. If not present in the user level then it is looked up in the enterprise level. Finally if it is not defined in both user and enterprise level, it is retrieved from the application level.

User Preference

User level preference is stored using Java Preference.

Note:

By default, Java Preference stored preference data in:

- Windows Registry for Windows OS
- file system for LINUX or UNIX based OS

Enterprise Preference

Enterprise preference is stored in a XML file named <Enterprise ID>-preference.xml. For example, for an enterprise with enterprise ID XYZ, the preference is stored in XYZ-preference.xml file. This file is placed under META-INF/pref directory in the class path. In a WAR file, this file will be located under <WAR Root>/WEB-INF/classes/META-INF/pref/.

Application Preference

Application preference is also stored in a XML file named app-pref.xml. This file is placed under META-INF/pref directory in the class path. In a WAR file, this file will be located under <WAR Root>/WEB-INF/classes/META-INF/pref/.

Add New Preference Field

To add new preference field or property:

1. Add the entry in the keys.properties file located under META-INF/pref folder
2. Define default value of the preference field in the application preference file

Add Module Specific Preferences

Application developer can append additional module specific preference to finch. Click [here](#) to know the process of customizing user preference.

3.4.6.1.6 Notification

Introduction

finch infrastructure has the functionality of publishing notifications. It is based on AKKA and Atmosphere. The publication of notifications are handled by finch by an API and the necessary configurations have been discussed below.

Step 1. Configuration

Step 1a:

Add the bean definition, as displayed in the following code snippet, to the spring application-context configuration file for both web and console.

```
Bean Definition  
<bean id="actorSystem" class="akka.actor.ActorSystem" factory-method="create" lazy-init ="true">  
    <constructor-arg type="java.lang.String" value="finchActorSystem" />  
</bean>
```

Also add the following bean in application-context.xml for console. It is mandatory to include.

```
application-context.xml  
<bean id ="notificationServerService" class = "com.nrft.fin.ch.inf.notification.server.NotificationServerService" />
```

Step 1b:

Add the bean definition, as displayed in the following code snippet, to the spring application-context configuration file for web.

Session Listener

```
<bean name="authenticationSuccessListener"
      class="com.nrft.fin.ch.inf.push.service.listner.FinchNotificationEventListner" />
```

Step 1c:

Add an entry in the table INF_ACTION_ROLE_PARTICIPANT for the action FCHNFT.

Step 2. Creating Notifications

Application developer can create a Group, which might consist of one or more Employee as participants. This should be specified by the application in the INF_EMPLOYEE_GROUP_PCPT table. A notification is denoted by the domain object com.nrft.fin.ch.inf.domain.entity.Notification.

A Notification object should also consist of a set of com.nrft.fin.ch.inf.domain.entity.NotificationDestination entities. A NotificationDestination entity holds all the information relating to the delivery of a notification. Each NotificationDestination entity is set with a group or employee along with the destination channelType(i.e, whether it is Dashboard or Email). It must be noted here that for a certain NotificationDestination entity both the Employee and Group cannot be set. In the case of a email notification, the default email address of the recipient is also set here by setting the channelAddress. All the fields are set by calling their setter methods.

Note:

If a notification contains Notification Destinations with duplicate channel type, the notification will be delivered multiple time to the same channel type.

The Notification entity can also consist of attachments in the case of a mail notification and in that case the parent entity Notification will have reference to a List of com.nrft.fin.ch.inf.domain.entity.NotificationAttachment entities. Each entity relates to a single attachment.

The attachmentName is the name of the attachment that will be shown in the mail, whereas attachmentType is the mime-type of the file that is to be attached. The actual file is to be provided as an byte Array and is set in the variable attachmentContent. As with NotificationDestination, each of the field is set with the help of the corresponding setter method.

It must be noted that if a dashboard notification contains attachments then in that case an IllegalStateException will be thrown.

If a notification is created with the sender having a consoleldentity, it is prescribed that the sender should have the enterprise and the branch set properly.

A Notification object contains a ExtRefNumber that an application must set before passing it on to finch. The following code snippets provides an example of creating notification :-

DASHBOARD NOTIFICATIONS WITHOUT MAIL ATTACHMENTS

```
NotificationDestination notifDest = new NotificationDestination();
notifDest.setChannelType(ChannelType.EMAIL.toString());
notifDest.setDeliveryStatus('y');
notifDest.setGroup(group);

NotificationDestination notifDest2 = new NotificationDestination();
notifDest2.setChannelType(ChannelType.DASHBOARD.toString());
notifDest2.setDeliveryStatus('y');
notifDest2.setGroup(group);

Notification notif = new Notification();
notifDest2.setNotification(notif);
notifDest.setNotification(notif);
notif.setExtRefNumber(new Random().nextInt());
notif.setDetail("Notification");
notif.setSummary("TRade Cancelled");
notif.setSourceType("TRD");
notif.setNotificationDestinations(Sets.newHashSet(notifDest,notifDest2));
pushService.push(notif, Operation.getInstance().getContext().getCallerIdentity());
```

MAIL NOTIFICATIONS WITH ATTACHMENTS

```
Notification notif1 = new Notification();
notif1.setExtRefNumber(new Random().nextInt());
notif1.setDetail("");
notif1.setSummary("Testing Multiple Mail attachments");
notif1.setSourceType("TRD");
notif1.setNotificationType(NotificationType.INFORMATION.toString());
NotificationDestination notifDest1 = new NotificationDestination();
notifDest1.setChannelType(ChannelType.EMAIL.toString());
notifDest1.setDeliveryStatus('y');
notifDest1.setChannelAddress("zeeshank@nrifintech.com");
notifDest1.setGroup(1L);
NotificationDestination notifDest2 = new NotificationDestination();
notifDest2.setChannelType(ChannelType.EMAIL.toString());
notifDest2.setDeliveryStatus('y');
notifDest2.setChannelAddress("zeeshank@nrifintech.com");
notifDest2.setGroup(2L);
notifDest1.setNotification(notif1);
notifDest2.setNotification(notif1);
notif1.setNotificationDestinations(Sets.newHashSet(notifDest1,notifDest2));

List<NotificationAttachment> notificationAttachments = Lists.newArrayList();

NotificationAttachment notificationAttachment1= new NotificationAttachment();
notificationAttachment1.setAttachmentName("finch group picture");
notificationAttachment1.setAttachmentType("image/jpeg");
notificationAttachment.setAttachmentName("finch group picture");
try {
    notificationAttachment1.setAttachmentContent(Files.toByteArray(new File("D:/finch_group_pic.jpg")));
    notificationAttachment1.setNotification(notif1);
    notificationAttachments.add(notificationAttachment1);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

NotificationAttachment notificationAttachment2= new NotificationAttachment();
notificationAttachment2.setAttachmentName("pdf attachment");
notificationAttachment2.setAttachmentType("application/pdf");
notificationAttachment.setAttachmentName("finch group picture");
try {
    notificationAttachment2.setAttachmentContent(Files.toByteArray(new File("D:/responsive_web_design.pdf")));
    notificationAttachment2.setNotification(notif1);
    notificationAttachments.add(notificationAttachment2);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

notif1.setNotificationAttachments(notificationAttachments);
pushService.push(notif1, Operation.getInstance().getContext().getCallerIdentity());
```

Note:

If a notification is created with a tooltip description, it is prescribed that the application developer should not set only

empty/blank html expression (e.g. <div></div> or etc) in the setDetail method of Notification object.

Step 3. API for publishing Notifications

The following code snippet is public API for publishing notification:

```
PushService.push(Object pushObject, Identity identity)
```

In order to publish notifications the **PushObject** should be of type `com.nrft.fin.ch.inf.domain.entity.Notification`. The following code snippet to be used for obtaining the **PushService**:

```
Operation.getInstance().getContext().getReference(PushService.class);
```

The **PushService** returns a `List<Employee>` which contains all the employees to whom the notifications will be published. After returning the list, the subsequent actions are done asynchronously and so any error in the subsequent flow is not reported. It means that the list returned indicates that for the returned list finch will try to publish notifications.

Step 4. Modifications in web.xml

For Notification to work properly the following code must be added to the web.xml file. Please make sure that the AtmosphereFilter is the first filter in the web.xml file:

```
<filter>
    <description>AtmosphereFilter</description>
    <filter-name>AtmosphereFilter</filter-name>
    <filter-class>org.atmosphere.cpr.AtmosphereFilter</filter-class>
    <async-supported>true</async-supported>
    <init-param>
        <param-name>org.atmosphere.cpr.broadcasterClass</param-name>
        <param-value>org.atmosphere.cpr.DefaultBroadcaster</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.useStream</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.useWebSocketAndServlet3</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.cpr.AtmosphereInterceptor</param-name>
        <param-value>com.nrft.fin.ch.inf.push.FinchAtmosphereInterceptor</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.cpr.CometSupport.maxInactiveActivity</param-name>
        <param-value>30000</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.cpr.broadcaster.shareableThreadPool</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.cpr.broadcaster.maxProcessingThreads</param-name>
        <param-value>2</param-value>
    </init-param>
    <init-param>
        <param-name>org.atmosphere.cpr.broadcaster.maxAsyncWriteThreads</param-name>
```

```

<param-value>2</param-value>
</init-param>
<init-param>
<param-name>org.atmosphere.cpr.sessionSupport</param-name>
<param-value>true</param-value>
</init-param>

<!-- For JBOSS only
<init-param>
<param-name>org.atmosphere.cpr.asyncSupport</param-name>
<param-value>org.atmosphere.container.Servlet30CometSupport</param-value>
</init-param>
-->
</filter>

<filter-mapping>
<filter-name>AtmosphereFilter</filter-name>
<!-- Any mapping -->
<url-pattern>/push</url-pattern>
</filter-mapping>

<!-- Change to be done in the Referrer filter, '/push' should be excluded in the Referrer filter and async support must be
set as true-->
<filter>
<filter-name>referrerFilter</filter-name>
<filter-class>com.nrft.finch.inf.web.filter.ReferrerFilter</filter-class>
<async-supported>true</async-supported>
<init-param>
<param-name>exclude</param-name>
<param-value>.htm,/push,/report,/exportHoliday</param-value>
</init-param>
</filter>
```

```

<listener>
  <listener-class>com.nrft.finchnf.startup.web.listner.FinchSessionListner</listener-class>
</listener>

```

Step 5. Supported Servers

The Notification Service has been tested with 7.0.47 and Jboss 7x. It must be noted that the application container must implement Servlet specification 3.0 or above. If an application is using Tomcat 7, it is recommended to use Tomcat versions 7.0.47 to 7.0.53. The following table lists the supported servers:

Server Name	Version	SSE	Long-Polling
Tomcat	7.0.47 - 7.0.53	X	X
JBoss	7.x	X	X
WAS	8.5.5	X	X

Step 6. Supported Browsers

The following table lists the supported browsers:

Browser	Version	SSE	Long-Polling
Firefox	3.x to 8.x		X
Firefox	9.x to 12.x	X	X
Chrome	12.x and lower		X
Chrome	13.x and higher	X	X
Internet Explorer	8x to 9.x		X
Internet Explorer	10.x and above	X	X

Step 7. Remote Push

The finch-client maven artifact is needed for making a remote push. The following code snippet shows an example of how to publish a notification remotely:

```

FinchClientService pushService = Operation.getInstance().getReference(FinchClientService.class);
//create a notification object notif
pushService.pushToQueue(notif, Operation.getInstance().getContext().getCallerIdentity());

```

Currently finch uses a queue to publish notifications remotely. So the following code snippet should be added into jms.properties file:
queue.mapping.NTF.inq=NTFFIN

Add jmsComponent bean in jms-context.xml as shown in the following code snippet (If already present, skip this part). Also add the camel-context along with it as shown in the code-snippet

```

<bean id="jmsComponent" class="org.apache.camel.component.jms.JmsComponent">
  <property name="connectionFactory" ref="cachingConnectionFactory" />
  <property name="cacheLevelName" value="CACHE_AUTO" />
  <property name="concurrentConsumers" value="${jms.concurrentConsumers}" />
  <property name="maxConcurrentConsumers" value="${jms.maxConcurrentConsumers}" />
  <property name="transacted" value="true" />
  <property name="transactionManager" ref="global-tm" />
  <property name="destinationResolver" ref="LogicalQueueResolver" />
</bean>

<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <package>com.nrft.finchnf.console.utility</package>
</camelContext>

```

Step 8. Email Notification

In order to do serve email notifications, the application must set the ChannelType to EMAIL in NotificationDestination. Also configure the following code snippet into the Spring application-context:

```
<bean id="notificationProperties"
  class="org.springframework.beans.factory.config.PropertiesFactoryBean">
  <property name="properties">
    <props>
      <prop key="emailUri">smtp://example.com</prop>
      <prop key="from">abc@xyz.com</prop>
    </props>
  </property>
</bean>
```

3.4.6.1.7 Exception Handling Framework

FINCH COMMON EXCEPTION HANDLING FRAMEWORK

Currently any application developer has to catch any Checked Exception in his code and then do the necessary handling. This leads to code duplication as the same code can be potentially repeated in multiple places.

In order to remove this finch provides a new exception handling framework. Any application developer can specify a common Handler for a exception type or types, and inside a catch block fire a event. All the handlers for the declared exception type will be called.

To use the exception handling framework following configuration is to be done in application-context.xml file

- 1) provide com.nrft.fin.ch.inf.exceptions.service package name in the component scan path.
- 2) add following bean declaration

Bean Declaration

```
<bean id="exceptionProperties"
  class="org.springframework.beans.factory.config.PropertiesFactoryBean">
  <property name="properties">
    <props>
      <prop key="scanEagerly">true</prop>
    </props>
  </property>
</bean>
```

In order to declare a class as a legitimate exception handler the marker annotation HandlesExceptions.java. The signature of the exception is given below.

HandlesException

```
/**  
 * Marker for types containing Exception Handler methods.  
 */  
@Retention(RetentionPolicy.RUNTIME)  
@Target(ElementType.TYPE)  
@Documented  
public @interface HandlesExceptions {  
}
```

The method that will be responsible for handling the exceptions , should be annotated with Handles annotation. The signature of the annotation is given below :-

```
/**  
 * Marker annotation for a method to be considered an Exception Handler.  
 */  
@Retention(RetentionPolicy.RUNTIME)  
@Target({ElementType.METHOD, ElementType.PARAMETER, ElementType.TYPE})  
@Documented  
public @interface Handles {  
    /**  
     * Direction of the cause chain traversal to listen.  
     */  
    public abstract TraversalMode during() default TraversalMode.DEPTH_FIRST;  
    /**  
     * Precedence relative to handlers for the same type  
     */  
    public abstract int precedence() default Precedence.DEFAULT;  
    /**  
     * Default Exception which the handler will be called for.  
     */  
    public abstract Class<? extends Throwable> handles () default Exception.class;  
}
```

If a exception is published to the finch framework, then finch will traverse the exception chain and fire of an event for each exception type. So in the following code the handler for NullPointerException and RunTimeException, both will be called. But by default the NullPointerException handler will be called first and then the handler for RunTimeException will be called. If an application developer needs the exceptions to be published in reverse order then they can pass the traversal mode as a parameter in an overloaded version. Any handler that has been specified for DEPTH_FIRST(bottom up) traversal will not be called if the TraversalMode is BREAD_FIRST (top down). The priority for the handlers can be set with precedence property in the annotation Handles. By default all annotation has the same precedence but if a handler has a precedence of 50 and one of -50, then the first one will be called before the second one is called (provided they both subscribe to the same type of Exception).:-

```
NullPointerException exception = new NullPointerException();  
RuntimeException exception2 = new RuntimeException(exception);  
exceptionRegistryService.registerHandler(new DefaultHandler3());
```

An example handler should like below :-

```

public class DefaultHandler2 implements BaseHandler<RuntimeException> {
    private static boolean handlerCalled = false;

    @Override
    @Handles(handles=RuntimeException.class)
    public void handle(RuntimeException t, CaughtException<RuntimeException> caughtException) {
        handlerCalled = true;
    }
    public static boolean isHandlerCalled() {
        return handlerCalled;
    }
}

```

3.4.6.1.8 Add a new module

1. Introduction

1.1. Summary

finch framework built with multiple modules like finch-core, finch-batch, finch-infra, finch-dbd etc. Application built on top of finch can create multiple module as per their needs. Each module in any application serves a specific purpose. If for a requirement there does not exists any appropriate module then application need to create a new module.

1.2. Scope

This document describe how to create a new module - what the nomenclature of new module will be, folder structure, how configuration files are kept, etc.

2. Pre-requisites

To create a new module, user must have knowledge of Maven, Java and finch. This document assume that application has already being created using finch framework and application now wants to add a new module.

3. Sequence and Steps

3.1. Naming convention of modules

Naming of module is important since it helps us to understand the purpose of the module and at the same time identify it quickly. Confusion should not arise which feature this module supports by just looking at the name. Module name should be short possibly 3 or 4 alphabets and should not be alpha numeric.

3.2. Directory Structure

Directory structure of each module should follow similar pattern. It is up to application to define directory structure.

3.3. Steps

Please see [this link](#) to create a module. Please skip step 1 (create blank-app using finch), as this document expects that, user is adding new module into existing project.

3.4. Table

3.4.1. INF_COMPONENT

This table lists all components. A new records needs to be added here. Example for TRD module has been shown below.

	COMPONENT_ID	COMPONENT_NAME	COMPONENT_DESCRIPTION	APP_REGI_DATE	APP_UPD_DATE	APP_STATUS
1	TRD	TRD	Trade	03-01-14	03-01-14	03-01-14

4. Document History

Version	Date	Description of changes	Author
1.0	12-02-2013	Initial Document	Prasun Pal

3.4.6.1.9 Configure an Online/Batch Report

Introduction

Online reports help user to export the query result in different report formats like PDF,XLS,XLSX,CVS etc. This page explains the steps required to configure online reports.

Steps to create Online Report using finch

finch provides default templates for Report (xls and pdf). These default templates include title, page-header and page-footer templates for report. User can override these finch provided default templates and configure them specific to the application.

Followings are path default template files:

- report¥templates¥common¥title-pdf.jrxml
- report¥templates¥common¥header-pdf.jrxml

- report¥templates¥common¥footer-pdf.jrxml

Similar files for excel like report¥templates¥common¥title-pdf.jrxml also provides some styles files in report¥templates¥styles¥header-style.jrtx etc.

User can use finch provided template for there report or user also can create there own report template for there report.

Override finch Common Template

The following steps need to follow to override finch provided report template:

1. create title,header and footer jrxml any where and with any name in your project directory and include these report as a sub-report into your main report in appropriate position
2. compile and create corresponding.jasper file
3. Provide the following parameters
 - a. TITLE
 - b. FOOTER
 - c. COMMON_SUBREPORT_TITLE_PATH
 - d. COMMON_SUBREPORT_HEADER_PATH
 - e. COMMON_SUBREPORT_FOOTER_PATH
 - f. QUERY_SUBREPORT_PATH
4. There is default value provided for the above parameters from finch user can use those values. There is separate methods in finch for pdf and xls. The example code for pdf has been displayed in the following code snippet.

```

protected String getCommonTitleSubreportTemplatePathForPdf() {
    return "report/templates/common/title-pdf";
}

```

5. If user wants to create there own file (other than finch provided file), user need to override the above method of AbstractQueryController.java from finch.

```

e.g.
@Override
protected String getCommonTitleSubreportTemplatePathForPdf() {
    return "report/templates/common/title";
}

```

6. The last part of the relative path is the file name(in the above example 'report/templates/common/title' "title" is the file name used by the application).
7. If the user wants to use the same file name as provided by finch but there own design, user first need to exclude the finch provided file.

```

<!-- Exclude Finch Report Template File if application using the same file name as finch -->
<exclude>**/WEB-INF/classes/report/templates/common/*.jrxml</exclude>
<exclude>**/WEB-INF/classes/report/templates/common/*.jasper</exclude>

```

If user create different file name then no need to exclude finch provided jrxml files

8. User can also use the style files provided by finch to design there template.

```

<jasperTemplate>
    <style name="headerStyle" mode="Opaque" forecolor="#000000" backcolor="#CCCCCC" fill="Solid"
fontName="ARIALUNI" fontSize="14">
        <box topPadding="1" leftPadding="1" bottomPadding="1" rightPadding="1"/>
        <paragraph firstLineIndent="1" leftIndent="1" rightIndent="1" spacingBefore="1" spacingAfter="1"/>
    </style>
</jasperTemplate>

```

9. User also can override finch provided style files (below is the example how to override style files) and can use there own style files into there own template.

```

<jasperTemplate>
    <style name="headerStyle" mode="Opaque" forecolor="#000000" backcolor="#CCCCCC" fill="Solid"
fontName="ARIALUNI" fontSize="14">
        <box topPadding="1" leftPadding="1" bottomPadding="1" rightPadding="1"/>
        <paragraph firstLineIndent="1" leftIndent="1" rightIndent="1" spacingBefore="1" spacingAfter="1"/>
    </style>
</jasperTemplate>

```

10. finch provides the default TITLE name of the report user always override the name and can display there own name in there report.similar for FOOTER.

```

@Override
protected String getReportTitle() {
    return messageSource.getMessage("trade.label.trade.reporttitle", null, I18NUtils.getLocale());
}

```

After creating report template, user need to call the report from jspx and include or override some methods into there query controller using following steps.

- Create the jspx file which contains the two buttons for pdf and xlsx. For example, refer to the following code snippet used in jspx file:

```

var grid_result_settings = {
enableToolbar:true,
consolidateActionFlag:true,
buttons:{
print:false,
xls:true,
pdf:true,
columnPicker:true,
save:true
},
pagingInfo:{
isNext : isNext,
url: 'trd/query/count.json'
},
urls:{
nextPage : '/trd/query/result.json?fetch=next',
prevPage : '/trd/query/result.json?fetch=previous',
pdfReport: '/trd/query/report/trade.pdf?filetype=pdf',
xlsReport: '/trd/query/report/trade.xlsx?filetype=xlsx'
}
};

```

- Configure webmvc-config.xml, as shown below:

```

<bean class="org.springframework.web.servlet.view.XmlViewResolver">
<property name="order" value="2" />
<property name="location" value="/WEB-INF/report/jasper-views.xml"/>
</bean>

```

- Create corresponding jasper-views.xml and write the code as shown below.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">

    <!-- <bean id="trdquery" class="com.nrft.sample.trd.report.MultiFormatReportView"> -->
    <bean id="trdquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="trdqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="trdquerycsv" class="com.nrft.fin.ch.inf.web.report.JasperReportsCsvView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="accesslogquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/inf/AccessLogSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="accesslogqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/AccessLogSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="empquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/inf/UserSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="empqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/UserSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="holidayreportxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/Holiday-Calendar-Report-Excel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>
</beans>
```

- Create controller which extends finch AbstractQueryController now override getReportParameters() method of AbstractQueryController class, as shown below.

```

@Override
protected Map<String, Object> getReportParameters(
    @ModelAttribute("commandForm") TrdQueryCommandForm cmdForm)
throws FinchException {
Map<String, Object> params = Maps.newHashMap();
Map<String, String> criteriaMap = Maps.newLinkedHashMap();

addCriteria(criteriaMap, "Trade Type", cmdForm.getTradeType());
addCriteria(criteriaMap, "Trade Reference No", cmdForm.getReferenceNo());
addCriteria(criteriaMap, "Account No", cmdForm.getAccountNo());
addCriteria(criteriaMap, "Inventory Account No", cmdForm.getInventoryAccountNo());
addCriteria(criteriaMap, "Security", cmdForm.getSecurityCode());
addCriteria(criteriaMap, "Trade Currency", cmdForm.getTradeCcy());
addCriteria(criteriaMap, "Settlement Currency", cmdForm.getSettlementCcy());
addCriteria(criteriaMap, "Trade Date From", cmdForm.getTradeDateFrom());
addCriteria(criteriaMap, "Trade Date To", cmdForm.getTradeDateTo());
addCriteria(criteriaMap, "Value Date From", cmdForm.getValueDateFrom());
addCriteria(criteriaMap, "Value Date To", cmdForm.getValueDateTo());
addCriteria(criteriaMap, "Instrument Type", cmdForm.getInstrumentType());
addCriteria(criteriaMap, "Execution Market", cmdForm.getExecutionMarket());

addCriteria(criteriaMap, "Buy/Sell Flag", cmdForm.getBuySellFlag());
addCriteria(criteriaMap, "Principal/Agent Flag", cmdForm.getPrincipalAgentFlag());
addCriteria(criteriaMap, "Entry Date Form", cmdForm.getCreationDateFrom());
addCriteria(criteriaMap, "Entry Date To", cmdForm.getCreationDateTo());
addCriteria(criteriaMap, "Last Entry Date From", cmdForm.getUpdateDateFrom());
addCriteria(criteriaMap, "Last Entry Date To", cmdForm.getUpdateDateTo());
addCriteria(criteriaMap, "External Reference No", cmdForm.getExternalReferenceNo());
addCriteria(criteriaMap, "Cancel Reference No", cmdForm.getCancelReferenceNo());
addCriteria(criteriaMap, "Account Balance Type", cmdForm.getAccountBalanceType());
addCriteria(criteriaMap, "Trade Status", cmdForm.getStatus());
addCriteria(criteriaMap, "Data Source", cmdForm.getDataSource());

params.put("criteriaMap", criteriaMap);
return params;
}

```

- Create another method addCriteria into the controller to put the value.

```

private void addCriteria(Map<String, String> map, String key, Object value) {
    if((value != null) && (!Strings.isNullOrEmpty(value.toString()))) {
        map.put(key, value.toString());
    }
}

```

3.4.6.1.10 Enterprise specific db schema management

1. Introduction

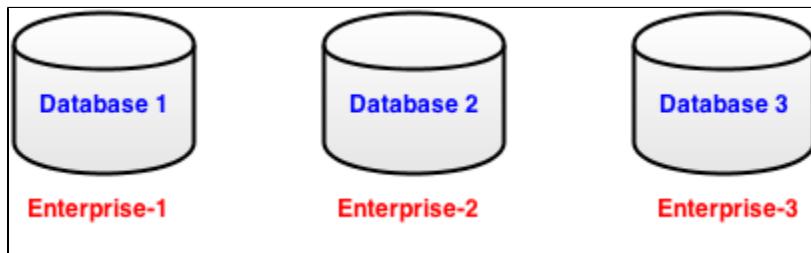
finch application is designed to develop multi-tenant enterprise applications. Application deployed as SAAS model can be developed using finch. It provides the features of configuring tenant or enterprise specific configuration, including database schema etc.

2. Approaches for Database Schema

Multi-tenancy in finch can be achieved through the following approaches for Database Schema.

2.1. Separate Database for Each Enterprise

In this approach, a different database is maintained for each enterprise. The enterprise will only refer the database to which it has been mapped to. The following image displays the graphical representation of this approach.



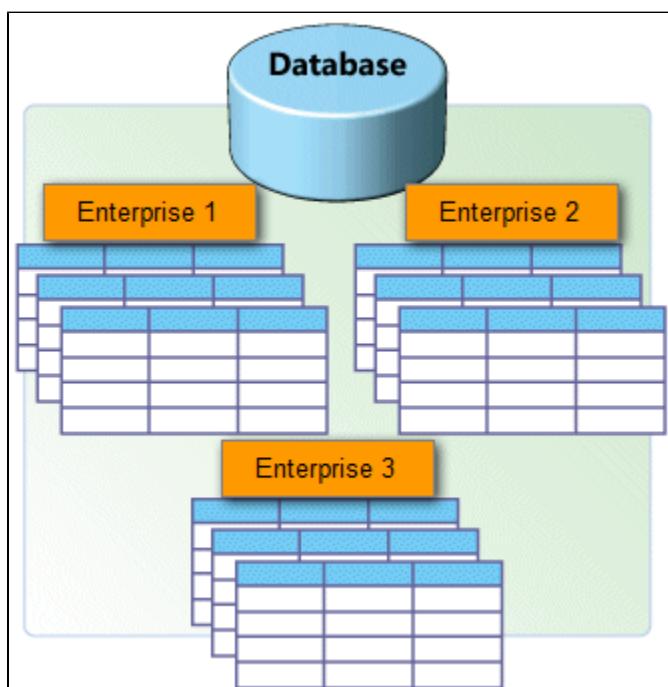
For setting up multiple enterprise following steps needs to be done-

1. Enterprise specific data set up. For details see [Guideline to Add New Enterprise](#).
2. Enterprise specific application-context. Application context name should be <enterprise name in lowercase>-application-context.xml. For example enterprise name FIN, application context name should be fin-application-context.xml. For a enterprise controller, service ,repository should be in one context and for another enterprise they should be in another context.
3. In enterprise specific application-context file, mention that enterprise specific datasource-context file name. Here add transaction manger specific to that enterprise.Global transaction manager should remain same in all the datasource-context file.
4. In enterprise specific datasource-context file mention enterprise specific datasource-properties file name.
5. In enterprise specific datasource-properties mention which database and schema to connect.

Do not autowire any service in controller level or store it in member variable. Always take a service dynamically.

2.2. Shared Database and Separate Schema for Each Enterprise

In this approach, multiple enterprise information is stored in the same database. In this database, separate schema is created for each and individual enterprise. Each enterprise has their own set of tables. Tables of a particular enterprise are grouped into a schema, which is created particularly for that enterprise.



For setting up multiple enterprise following steps needs to be done-

- Enterprise specific data set up. For details see [Guideline to Add New Enterprise](#).
- Enterprise specific application-context. Application context name should be <enterprise name in lowercase>-application-context.xml. For example enterprise name FIN, application context name should be fin-application-context.xml. For a enterprise controller, service ,repository should be in one context and for another enterprise they should be in another context.
- In enterprise specific application-context file, mention that enterprise specific datasource-context file name. Here add transaction manger specific to that enterprise.Global transaction manager should remain same in all the datasource-context file.
- In enterprise specific datasource-context file mention enterprise specific datasource-properties file name.
- In enterprise specific datasource-properties mention which database and schema to connect.

Do not autowire any service in controller level or store it in member variable. Always take a service dynamically.

2.3. Shared Database and Shared Schema for Each Enterprise

In this approach, same database, same schema and same set of tables are used to host data of multiple enterprises. The Enterprise ID column of every table signifies that the particular row contains record of which enterprise. This column value is mandatory for each record and contains the Enterprise ID value of any existing enterprise. The following tables display how the data of multiple enterprises are stored in a single table and how the Enterprise ID column values (ENT1, ENT2, ENT3) signify the different enterprises.

Enterprise Id	Instrument Id	Instrument Name
ENT1	INS1	Instrument 1
ENT2	INS2	Instrument 2
ENT3	INS3	Instrument 3

Enterprise Id	Account Id	Account name
ENT1	ACC1	Account 1
ENT2	ACC2	Account 2
ENT3	ACC3	Account 3

3. Document History

Version	Date	Description of changes	Author
1.0	12-02-2015	Initial document	Prasun Pal

3.4.6.1.11 Enterprise specific implementation/customization

1. Introduction

This section describes the guideline of creating a new enterprise.

Important Note:

The steps mentioned below are written from finch framework perspective. There may be few more application specific steps required and those steps should be provided by the corresponding application.

Service Enterprise

A service enterprise is an enterprise that provides an application as service (SAAS). A particular enterprise, among all enterprises, is

marked as service enterprise. Adding a new enterprise in an application is the responsibility of service enterprise. Only user or employee belonging to the service enterprise is allowed to view configuration data such as actions, branch, application roles for all the existing enterprises. User or employee belonging to other than service enterprise is allowed to access data belonging to his/her enterprise only.

2. Adding New Enterprise

Steps for adding a new enterprise have been mentioned below.

Step-1 : Add new Enterprise record

Insert a new record for enterprise in INF_ENTERPRISE table. This is done using SQL script. finch does not provide any user interface for this step. For example:

ENTERPRISE_ID	MNEMONIC	SHORT_NAME	OFFICIAL_NAME	MAX_PASSWD_LENGTH	MIN_PASSWD_LENGTH	PASSWD_CHANGE_FREQ	MAX_INACTIVE_PERIOD_IN_SEC
NRI	NRI	NRI FINTECH	NRI ENTERPRISE	20	4		600

Step-2 : Add Branches for new Enterprise

An enterprise can have one or more branch offices. In finch, branch offices are stored in INF_BRANCH table. To add branch for an enterprise, insert records in INF_BRANCH table. This is done using SQL script. finch does not provide any user interface for this step. For example:

BRANCH_PK	BRANCH_ID	ENTERPRISE_ID	MNEMONIC	SHORT_NAME	OFFICIAL_NAME	TIME_ZONE	STATUS
1	NRIBR	NRI	NRIBR	NRI Branch	NRI ENTERPRISE BRANCH		NORMAL

Step-3 : Define Set of Actions for new Enterprise

An application provides a set of features. In finch, features are accessed and controlled through Action (INF_ACTION). Examples for Actions are:

- Trade Query
- Instrument Amend

An enterprise has the access to a sub-set of all the actions provided by the application. While adding a new enterprise, a set of actions permitted for the enterprise to be defined. If an action is not defined for an enterprise, user belonging to that enterprise will not be able to access the action.

This step is usually performed by an employee of service enterprise. This can be done by inserting records for each combination of enterprise and action in INF_ACTION_ENTERPRISE_PCPT. For example:

ACTION_ENTERPRISE_PCPT_PK	ENTERPRISE_ID	ACTION_ID	STATUS
1	1	TRDTEN	NORMAL
2	1	TRDTQR	NORMAL

Step-4 : Define Administration Role and User for new Enterprise

While adding a new enterprise, an administration role and administration user should be created. This step should be performed by an employee of the service enterprise. The administration role must have the following permissions:

- Sets up actions for the role. These actions must be a subset of actions defined for the enterprise.
- Create and Amend application role for the enterprise to which it belongs to.
- Create and Amend employee for the enterprise to which it belongs to.

This administration user can later define other roles and users for the enterprise.

- To define administration role:

- Insert record in INF_APPLICATION_ROLE table for enterprise specific application role. In finch application roles are defined at branch level. For example:

APPL_ROLE_PK	APPLICATION_ROLE_NAME	BRANCH_PK	ACCOUNT_ACCESS_RESTR_FLAG	REMARKS	STATUS
1	ROLE1	1	Y		NORMAL

- For administration application role, define the set of actions required for administrative purpose. This can be done by inserting record in the INF_ACTION_ROLE_PARTICIPANT table. For example:

ACTION_ROLE_PCPT_PK	APPL_ROLE_PK	ACTION_ID
1	1	TRDTEN
2	1	TRDTQR

An administrative user can be created from the Create Employee user interface. Refer to Employee Creation for creating employee and Application Role for authorization.

Note:

The above records can be inserted through the application UI later.

Step-5 : Insert Application date if application supports Application Date functionality(Optional)

If application supports application date functionality then user need to insert data in INF_APPLICATION_DATE as follows.

ENTERPRISE_ID	COMPONENT_ID	APP_MODE	DESCRIPTION	APPLICATION_DATE
FIN	INF	OL	FIN_INF_OL	03-MAR-14 11:40:46
FIN	INF	BT	FIN_INF_OL	03-MAR-14 11:40:46

Step-6 : Define Application Role per Employee(Optional)

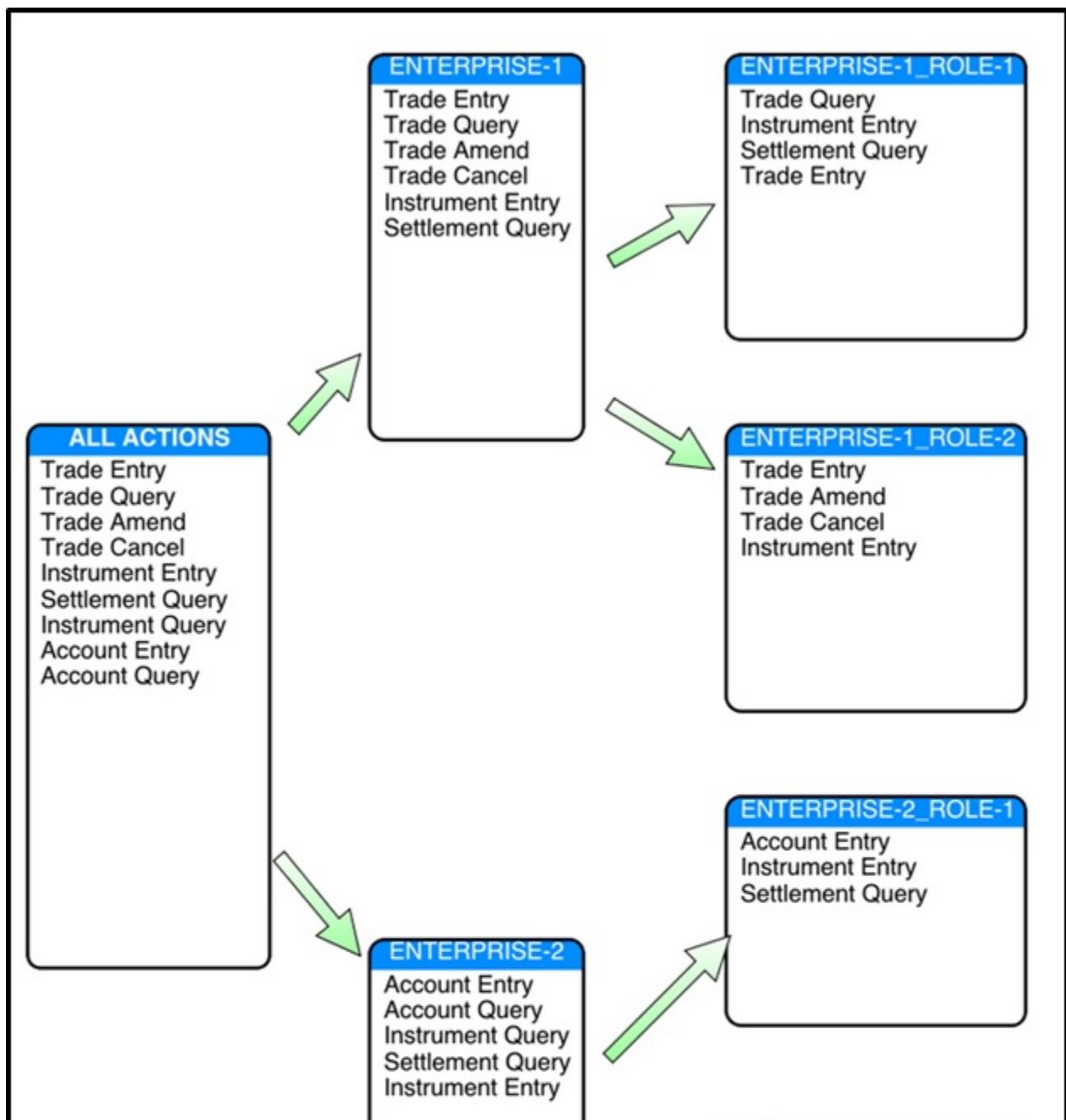
User can define application role per employee by defining the role in INF_EMP_APPLN_ROLE_PARTICIPANT table as follows

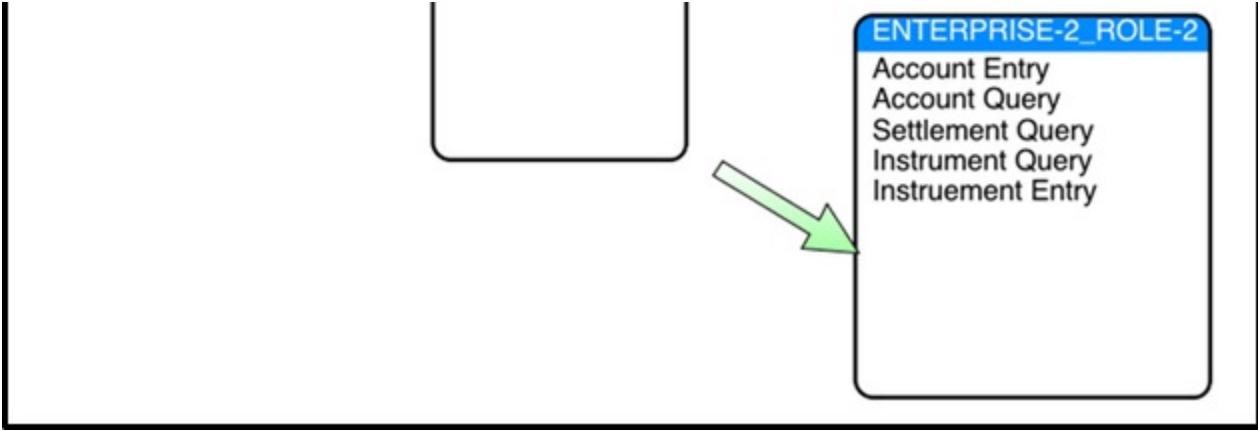
APPL_ROLE_PK	EMPLOYEE_PK
FIN	INF
FIN	INF

Considerations

- A user from ENTERPRISE-1 with ROLE-1 cannot access Account Entry, Trade Cancel but can access Trade Query.
- A user from ENTERPRISE-1 with ROLE-2 cannot access Trade Query, Account Query but can access Trade Entry.
- A user from ENTERPRISE-2 with ROLE-1 cannot access Instrument Query, Trade Entry but can access Account Entry.

The following image displays the illustrations:





3. Document History

Version	Date	Description of Changes	Author
1.0	12-02-2015	Initial document	Prasun Pal

3.4.6.1.12 Handle of Concurrent operation

1. Introduction

A database record can be updated by more than one process simultaneously which may lead to data inconsistency due to concurrent access. In order to maintain consistency across all users, concurrent operations has to be dealt with very carefully. Locking mechanisms are to be used to ensure transactional integrity and database consistency. The scope of this document is to provide developers with information on how to handle concurrent database updates.

2. Design

finch uses JPA for database operations. JPA provides option to define a column for version controlling. Using this attribute (version attribute) JPA controls concurrency. Every records in database is represented by an entity. While updating an entity, if version value at entity differ from version value at database for that entity, then someone else has already updated that entity, hence the transaction involving that dirty entity is rollback. This ensure that concurrency related problems will not exists.

3. Implementation

In finch, version attribute has been put in the EntityBaseWithVersion class with the help of @Version annotation. Database column that has been used for version controlling in CC_CHECK. Every table in finch has this column. Any concrete entity class extends from EntityBaseWithVersion inherits version attribute required for concurrency. This attribute helps in concurrency control. Following is the code snippet of EntityBaseWithVersion class

```
public class EntityBaseWithVersion extends EntityBase {

    private Integer concurrencyCheck;
    @Version
    @Column(name = "CC_CHECK")
    public Integer getConcurrencyCheck() {
        return concurrencyCheck;
    }
    public void setConcurrencyCheck(Integer concurrencyCheck) {
        this.concurrencyCheck = concurrencyCheck;
    }
}
```

Any concrete class that will extends from this class will inherit version attribute. Following is an example of such implementation

```
public class ApplicationDate extends EntityBaseWithVersion implements Serializable {  
    ...  
}
```

4. Document History

Version	Date	Description of changes	Author
0.1	26-02-2015	Initial Version	Prasun Pal

3.4.6.1.13 Manage Transaction

1. Introduction

1.1. Summary

In finch, transaction can span across UI as well as command line interfaces like batch and services. Although the generic transaction management remains same. finch does not provide any support for transaction management out-of-the-box and it uses the transaction management of spring framework.

1.2. Scope

This document transaction management implemented by finch framework. finch uses transaction management functionality of spring framework. To know more about transaction management of spring please refer to reference document.

1.3. Prerequisites

Please read the document about transaction management

[Transaction Management](#)

2. Implementation details

Prerequisites section has already discuss what finch do to implement transaction management. Method in service layer can specify more configuration related to transaction settings. For example, If any method wants to start a new transaction when the method will be called then it can specify the same in @Transactional annotation with the help of "propagation" properties with value "Propagation.REQUIRES_NEW". Following is one such example

```
@Transactional(value = "global-tm", readOnly = true, propagation=Propagation.REQUIRES_NEW)  
public void methodName(...){  
    ...  
    ...  
    ...  
}
```

If a method in service layer throws exception, then spring will rollback the transaction and developer needs to catch that exception in controller to display proper message to user. If method does not throw any exception then transaction will be committed. More details about transaction management can be found in the reference section.

3. Reference

<http://docs.spring.io/spring-framework/docs/3.2.0.RC2/reference/htmlsingle/#spring-data-tier>

4. Document History

Version	Date	Description of Changes	Author
1.0	17-02-2015	Initial Document	Prasun Pal

3.4.6.1.14 Multilingual i18n Support

1. Introduction

1.1. Summary

finch provides i18n support for the following :

- The menu, screen and section names
- User interface display labels and messages
- The client side validation and information messages
- Error Messages
- The Report

See the [i18n](#) document for more details on both client side and server side i18n Implementation.

1.2. Scope

This document discuss how to implement localization support for application

2. Sequence of steps

In finch, standard Java Locale has been used for i18n implementation. See <http://docs.oracle.com/javase/6/docs/api/java/util/Locale.html> for complete details.

2.1. i18n implementation for menu name

finch provides i18n support for menu as an application can have menu in different language. For i18n support finch used localized properties file to resolve menu name.

e.g If application has support for English and Japanese language then, menu name is to be configured in following properties file

- a. menu_en.properties file - properties file for displaying menu in English
- b. menu_ja.properties file - properties file for displayed menu in Japanese

Consider the INF_UI_MENU table

MENU_PK	SCREEN_KEY	MENU_NAME	MENU_DESCRIPTION
1	(null)	Main Menu	Main Menu
2	(null)	Trade	Parent of Trade
3	1	Entry	Trade Entry
4	2	Query	Trade Query

For i18n support there should an entry in both properties files with the following convention

MENU_ID.name = <menu display name>

e.g.

menu_en.properties file

TRD.name = Trade

TRDEN.name = Entry

TRDQR.name = Query

menu_ja.properties file

TRD.name = ¥u53D6¥u5F1

TRDEN.name = ¥u767B¥u9332

TRDQR.name = ¥u30AF¥u30A8¥u30EA¥u30FC

Based on the application language setup menu will be displayed either in English or Japanese.

finch provides i18n support for screen name. For i18n support finch used localized properties file to resolve screen name.

e.g If application has support for English and Japanese language then, screen name is to be configured in following properties file

- a. screen_en.properties file - properties file for displaying screen name in English
- b. screen_ja.properties file - properties file for displaying screen name in Japanese

Consider the INF_UI_SCREEN table

SCREEN_KEY	SCREEN_ID	SCREEN_NAME	SHORT_NAME	DESCRIPTION	NEXT_SCREEN_PK	VERSION_NUMBER	COMPONENT_ID
10001	TRDEN	Trade Entry	Trade Entry	Trade Entry	10012	1	TRD
10012	TENUC	Trade Entry User Confirmation	Trade Entry UC	Trade Entry User Confirmation	10013	1	TRD
10013	TENSC	Trade Entry System Confirm	Trade Entry SC	Trade Entry SC Confirmation		1	TRD

For i18n support there should an entry in both properties files with the following convention

SCREEN_ID.name = <screen name>

e.g.

screen_en.properties file

TRDEN.name=Trade Entry

TENUC.name=Trade Entry User Confirmation

TENSC.name=Trade Entry System Confirm

screen_ja.properties file

TRDEN.name=¥u8CBF¥u6613¥u30A8¥u30F3¥u30C8¥u30EA¥u30FC

TENUC.name=¥u8CBF¥u6613¥u30A8¥u30F3¥u30C8¥u30EA¥u30FC¥u30E6¥u30FC¥u30B6¥u30FC¥u306E¥u78BA¥u8A8D

TENSC.name=¥u8CBF¥u6613¥u30A8¥u30F3¥u30C8¥u30EA¥u30FC¥u30B7¥u30B9¥u30C6¥u30E0¥u306F¥u78BA¥u8A8D

Based on the application language setup screen name will be displayed either in English or Japanese.

finch use List of LabelValueBean to display query sort fields. Application has to provide the the List of LabelValueBean for sort fields.

Following code snippet populated the sort field for sort fields for Employee Query screen.

sort fields Expand

```
protected void doInit(EmployeeQueryCommandForm form, FinchModelMap modelMap)      source
    throws FinchException {
    ...
    List<LabelValueBean> sortFieldList = new LinkedList<LabelValueBean>();
    sortFieldList.add(new LabelValueBean(messageSource.getMessage("inf.employee.label.userid", null, locale),
                                         "userId"));
    sortFieldList.add(new LabelValueBean(messageSource.getMessage("inf.employee.label.firstname", null,
                                                               locale), "firstName"));
    sortFieldList.add(new LabelValueBean(messageSource.getMessage("inf.employee.label.startdate", null,
                                                               locale), "startDate"));
    initCommandForm.setSortFieldList(sortFieldList);
    ...
}
```

Configuration for i18n support in sort field

The configuration for i18n support in sort field is same as configuration for Server side info message in section 3.3.3
Server-side Info Messages

If developer wants some constraints to be shown when user preference of language is set a particular language then add language code of that language in LANGUAGE_CODE column. Give valid UI_DISPLAY_VALUE as this value will be shown in UI.

INF_CONSTRAINT_VALUES

CONSTRAINT_PK	CONSTRAINT_VALUE	DEFAULT_MSG	DEFAULT_UI	DISPL
1001	FI			1
1001	EQ			2
1001	FI			1
1001	EQ			2

Whenever language preference is set as English, then EXTERNAL and INTERNAL will be displayed and when language preference is set to Japanese, then 定の収入, エクイティ be displayed.

i18n implementation for user interface display labels and messages

We use Spring MVC for finch web application. Spirng MVC framework supports i18n. In finch application, i18n related configuration is done in webmvc-config.xml file. This file is kept under the following folder - /finch-dist/src/main/web/WEB-INF/config

The following entry is done in the above mentioned file for configuring the message properties file.

```
webmvc-config.xml

<bean class="org.springframework.context.support.ReloadableResourceBundleMessageSource"
id="messageSource"
    p:fallbackToSystemLocale="false"
    p:defaultEncoding="UTF-8">
    <property name="basenames">
        <list>
            <value>WEB-INF/classes/i18n/app</value>
            <value>WEB-INF/classes/i18n/messages</value>
            <value>WEB-INF/classes/i18n/ref-messages</value>
        </list>
    </property>
</bean>
```

Description of module specific message file

In a module specific message file, name of the labels, info messages, etc. which are displayed on screen, are mentioned as text values. These text values are displayed in user preferred language [here, English / Japanese].

- Sample - module specific message file for default language

```
ref-messages.properties

...
ref.label.application.role.name=Application Role Name
ref.label.office.id=Office ID
ref.label.account.restriction=A/C Restriction
ref.label.status=Status
ref.label.datasource=Data Source
...
```

[Expand](#)

[source](#)

- Sample - module specific message file for Japanese language

```
ref-messages_ja.properties

...
ref.label.application.role.name=¥u30A2¥u30D7¥u30EA¥u30B1¥u30FC¥u30B7¥u30E7¥u30F3¥u30ED¥u30FC¥u30EB¥u540D
ref.label.office.id=¥u30AA¥u30D5¥u30A3¥u30B9¥u306EID
ref.label.account.restriction=A¥C ¥u5236¥u9650
ref.label.status=¥u30B9¥u30C6¥u30FC¥u30BF¥u30B9
ref.label.datasource=¥u30C7¥u30FC¥u30BF¥u30BD¥u30FC¥u30B9
...
```

[Expand](#)

[source](#)

Naming Convention

- English:<module_name>-messages_en.properties
- Japanese: <module_name>-messages_ja.properties
- Default : <module_name>-messages.properties

Source Folder Location

The module specific message file should be in '...<module_name>/src/main/web/WEB-INF/i18n'

2.6. Implementation of client side validation / error message

For client-side error message, error messages are defined in '<fileName>-<module_name>-i18n_en.js' file. The 'en' used in name signifies the short form of the CHARSET_CODE used, i.e. for English it is en. This file is placed under module root package, i.e. .../devel/<module_name>/src/main/web/scripts/<module_name>/...

Example

For default language, REF module's 'sample-ref-i18n_en.js' is placed in '.../devel/sample-ref/src/main/web/scripts/ref/sample-ref-i18n_en.js'.

sample-ref-i18n_en.js » Expand

SAMPLE\$REF\$i18n = {
 dashboard:{
 no_record_found : 'No record selected for rejection.',
 no_record_for_rejection : 'Rejected record(s) cannot be rejected again.'
 },
 required_fields_empty_alert : 'Some modified data has not yet been saved, Do you want to continue?',
 authorization: {
 common : {
 empty_record : 'No record selected for authorization',
 rejected_record : 'Rejected record(s) cannot be authorized.'
 }
 },
 ...
};

[source](#)

If we need error message in any other language then we need to define corresponding *.js file in the same package. For example, Japanese error message file for REF is defined in 'sample-ref-i18n_ja.js' file in the '/devel/sample-ref/src/main/web/scripts/ref/' package.

sample-ref-i18n_ja.js

```
SAMPLE$REF$i18n = {
  dashboard: {
    no_record_found : 'No record selected for rejection.',
    no_record_for_rejection : 'Rejected record(s) cannot be rejected again.'
  },
  required_fields_empty_alert : 'Some modified data has not yet been saved, Do you want to continue?',
  authorization: {
    common : {
      empty_record : 'No record selected for authorization',
      rejected_record : 'Rejected record(s) cannot be authorized.'
    }
  },
  ...
};
```

These javascript resource files are loaded using the 'loadLocalizedScript' function. Based on the current language it loads the corresponding javascript resource files asynchronously.

The loadLocalizedScript function definition is shown below:

loadLocalizedScript

[Expand](#)

```
var loadLocalizedScript = function(_scripts, options) {
  // fixed scripts
  jQuery.each(_scripts, function(index, script) {
    var dotJs = script.path.indexOf('.js');
    if (dotJs != -1)
      script.path = script.path.substring(0, dotJs);
    if (FINCH.context.locale != 'en')
      script.fallbackPath = script.path + '_en.js';
    script.path = script.path + '_' + FINCH.context.locale + '.js';
    script.error = function() {
      scripts[script.path].loaded = false;
      script.loaded = true;
      if (script.fallbackPath)
        loadScript([{
          path : script.fallbackPath
        }]);
    };
    loadScript(_scripts, options);
  });
};
```

[source](#)

The usage of loadLocalizedScript function is shown below:

```
FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/ref/sample-ref-i18n.js', async:
```

```
false}]);
```

2.7. Implementation of server side message

Application may also display some info messages to user. finch also provides support for display server side info messages. Info message can be displayed from the properties files configured in section 3.1 User interface display labels and messages.

To display message application controller must implement `MessageSourceAware`.

For example - see the following controller code snippet

EmployeeEntryController

 [Expand](#)

[source](#)

```
@Controller  
@SessionAttributes("commandForm")  
@RequestMapping(value = "/inf/employee/entry/**")  
public class EmployeeEntryController extends  
FinchAbstractEditWizardController<EmployeeEntryCommandForm> implements  
MessageSourceAware {  
...  
private MessageSource messageSource;  
@Override  
public void setMessageSource(MessageSource messageSource) {  
    this.messageSource = messageSource;  
}  
...  
}
```

Following the code snippet gets the message and set in the form when a Employee record is entered:

doConfirm() method of EmployeeEntryController

 [Expand](#)

[source](#)

```
protected void doConfirm(EmployeeEntryCommandForm form)  
throws FinchException {  
...  
String successMessage = messageSource.getMessage("inf.employee.entry.action.success", param, userLocale);  
form.setSuccessMessage(successMessage);  
...  
}
```

This message is displayed in Employee Entry System Confirmation screen

userDetails.jspx

› [Expand](#)

```
var mode = "<c:out value='${commandForm.mode}' />";  
if(mode == "CONFIRMED"){  
    var successMessage = "<c:out value='${commandForm.successMessage}' />";  
    FINCH.postNotice(FINCH.notice.type.info, successMessage, true);  
}
```

[source](#)

2.8. i18n implementation for Report

3. See also

Refer

- http://en.wikipedia.org/wiki/Internationalization_and_localization
- <http://www.irt.org/articles/js179/>
- <http://www.journaldev.com/2610/spring-mvc-internationalization-i18n-and-localization-l10n-example>
- <http://docs.oracle.com/javase/6/docs/api/java/util/Locale.html>

4. Document History

Version	Date	Description of Changes	Author
1.0	18-02-2015	Initial document	Prasun Pal

3.4.6.1.15 New Batch Process

1. Introduction

This page describes the steps to add a new batch process into finch.

MUST SEE

[Batch Process](#)

[Batch UI Use Case](#)

1.1 Scope

Batch processing is the execution of a series of programs or jobs without manual intervention. finch has various batches throughout the modules for various jobs.

After reading this document, a developer will be able to add a new batch process into finch.

2.Pre-requisites

- Transaction Handling : To know more about Transaction Handling in finch, read [Transaction Handling](#).
- Logging : To know the details of Logging framework in finch, see [Logging](#).
- Java Exception Handling : To better understand Java Exception Handling in finch, go to [Java Exception Management](#).

3. Sequence of Steps

To add a new batch process into finch, the following steps need to be followed

1. The class representing the batch-job must extend com.nrft.fin.ch.inf.console.batch.ConsoleMetaData and must be annotated with com.nrft.fin.ch.inf.batch.metadata.Batch to avail the finch support for batch-operations.

com.nrft.fin.ch.inf.batch.metadata.Batch annotations has four properties

1. id - Should be same with the batch id
2. authenticate- It accepts true/false.If true then authentication enabled else false means authentication disabled
3. register - It accepts true/false.If true the it register with ServiceController.If false then no need to register with ServiceController
4. componentId - It takes application component Id.

2. finch uses args4j library for command line parameters. Add batch specific command line parameters parsing using @Option annotation of args4j library in the batch class. For example,

```
@Option(name="-q")
private String queueName
```

Please note that following option names are used by finch console classes and Application should NOT use those option names:

- h -- Option to print help or usage message
- e -- Option for enterprise Ids

Application client can configure the showUsage() which is required to display the switches while running the batch if any required parameter is missing. finch generally display all the usage options (switches) when any required parameter is missing while running the batch.

SampleDBtoCVSBatch.java

[Expand](#)

[source](#)

```
@SuppressWarnings("unchecked")
@Override
public Optional<String> showUsage() {
    ResourceBundle rb = null;
    rb = new ExtendedBundleDecorator(CONSOLE_OPTION_USAGE_RESOURCE, I18NUtils.getLocale());
    StringBuilder showUsages = new StringBuilder();
    showUsages.append("Showing Usages\n");
    Set<Field> fields = ReflectionUtils.getAllFields(this.getClass());
    for (Field field : fields) {
        if (field.isAnnotationPresent(Option.class)) {
            Option opt = field.getAnnotation(Option.class);
            boolean required = opt.required();
            if (required) {
                showUsages.append("\t" + opt.name() + " " + field.getType().getSimpleName() + "\t"
                    + rb.getString(opt.usage()) + "\n");
            }
        }
    }
    Optional<String> usages = Optional.of(showUsages.toString());
    return usages;
}
```

3. Following needs to be annotated to use the Dropdown, checkbox etc. on the Batch UI page. For example,

```
@Option(name="-f", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown)
private String file;
```

The dropdown generated in Batch UI can be configured in various ways:

From Constraint

The dropdown value can be fetched from constraint table.The client has to configure the following:

DemoBatch.java

```
@Option(name="-f", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,userConstraintName("<YOUR_CONSTRAINT_NAME>"))
private String file;
```

From Static Value

The dropdown value can be fetched from a class that returns a map consisting of key value pairs.The client has to configure the following:

DemoBatch.java

```
@Option(name="-fn", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,isDisplayAble=true,generator=SampleGenerator.class)
private String file;
```

This is a sample code which shows the implementation for SampleGenerator.java

SampleGenerator.java

[Expand](#)

[source](#)

```
public class SampleGenerator implements DropDownDataGenerator {
    /**
     * The <code>Log</code> instance for this class.
     */
    private static final Logger log = LoggerFactory.getLogger(SampleGenerator.class);
    @Override
    public Map<String, String> getData() throws FinchException{
        Map<String, String> dummyMap = Maps.newHashMap();
        dummyMap.put("Key1","value1");
        return dummyMap;
    }
}
```

From Dynamic Table :

The dropdown value can be fetched from a class that returns a map consisting of key value pairs fetched from a custom table.The client

has to configure the following:

DemoBatch.java

```
@Option(name="-fn", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,isDisplayAble=true,dynamicGenerator=SampleDynamicGenerator.class,isDynamicDropdown=true)
private String file;
```

This is a sample code which shows the implementation for SampleDynamicGenerator.java

SampleDynamicGenerator.java

[Expand](#)

[source](#)

```
public class SampleDynamicGenerator implements DropDownDataGenerator {
    /**
     * The <code>Log</code> instance for this class.
     */
    private static final Logger log = LoggerFactory.getLogger(SampleDynamicGenerator.class);
    @Override
    public Map<String, String> getData() throws FinchException {
        TradeService tradeService = Operation.getInstance().getReference(TradeService.class);
        List<TradeDetailsView> tradeDetails = null;
        Map<String, String> dummyMap = Maps.newHashMap();
        try {
            tradeDetails = tradeService.getAllNormalTrades();
            for (TradeDetailsView trade : tradeDetails) {
                dummyMap.put(String.valueOf(trade.getTradePk()), trade.getExternalReferenceNo());
            }
        } catch (TrdException e) {
            log.error("Failed to get the trade details", e);
            throw new TrdException(e.getKey());
        }
        return dummyMap;
    }
}
```

Provide getter and setter for each instance variables defined and Batch class can override these methods.

3.4 Batch Job Exception Handling

Catch ComponentKeyedException or its subtypes or JobExecutionException exceptions or its subtypes for handle the exceptions

4. Common Configurations to Run Batch Job

4.1 Generate Batch Metadata

To generate metadata, configure the meta data processor in the package phase of build process of the application.

Generate Spring Batch metadata

```
....  
<plugin>  
  <groupId>org.codehaus.mojo</groupId>  
  <artifactId>exec-maven-plugin</artifactId>  
  <version>1.2.1</version>  
  <executions>  
    <execution>  
      <phase>package</phase>  
      <goals>  
        <goal>java</goal>  
      </goals>  
      <configuration>  
        <mainClass>com.nrft.fin.ch.inf.batch.scanner.FinchBatchMetaDataTableBuilder</mainClass>  
        <arguments>  
          <argument>${project.build.directory}/batch-meta</argument>  
        </arguments>  
      </configuration>  
    </execution>  
  </executions>  
</plugin>  
....
```

4.2 Creating a Batch Job

4.2.1 Batch Job Creation Steps

To use finch batch user need to include batch-context.xml in there application.Below is the example

batch-context.xml

[Expand](#)

[source](#)

```
<beans xmlns="http://www.springframework.org/schema/beans"  
      xmlns:p="http://www.springframework.org/schema/p"  
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
      xsi:schemaLocation="http://www.springframework.org/schema/batch  
                          http://www.springframework.org/schema/batch/spring-batch-2.2.xsd  
                          http://www.springframework.org/schema/beans  
                          http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">  
  <bean id="jobRegistry"  
        class="org.springframework.batch.core.configuration.support.MapJobRegistry" />  
  <bean  
        class="org.springframework.batch.core.configuration.support.JobRegistryBeanPostProcessor">  
    <property name="jobRegistry" ref="jobRegistry" />  
  </bean>  
  
  <bean id="executors" class="java.util.concurrent.Executors"  
        factory-method="newFixedThreadPool">  
    <constructor-arg value="10"/>  
  </bean>  
  
  <bean id="taskExecutor" class="org.springframework.scheduling.concurrent.ConcurrentTaskExecutor">  
    <constructor-arg ref="executors" />  
  </bean>
```

```

<bean id="jobLauncher"
  class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
  <property name="jobRepository" ref="jobRepository" />
  <property name="taskExecutor" ref="taskExecutor" />
</bean>
<bean id="jobRepository"
  class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean">
  <property name="dataSource" ref="GLOBAL" />
  <property name="transactionManager" ref="batchTransactionManager" />
  <property name="databaseType" value="oracle" />
  <property name="isolationLevelForCreate" value="ISOLATION_DEFAULT" />
</bean>
<bean name="processShutdownListener" class="com.nrft.finch.inf.batch.ProcessShutdownListener">
  <property name="executor" ref="taskExecutor" />
  <property name="eventBus" ref="eventBus" />
  <property name="explorer" ref="jobExplorer" />
</bean>
<job id="baseJob" abstract="true"
  xmlns="http://www.springframework.org/schema/batch">
  <listeners>
    <listener ref="processShutdownListener" />
  </listeners>
</job>
<!-- <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
  <property name="entityManagerFactory" ref="sample-em" />
  <property name="dataSource" ref="SAMPLE" />
</bean> -->
<bean id="batchMetaDataProperties"
  class="org.springframework.beans.factory.config.PropertiesFactoryBean">
  <property name="properties">
    <props>
      <prop key="filePath">classpath: META-INF/batch/*.json</prop>
    </props>
  </property>
</bean>

<bean id="eventBus" class="com.google.common.eventbus.EventBus">
<constructor-arg value="batchEventBus"/>
</bean>

<bean id="batchTransactionManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="GLOBAL"/>
</bean>

```

```

<bean id="jobExplorer" class="org.springframework.batch.core.explore.support.JobExplorerFactoryBean"
    p:dataSource-ref="GLOBAL" p:tablePrefix="BATCH_" />
</beans>

```

To create a batch job, define in xml, which will be available to Spring Context at run-time. A sample demo has been provided in sample-job.xml in sample-app, with the name sampleDBtoCVSBatch. The name of the job id in xml should be exactly the same as the id attribute specified in the Batch annotation.

Sample Job

```

<job id="sampleDBtoCVSBatch" xmlns="http://www.springframework.org/schema/batch" restartable="true"
parent="baseJob" >
<step id="step1">
<tasklet transaction-manager="sample-tm" >
<chunk reader="itemReader" writer="csvWriter" commit-interval="4" />
</tasklet>
</step>
</job>

```

As shown in the above code block, each job must inherit from the finch provided baseJob to use the finch infrastructure. If the job does not inherit from baseJob, the finch provided abstraction will not be available.

The configuration for the itemReader, is provided below.

Item Reader

```

<bean id="itemReader" class="org.springframework.batch.item.database.JpaPagingItemReader" scope="step">
<property name="entityManagerFactory" ref="sample-em" />
<property name="queryString" value="select t from Trade t" />
<property name="pageSize" value="3" />
</bean>

```

The configuration for the csvWriter, is provided below.

CsvWriter Configuration

[Expand](#)

```

<bean id="csvWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">
<property name="resource" value="file:target/test-outputs/dbToCSV.csv" />
<property name="lineAggregator">
<bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">
<property name="delimiter" value="," />
<property name="fieldExtractor">
<bean class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor">
<property name="names"
value="tradePk,accountBalanceType,cancelDate,cancelReferenceNo,dataSource,employeePk,externalReferenc
eNo" />
</bean>
</property>
</bean>
</property>
</bean>

```

4.3 Data Setup for Batch Job

To run a batch job from UI one has to do the following steps :

1. First create the following tablesInsert values to the above tables.For Example:

INF_BATCH_MASTER

INF_BATCH_ROLE_PCPT

2. Insert values to the above tables.For Example

BATCH_NAME	STATUS	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE
1 test	NORMAL	27-SEP-13 20:28:52	27-SEP-13 20:28:52	FINCH-SAMPLE	27-SEP-13 20:28:52
BATCH_ROLE_PCPT_PK	BATCH_NAME	APPL_R...	APP_RE...	APP_UP...	CREATE...
1	1 test	1	27-SEP-13 20:28:52	27-SEP-13 20:28:52	FINCH-SAMPLE
2	2 sampleDBtoCSVBatch	1	27-SEP-13 20:28:52	27-SEP-13 20:28:52	FINCH-SAMPLE

4.4 Steps to Use Application Provided Settings for Batch Process

Steps are as follows:

1. Create app-ext folder in their application (for example, sample-dist/src/main/resources/console/app-ext)
2. After creating the above folder, create the file with the name console-option-usage-resource.properties.

Important Note

Name of the files must be identical as it is specified above.

After above file is created, insert application specific properties in that file.

5. Steps to Run Batch Runner in Console

Steps to register batch job is Zookeeper

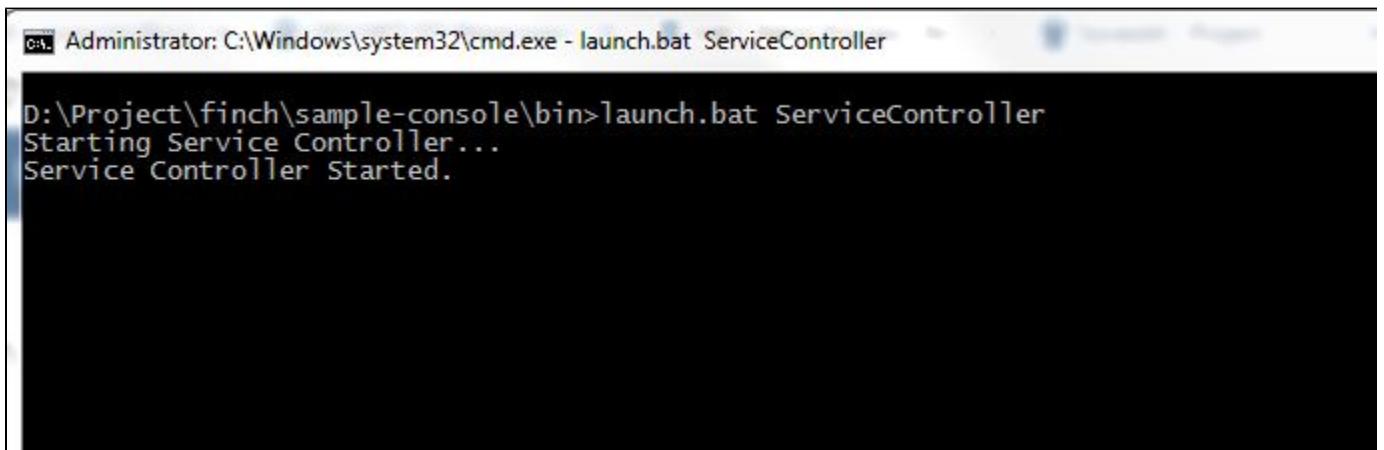
To register the batch in the Zookeeper registry nothing needs to be done. As this is enabled by default. If however an application decides that it does not need any of its batch to register in Zookeeper registry (started with ServiceController) then the developer has to set the property batch.registerInZooKeeper to true in console.properties.

If however the developer needs only some batches to register in Zookeeper, then he needs to specify the attribute register in com.nrft.fin.ch.inf.batch.metadata.Batch to false, while leaving the batch.registerInZooKeeper in console.properties as true.

If however batch.registerInZooKeeper is set to false, then finch will not register the batch to Zookeeper even if the register property is set to true.

Steps to run the batch runner are as follows:

- 1 . In Batch annotation if register = true then from console bin run the command to run the ServiceController: launch.bat ServiceController



Administrator: C:\Windows\system32\cmd.exe - launch.bat ServiceController

```
D:\Project\finch\sample-console\bin>launch.bat ServiceController
Starting Service Controller...
Service Controller Started.
```

2 .User can update the ports in the yml configuration file provided by finch (batch-rest-config.yml) .For more details please see the link <https://dropwizard.github.io/dropwizard/manual/configuration.html>.

6. Running Batch Job using Console

To run a batch job one has to use the following command from the console bin:

Command to run job

```
launch2.bat sampleDBtoCVSBatch -e FIN -f c:\messages.txt
```

finch provides authentication on batch.It can be configurable per batch.Folowing is the command to run batch for the authenticated user.To use authentication user need to do the above database setup. Once the command is submitted finch first checks if the username and password given match , if yes then it checks if the user has access to the batch by matching the roles configured for the batch in INF_BATCH_ROLE_PCPT.

Command to run job

```
launch2.bat sampleDBtoCVSBatch -e FIN -u <username> -p <password> -f c:\messages.txt
```

7. Document History

Version	Date	Purpose	Author
1.0	02-Feb-2015	Initial document	rahulc

3.4.6.2 Web Features

Introduction

This section describes about the following Web related features:

3.4.6.2.1 Dashboard

- 3.4.6.2.1.1 Business Counter
- 3.4.6.2.1.2 Chart Widget
- 3.4.6.2.1.3 Custom Widget Type
- 3.6.4.2.1.4 Menu

3.4.6.2.1.1 Business Counter

Introduction

Counter of a query result for a specific criteria can qualify as a business counter or feed. The business feed widget is displayed on the dashboard. It displays the feed along with a counter which specifies the total count of query results matches with the defined criteria. The business feeds are access controlled and thus only those business feeds can be viewed by the user who has the same application role to that of the business feed assigned. Query Result summary can be navigated from business feed counter. The result for an business feed can be viewed in the following modes:

- Query Result grid : Facilitates the user to perform several operations similar to finch query result screen.
- Pop-up Dialog mode : Result is shown in a pop-up dialog window. Multiple such pop-ups can be opened simultaneously. Refer to the [Business Counter Manual](#) for details.

Setup

Setup related information have been furnished below.

Database Table

The [DBD_FEED](#) database table contains the definition of the business feeds, as displayed below:

DBD_FEED_PK	DBD_FEED_NAME	DBD_FEED_URL
1	NORMAL TRD	trd/query/result?dashboardTradeType=NORMAL&status=NORMAL&screenId=TRDQR&versionNo=1.2&viewType=grid
2	CANCEL TRD	trd/query/result?dashboardTradeType=CANCEL&status=CANCEL&screenId=TRDQR&versionNo=1.2

The existence of viewType=screen parameter located at the end of the DBD_FEED_URL determines the feed result to be viewed in the query result grid in the main application window. If not present, the feed result should be viewed in dialog mode. This dialog can be minimized or restored.

Application Role

The mapping with the application role information is stored in [DBD_FEED_APPLN_ROLE_PTCP](#). The table structure contains the mapping of application role and business feed, as displayed below:

APPL_ROLE_PK	DBD_FEED_PK
1	1
2	2

The APPL_ROLE_PK is the foreign key reference of INF_APPLICATION_ROLE. Each APPL_ROLE_PK has an corresponding role assigned. For more details on relationship of Application role, refer to [Application Role](#).

The above mapping defines that user with APPL_ROLE_PK = 1 has access to NORMAL TRD but not to CANCEL TRD. As an user with APPL_ROLE_PK = 2, user has access to CANCEL TRD but not to NORMAL TRD.

Controller

Create a controller that fetches the count of the feed with the request mapping matching that of DATA_URL. This is used to get the total count of the feed, as displayed below:

```

@Controller
@RequestMapping(value = "/trd/dbdcounter")
public class TrdDbController {
    @RequestMapping(value = "/{mode}/{feedPk}", method = RequestMethod.GET)
    public void getTrdcount(ModelMap map, @PathVariable String mode, @PathVariable String feedPk) {
        FinchModelMap modelMap = new FinchModelMap(map);
        try{
            // do stuff
            int count = trdService.getRecordCount(mode);
            ImmutableMap<String, String> data = ImmutableMap.of("pk", feedPk, "count", Integer.toString(count));
            modelMap.addData(data);
        }catch (Exception e){
            log.error(e);
            modelMap.addError(e);
        }
    }
}

```

The trdService.getRecordCount(mode) gets the normal trade count, as shown below:

```

public int getNormalTradeCount() {
    String ql = "select count(*)from Trade t where t.versionNo = (select max(versionNo) from Trade t1 where
t1.referenceNo= t.referenceNo) and t.status='NORMAL'";
    Query q = getEntityManager().createQuery(ql);
    try{
        return ((Number)q.getSingleResult()).intValue();
    }catch (NoResultException nre){
        LOGGER.error("No result is returned for the given query :" + ql + "]");
        return 0;
    }
}

```

Pagination Support

To enable the pagination support in the feed result, the query controller used for fetching results must extend AbstractQueryController<?> and must contain a mapping for /count, as displayed below:

```

@RequestMapping(value = "/count", method = RequestMethod.POST)
public void getCount(HttpSession session, FinchModelMap modelMap,
@ModelAttribute("commandForm") TrdQueryCommandForm form) {

    @SuppressWarnings("unchecked")
    PagedListHolder<TrdQueryResultView> pagedResults = pagedListHolder<TrdQueryResultView>(
        session.getAttribute("pagedResults"));
    modelMap.getModelMap().put("count", pagedResults.getNrOfElements());
    modelMap.getModelMap().put("totalPageNo", pagedResults.getPageCount());

}

```

i18n Support for Business Feed

In finch, the business feed names can be i18n enabled. To enable i18n for business feed, insert a row in the DBD_FEED_NAME_CROSS_REF table. In this table data needs to be inserted against DBD_WIDGET_PK of the DBD_FEED table. Once data is setup, then the feed name would be taken from the DBD_FEED_NAME_CROSS_REF table. If there is no name cross ref information setup for a particular language

code (locale), the system tries to fetch the same with applications default language code (locale). Note if there is no entry in the DBD_FEE_D_NAME_CROSS_REF table, the feed name would be taken from the DBD_FEED table, as displayed below:

DBD_FEED_NAME_CROSS_REF_PK	DBD_FEED_PK	DBD_FEED_NAME	LANGUAGE_CODE
10001	10001	Trade Error	EN
10002	10001	貿易エラー	JP

3.4.6.2.1.2 Chart Widget

Introduction

The process of creating Chart Widget using finch framework has been described in this section. Application developer needs to follow the steps mentioned below to use chart widget features.

Steps for Using Chart Widget

In finch, definition of a Chart feed is stored in database table named DBD_CHART. The following image displays DBD_CHART table structure.

DBD_CHART_PK	DBD_CHART_NAME	DBD_CHART_URL	REQUEST_LAG_TIME_MSEC	STATUS	DATA_URL
10002	Trades	NO_HYPERLINK	600000	NORMAL	/trd/chart/NORMAL
10003	Trades 2	NO_HYPERLINK	600000	NORMAL	/trd/chart/NORMAL

Two most important attributes in the DBD_CHART table are DBD_CHART_NAME and DATA_URL. The DBD_CHART_NAME attribute contain the text that will be shown to user when they will select feed for the chart. Application developer need to insert feed name according to their requirement. Second attribute is the DATA_URL, this URL basically indicate the URL pattern that controller will handle.

In the above example we have kept two feed names namely 'Trades' & 'Trades 2' and both of this map to same URL '/trd/chart/NORMAL' (this is just for simplicity, application can have different URL).

Next, application developer need to create Controller class that will handle specified URL pattern and will return a JSON object which will contain chart data. JSON object which will contain chart data must contain all the required information related to chart.

The following code snippet contains the Controller class definition which will handle URL specified in this example.

Controller Class

```
@Controller
@RequestMapping("/trd/chart")
public class TradeChartController {

    //for mapping URL like '.../NORMAL/1003.json'
    @RequestMapping(value = "/NORMAL/{feedPk}", method = RequestMethod.GET)
    public void getTrdChartData(ModelMap map, @PathVariable("feedPk") String feedPk) {
        process(map, feedPk, null);
    }

    //for mapping URL like '.../NORMAL/1003/spline.json'
    @RequestMapping(value="/NORMAL/{feedPk}/{chartType}", method=RequestMethod.GET)
    public void getTradeChartData(ModelMap map, @PathVariable("feedPk") String feedPk,
        @PathVariable("chartType") String chartType){
        process(map, feedPk, chartType);
    }

    //Receiving chart type 'null' means chart type is same as defined by the application.
    public void process(ModelMap map, String feedPk, String chartType){
        //Do required stuffs
        ...
        ...
        ...
        }
    }
```

Next, inside the method that will handle given URL, we need to create JSON object related to chart. Following is an example of JSON object that we will explain in this example.

JSON Object

```
{  
  chart: {  
    type : 'line'  
  },  
  title {  
    text : 'Normal & Cancel trade for last three months'  
  },  
  xAxis : {  
    categories : ['Jan', 'Feb', 'Mar']  
  },  
  yAxis : {  
    title: {  
      text: 'Trade Count'  
    }  
  },  
  series :[  
    {  
      name : 'NORMAL',  
      data : [4, 8, 5]  
    }, {  
      name: 'CANCEL',  
      data: [10, 1, 8]  
    }]  
}
```

In this example we have created two series, one correspond to ‘NORMAL’ trade and another correspond to ‘CANCEL’ trade. Title of our chart is ‘Normal & Cancel trade for last three months’, It is a ‘line’ chart. This chart has 3 categories namely ‘Jan’, ‘Feb’ ‘Mar’.

Following is the example of the method that basically returns above JSON object.

URL request handling method

```
public void process(ModelMap map, @PathVariable String feedPk, String chartType) {  
    FinchModelMap modelMap = new FinchModelMap(map);  
    TradeChartObject chartObj = new TradeChartObject();  
    Chart chart = new Chart();  
    chart.setType("line");  
    chartObj.setChart(chart);  
    Title title = new Title();  
    title.setText("Normal & Cancel trade for last three months");  
    chartObj.setTitle(title);  
    XAxis x = new XAxis();  
    x.setCategories(ImmutableList.of("Jan", "Feb", "Mar"));  
    chartObj.setxAxis(x);  
    YAxis y = new YAxis();  
    title = new Title();  
    title.setText("Trade Count");  
    y.setTitle(title);  
    chartObj.setyAxis(y);  
    Series series1 = new Series();  
    series1.setName("NORMAL");  
    series1.setData(ImmutableList.of(4, 8, 5));  
    Series series2 = new Series();  
    series2.setName("CANCEL");  
    series2.setData(ImmutableList.of(10, 1, 8));  
    chartObj.setSeries(ImmutableList.of(series1, series2));  
    modelMap.addData(chartObj);  
}
```

How to Use?

Please refer to [Chart Widget manual](#).

3.4.6.2.1.3 Custom Widget Type

Introduction

finch provides 6 different widget types out of the box. These are MENU_SHORTCUT, SAVED TEMPLATE, SAVED QUERY, BUSINESS FEED, NOTIFICATION, TASK. These widget types perform some intended operation defined by finch. Application using finch can now also create their own widget type according to their own need.

How to create ?

finch provides a file named as 'app-init-config.js' to load client configuration just after the user is logged in. The file is currently stored under finch-dist/src/web/scripts but the client need to override the file.

1) Create a javascript file which contains the markup of the widget type wrapped in a function. The file structure must follow a pattern in order to work properly. For example,

trd-custom-widget.js

```
function FINCH$DBD$TRADE(parent) {
    this.hasFeeds = function() {
        return self.count !== 0;
    };
    this.inherit = FINCH$DBD$Content;
    this.inherit(parent, 'FINCH$DBD$TRADE');
    var self = this;
    this.feedDataUri = this.parent.contentUri;
    this.widgetPk = self.parent.pk;
    this.workspacePk = self.parent.parent.pk;
    this.widgetIdSelector = '#Widget' + self.parent.selector ;
    this.renderer = new FINCH$DBD$TRADE$renderer(this);
    this.feedByPk = function(pk) {
        return self.feeds[pk];
    };
    //other stuffs
    function FINCH$DBD$TRADE$renderer(object) {
        this.inherit = FINCH$DBD$Content$renderer;
        this.inherit(object);
        var self = this;
        var $widget = $('#Widget' + self.object.parent.selector);
        // markup
        this.myNotificationMarkup = "{{#each content}}"
        + "<div class='row notificationstatus={{status}} notificationpk={{notificationFeedPk}}>"
        + "<div class='left notificationBlockLabel >"
        + "<span class='notificationSummary' title=notification >{{tradeReferenceNo}}</span>"
        + "</div>"
        + " "
        + "<div class='clear rowfooter></div>"
        + "</div>"
        + "{{/each}}";
        self.template = Handlebars.compile(self.myNotificationMarkup);
        // other actions
    }
}
```

- 2) Create a configuration file which will contain the configuration settings like back color, fore color etc. for the custom widget type.

trd-custom-widget-config.js

```
function configure() {
    this.config = function(self,$widget){
        var configure = ''
            + '<div class="configureWidget contentArea">'
            + '  <div class="content">'
            + '    <div class="block">'
            + '      <div class="row">'
            + '        <div class="left contentBlockLabel">'
            + '          <span>' + FINCH$DBD$i18n.dashboard.background_color + '</span>'
            + '        </div>'
            + '        <div class="right contentBlockValue">'
            + '          <input id="backColor" value="" + self.object.backColor + ""/>'
            + '        </div>'
            + '        <div class="clear"></div>'
            + '      </div>'
            + '      <div class="row">'
            + '        <div class="left contentBlockLabel">'
            + '          <span>' + FINCH$DBD$i18n.dashboard.foreground_color + '</span>'
            + '        </div>'
            + '        <div class="right contentBlockValue">'
            + '          <input id="foreColor" value="" + self.object.foreColor + ""/>'
            + '        </div>'
            + '        <div class="clear"></div>'
            + '      </div>';
        configure += ''
            + ' <div class="clear"></div>'
            + ' </div>'
            + ' </div>'
            + ' </div>';
        var $configure = jQuery(configure);
        // configure back color, fore color
        return $configure;
    }
}
var TRADE = new configure();
```

3) Load the above two scripts in app-init-config.js

app-init-config.js

```
var scripts = [
    { path: FINCH.context.path + '/scripts/trd/trd-custom-widget-config.js'},
    { path: FINCH.context.path + '/scripts/trd/trd-custom-widget.js'}
];
FINCH.loadScript(scripts, {
    ordered: true,
    success: function() {
        console.log('All loaded from client side');
    }
});
```

4) Define the several attributes for custom widget type inside FINCH\$Load\$Custom\$Widget\$Config function :

app-init-config.js

```
var FINCH$Load$Custom$Widget$Config= function() {
    var json = {
        customWidget: [
            type: "TRADE", // the widget type same as DBD_WIDGET_TYPE
            cssClass: 'big task', // the css class to be used
            configureObject: TRADE, // the configurer object defined in trd-custom-widget-config.js
            rendererObject:FINCH$DBD$TRADE //the renderer object defined in trd-custom-widget.js
        ]
    };
    return json;
};
```

- 5) Define a controller for mapping the request for feed
- 6) Insert the widget type in DBD_WIDGET_TYPE. Set value of CUSTOM_WIDGET column to 'Y' (uppercase Y) for custom widget.
- 7) Map the widget type with the application role in DBD_WIDGET_TYPE_APP_ROLE_PT
- 8) Also make sure that the proper url pattern has been assigned for the action.

Templates

Custom Widget Renderer /Markup : [custom_widget_type_template.js](#)

Custom Widget Configuration : [custom_widget_type_config.js](#)

3.6.4.2.1.4 Menu

1. Scope

Menu in finch is a hierarchical tree structure, provides a platform to perform several operations in the application. Structure of menu has the parent-child relationship. Parent menu has associated anchors for each child menu. Parent menu uses this associated anchor in order to navigate to the child menu. This document covers the following

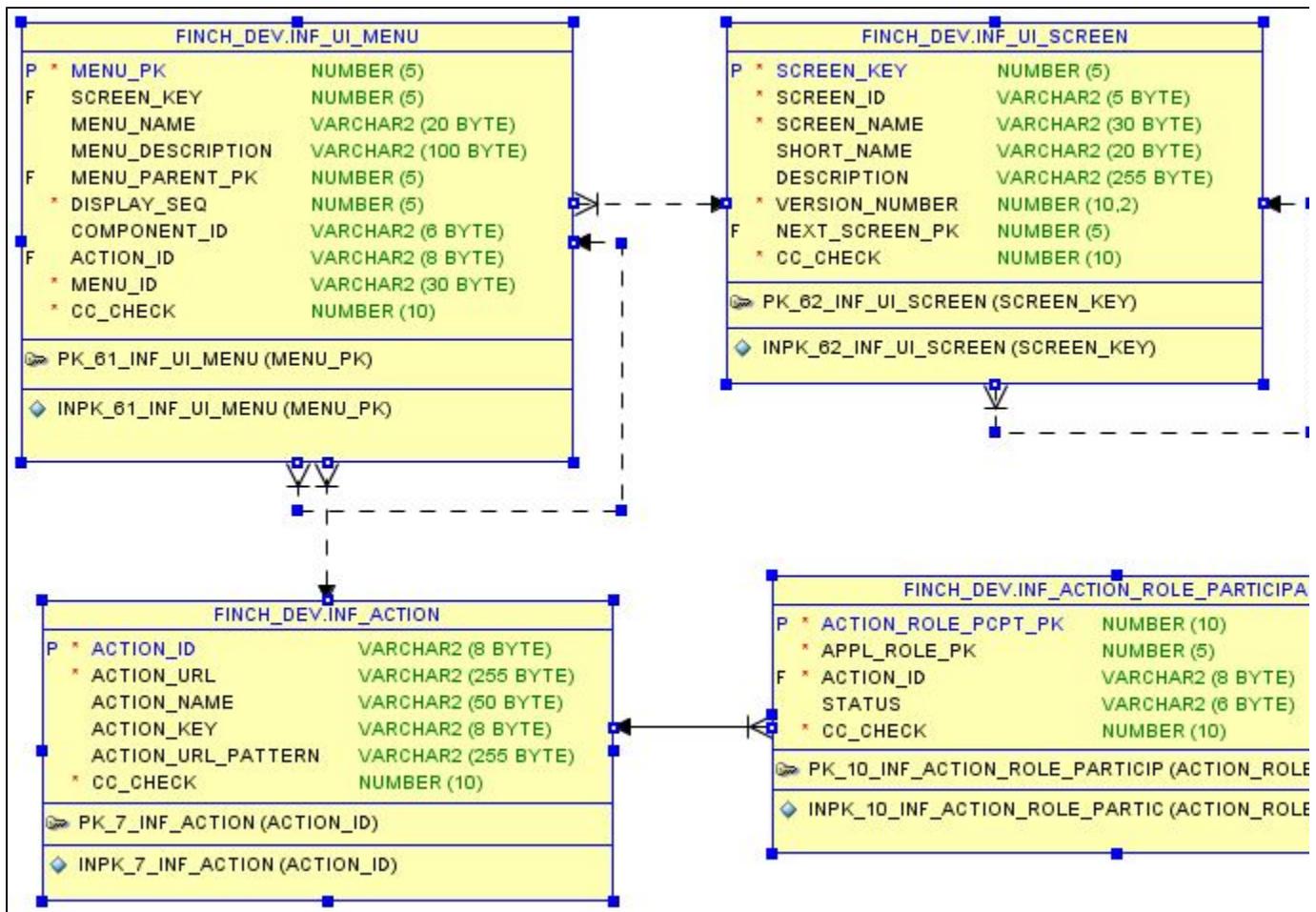
- DB changes to create menu.
- Access control of menu

2. Prerequisites

Developer must know what are menu, action and screen.

3. Sequence and steps

3.1 Table that needs to be configured to add new menu



3.1.1 Menu table

In finch, the menu definition is stored in a database table named INF_UI_MENU. The following table displays the example:

MENU_PK	SCREEN_KEY	MENU_NAME	MENU_DESCRIPTION	MENU_PARENT_PK	DISPLAY_SEQ	COMPONENT_ID	ACTION_ID	MENU_ID	CC_C
1	(null)	Main Menu	Main Menu	(null)	1	(null)	(null)	ROOT	1
2	(null)	Trade	Parent of Trade	1	1	TRD	(null)	TRD	1
3	1	Entry	Trade Entry	2	1	TRD	TRDTEN	TRDEN	1
4	2	Query	Trade Query	2	2	TRD	TRDTQR	TRDQR	1

- The 1st row is the root menu. Since it is the root menu, it does not have the MENU_PARENT_PK and its DISPLAY_SEQ as 1. Since it will not navigate to any screen, so ACTION_ID and SCREEN_KEY is null.
- The 2nd row is the Trade menu. Since it is the child of Main menu, its MENU_PARENT_PK is 1. Its DISPLAY_SEQ= 1 implies that it is the first child of Main menu. Its null value in ACTION_ID and SCREEN_KEY implies that it does not navigate to any screen.
- The 3rd row is the Trade Entry menu. Since it is the child of Trade menu, its MENU_PARENT_PK is 2. Its DISPLAY_SEQ = 1 implies that it is the first child of Trade menu. Its corresponding value in ACTION_ID and SCREEN_KEY signifies that this menu navigates

to a particular screen.

- The 4th row is the Trade Query menu. Since it is the child of Trade menu , its MENU_PARENT_PK is 2. Its DISPLAY_SEQ = 2 implies that it is the second child of Trade menu. Its corresponding value in ACTION_ID and SCREEN_KEY signifies that this menu navigates to a particular screen.

Important Note:

The 1st row in the above example must be created in the application database before creating any new menu.

3.1.2 Screen table

In finch, the screen key definition is stored in a database table named INF_UI_SCREEN. The following table displays the example:

SCREEN_KEY	SCREEN_ID	SCREEN_NAME	SHORT_NAME	DESCRIPTION	VERSION_NUMBER	NEXT_SCREEN_PK	CC_CHECK
1	TRDEN	Trade Entry	Trade Entry	Trade Entry	2	4	1
2	TRDQR	Trade Query	Trade Query	Trade Query	2	3	1
3	TRDRS	Trade Query Result	Trade Query Result	Trade Query Result	2	null	1
4	TENUC	Trade Entry User Confirmation	Trade Entry UC	Trade Entry User Confirmation	2	null	1

- The 1st row contains the definition of Trade Entry screen. The NEXT_SCREEN_PK = 4 signifies that the next screen to be fetched is the Trade Entry User Confirmation.
- The 2nd row contains the definition of Trade Query screen. The NEXT_SCREEN_PK = 3 signifies that the next screen to be fetched is the Trade Query Result.

3.1.3 Action table

In finch, the basic action definition is stored in a database table named INF_ACTION. The following table displays the example:

ACTION_ID	ACTION_URL	ACTION_NAME	ACTION_KEY	ACTION_URL_PATTERN	CC_CHECK
TRDTEN	trd/entry/init	Trade Entry	1	/trd/entry/.*	1
TRDTQR	trd/query/init	Trade Query	2	/trd/query/.*	1

- The 1st row defines the action related details for Trade Entry.
- The 2nd row defines the action related details for Trade Query.

3.1.4 Application Role and Action Mapping

In finch, the mapping between the application role and action is stored in a database table named INF_ACTION_ROLE_PARTICIPANT. Based on this mapping user can access the menu.

The following table displays the example:

ACTION_ROLE_PARTICIPANT_PK	APPL_ROLE_PK	ACTION_ID	STATUS	CC_CHECK
1	1	TRDTEN	NORMAL	1
2	2	TRDTQR	NORMAL	1

- The 1st row defines that it's the Main Menu, Trade Menu and Trade Entry is visible to the user whose application role is 1.
- The 2nd row defines that the Trade Query is visible to user whose application role is 2.

4. Configure screen name for multilingual support

Finch provides multilingual support for menu as an application can have menu in different language. For multilingual support Finch used localized properties file to resolve menu name.

e.g If application has support for English and Japanese language then, menu name is to be configured in following properties file

- menu_en.properties file - properties file for displaying menu in English
- menu_ja.properties file - properties file for displaying menu in Japanese
- menu.properties file - default properties file

Lookup properties file

If the user's language is set to English from preference then, in the following order the properties files will be searched

1. menu_en.properties - First this file will be searched.
2. menu.properties - If the menu_en.properties file is missing or not found then, this file will be searched.

Similarly if user's language is set to Japanese from preference then, in the following order the properties files will be searched

1. menu_ja.properties - First this file will be searched.
2. menu.properties - If the menu_ja.properties file is missing or not found then, this file will be searched.

Consider the INF_UI_MENU table

MENU_PK	SCREEN_KEY	MENU_NAME	MENU_DESCRIPTION	MENU_PARENT_PK	DISPLAY_SEQ	COMPONENT_ID	ACTION_ID	MENU_ID	CC_C1
1	(null)	Main Menu	Main Menu	(null)	1	(null)	(null)	ROOT	1
2	(null)	Trade	Parent of Trade	1	1	TRD	(null)	TRD	1
3	1	Entry	Trade Entry	2	1	TRD	TRDTEN	TRDEN	1
4	2	Query	Trade Query	2	2	TRD	TRDTQR	TRDQR	1

For multilingual support there should an entry in both properties files with the following convention

MENU_ID.name = <menu display name>

e.g.

menu.properties file

TRD.name = Trade

TRDEN.name = Entry

TRDQR.name = Query

menu_ja.properties file

TRD.name = ¥u53D6¥u5F1

TRDEN.name = ¥u767B¥u9332

TRDQR.name = ¥u30AF¥u30A8¥u30EA¥u30FC

Based on the application language setup menu will be displayed either in English or Japanese.

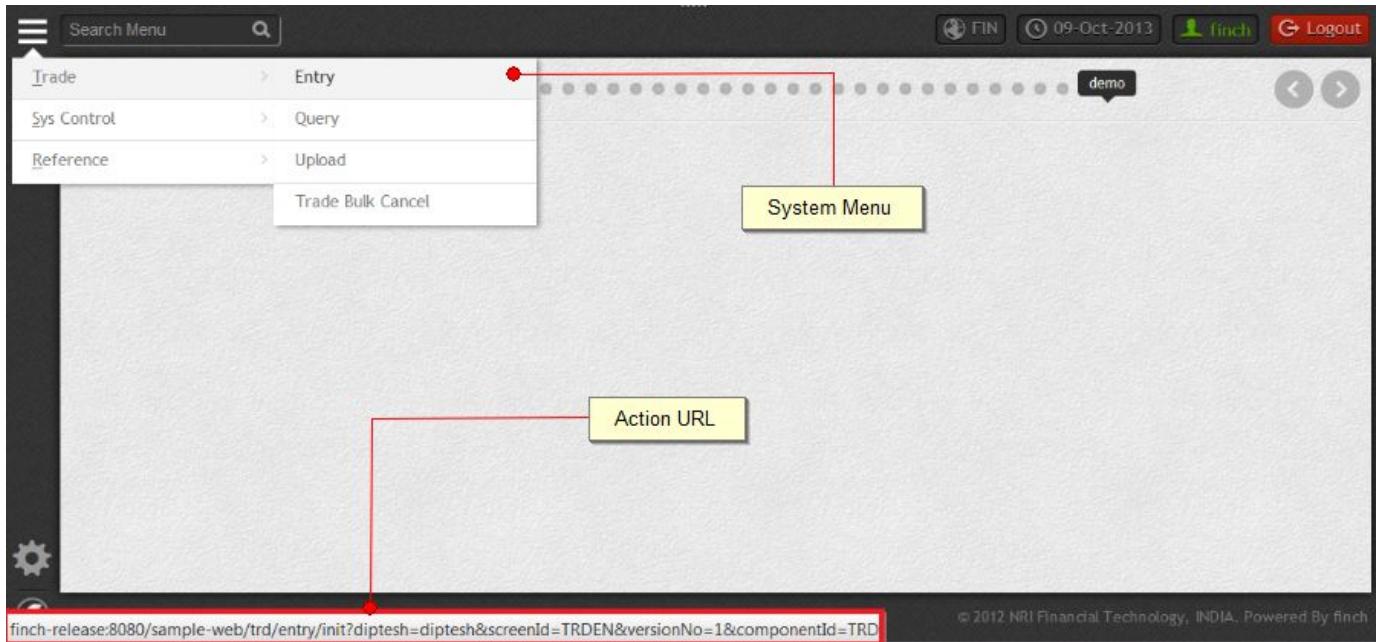
If the properties file is missing or application does not configure the properties file then, INF_UI_MENU.MENU_NAME will be displayed in menu.

5. Configure keyboard shortcut for accessing Menu

Application can also configure keyboard shortcut for accessing menu. See [Menu Shortcut Configuration Guide](#) for details of keyboard shortcut configuration.

6. Access to Menu

User having Application Role 1, only Trade Entry screen is accessible as shown below:



The action URL fetched from INF_ACTION is mapped with a controller which does the business processing.

Important Note:

The first level menu can not have a leaf menu (clicking on that menu opens a screen). It can have a menu that have one or more child menu.

7. Example to set up Menu

Now we will see an example to set up menu for sample application. Sample application do no need to insert the Main Menu in INF_UI_MENU table, as finch provide this entry in INF_UI_MENU table. Sample application only need to

insert the menu entry they want.

```
INF_UI_MENU table
insert into INF_UI_MENU
(
    MENU_PK , SCREEN_KEY , MENU_NAME , MENU_DESCRIPTION ,
    MENU_PARENT_PK , DISPLAY_SEQ , COMPONENT_ID , ACTION_ID ,
    MENU_ID , APP_REGI_DATE , APP_UPD_DATE , CREATED_BY ,
    CREATION_DATE , UPDATED_BY , UPDATE_DATE , CC_CHECK
)
values (
    <VALUE_FOR_MENU_PK> , <VALUE_FOR_SCREEN_KEY> , <VALUE_FOR_MENU_NAME> , <VALUE_FOR_MENU_DESCRIPTION> ,
    <VALUE_FOR_MENU_PARENT_PK> , <VALUE_FOR_DISPLAY_SEQ> , <VALUE_FOR_COMPONENT_ID> , <VALUE_FOR_ACTION_ID> ,
    <VALUE_FOR_MENU_ID> , <VALUE_FOR_APP_REGI_DATE> , <VALUE_FOR_APP_UPD_DATE> , <VALUE_FOR_CREATED_BY> ,
    <VALUE_FOR_CREATION_DATE> , <VALUE_FOR_UPDATED_BY> , <VALUE_FOR_UPDATE_DATE> , <VALUE_FOR_CC_CHECK
) ;
```

For example - If application has a menu 'xyz' and under 'xyz' there are two child menus 'abc' and 'pqr' then following scripts needed to run.

In INF_ACTION table leaf menu(clicking on that menu opens a screen) value is to be inserted. For this example need to insert two values for abc and pqr.

we have assigned ACABC action id for abc and ACPQR for pqr.

```
INF_ACTION table
INSERT INTO INF_ACTION
  (ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,
  APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,
  UPDATED_BY,UPDATE_DATE,ACTION_URL_PATTERN,CC_CHECK,COMPONENT_ID
  )
VALUES
  ('ACABC','/trd/entry/init','Trade Entry','1',
  sysdate,sysdate,'FINCH-SAMPLE',sysdate,
  'FINCH-SAMPLE',sysdate,'/trd/entry/.*',1,'TRD'
  );
INSERT INTO INF_ACTION
  (ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,
  APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,
  UPDATED_BY,UPDATE_DATE,ACTION_URL_PATTERN,CC_CHECK,COMPONENT_ID
  )
VALUES
  ('ACPQR','/trd/entry/init','Trade Entry','1',
  sysdate,sysdate,'FINCH-SAMPLE',sysdate,
  'FINCH-SAMPLE',sysdate,'/trd/entry/.*',1,'TRD'
  );
```

After executing the above script Screen Key definition is to be inserted in INF_UI_SCREEN table.

For this example 1789 is defined for screen abc and 2789 for screen pqr

```
INSERT INTO INF_UI_SCREEN
  (SCREEN_KEY,SCREEN_ID,SCREEN_NAME,SHORT_NAME,
  DESCRIPTION,VERSION_NUMBER,NEXT_SCREEN_PK,APP_REGI_DATE,
  APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
  UPDATE_DATE,CC_CHECK,COMPONENT_ID
  )
VALUES
  (1789,'ABCID','Menu 1','Menu 1',
  'Menu 1 desc',1,null,sysdate,
  sysdate,'FINCH-SAMPLE',sysdate,'FINCH-SAMPLE',
  sysdate,1,'TRD'
  );
INSERT INTO INF_UI_SCREEN
  (SCREEN_KEY,SCREEN_ID,SCREEN_NAME,SHORT_NAME,
  DESCRIPTION,VERSION_NUMBER,NEXT_SCREEN_PK,APP_REGI_DATE,
  APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
  UPDATE_DATE,CC_CHECK,COMPONENT_ID
  )
VALUES
  (2789,'PQRID','Menu 2','Menu 2',
  'Menu 2 desc',1,null,sysdate,
  sysdate,'FINCH-SAMPLE',sysdate,'FINCH-SAMPLE',
  sysdate,1,'TRD'
  );
```

Now insert value for xyz menu.

```
Insert value for "xyz" menu
INSERT INTO INF_UI_MENU
  (MENU_PK,SCREEN_KEY,MENU_NAME,MENU_DESCRIPTION,
  MENU_PARENT_PK,DISPLAY_SEQ,COMPONENT_ID,ACTION_ID,
  MENU_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,
  CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK
  )
VALUE
  (987,null,'xyz','xyz menu drscription',
  1,2,'INF',null,
  'SYS',sysdate,sysdate,'FINCH-SAMPLE',
```

```

    sysdate,'FINCH-SAMPLE',sysdate,1
);

```

In the above script SCREEN_KEY and ACTION_ID is set to null because it will not navigate to any screen instead it will have some two child menu abc and pqr.

To have these two child menu under 'xyz' menu the following script need to run

```

INSERT INTO INF_UI_MENU
(menu_pk, screen_key, menu_name, menu_description,
menu_parent_pk, display_seq, component_id, action_id,
menu_id, app_regi_date, app_upd_date, created_by,
creation_date, updated_by, update_date, cc_check
)
VALUES (9871, 1789, 'abc', 'abc screen',
987, 1, 'INF', 'ACABC',
'ABC',sysdate,sysdate, 'FDBA-60',
sysdate, 'FDBA-60', sysdate, 1
);
INSERT INTO INF_UI_MENU
(menu_pk, screen_key, menu_name, menu_description,
menu_parent_pk, display_seq, component_id, action_id,
menu_id, app_regi_date, app_upd_date, created_by,
creation_date, updated_by, update_date, cc_check
)
VALUES (9872, 2789, 'pqr', 'pqr screen',
987, 2, 'INF', 'ACPQR',
'PQR',sysdate,sysdate, 'FDBA-60',
sysdate, 'FDBA-60', sysdate, 1
);

```

In INF_ACTION_ROLE_PARTICIPANT table following entries is to be made

```

INF_ACTION_ROLE_PARTICIPANT table
INSERT INTO INF_ACTION_ROLE_PARTICIPANT
(ACTION_ROLE_PCPT_PK,APPL_ROLE_PK,ACTION_ID,APP_REGI_DATE,
APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
UPDATE_DATE,STATUS,CC_CHECK)
VALUES
(3789,1,'ACABC',sysdate,
sysdate,'FINCH-SAMPLE',sysdate,'FINCH-SAMPLE',
sysdate,'NORMAL',1
);
INSERT INTO INF_ACTION_ROLE_PARTICIPANT
(ACTION_ROLE_PCPT_PK,APPL_ROLE_PK,ACTION_ID,APP_REGI_DATE,
APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
UPDATE_DATE,STATUS,CC_CHECK)
VALUES
(4789,1,'ACPQR',sysdate,
sysdate,'FINCH-SAMPLE',sysdate,'FINCH-SAMPLE',
sysdate,'NORMAL',1
);

```

8. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	02-02-2015	Initial version	Mithu Tokder

3.4.6.2.2 Entry/Amend Wizard Template

1. Benefits of Using Wizard

finch wizard is designed to accept large information from the user in a sequential flow. A large number of pages is used sequentially through which the user will be guided to complete a premeditated action. The pages appear as tabs inside a single wizard page. The user can navigate through the pages using standard navigation buttons like Previous or Next and also by clicking on the Page icons placed just above the tabs. The basic idea is to contain all the pages as tabs inside a single wizard page and to accept information from the user one page at a time, until the desired action is complete.

2. Design

The following image shows how the user can navigate through different pages. The visible tab among the navigation tabs identifies the current active content.

Navigation tabs

TRADE ENTRY [09-Oct-2013]

1 Trade General 2 Trade Details

Trade Type: EQ Trade Date: Value Date: Buy/Sell Orientation:

Account No: Inventory Account No: Security: Quantity:

Trade Currency: Execution Market: Settlement Currency: Price:

Account Balance Type:

SAVE TEMPLATE

Next page

Reset values

RE SET

Click Next to access the next tab page.

TRADE ENTRY [09-Oct-2013]

1 Trade General 2 Trade Details

Trade Type: EQ Trade Date: 14-Nov-2013 Value Date: 13-Nov-2013

Buy/Sell Orientation: AB Account No: 121212005 Inventory Account No: 121212007

Security Code: IBM Quantity: 100 Price: 200

Execution Market: BR001 Trade Currency: USD Settlement Currency: USD

Account Balance Type: 01

RR

RR Role	RR Number	Commission Amount
Executing RR	1	1.0
RR	1	1.0
Split RR1	1	1.0
Split RR2	1	1.0
Trader	1	1.0

Net Amount: 20000.00 External Reference No: 1234

SAVE TEMPLATE

Submit the page

Previous page

RE SET **PREVIOUS** **SUBMIT**

Click Submit to proceed to the User Confirmation mode.

User confirmation

TRADE ENTRY USER CONFIRMATION [20131009]

1 Trade General 2 Trade Details

Trade Type: EQ Inventory Account No: 121212009 Inventory A/C Short Name: USA-JPMC-CUR

Account Balance Type: 1 Account No: 121212005 Short Name: JPN-HSBC-CUR

Security Code: IBM Price: 20 Quantity: 12

Execution Market: BR001 Trade Currency: JPY Settlement Currency: JPY

Active Tab marked with green

Back

Confirm

BACK

CONFIRM

Click Trade Details tab.

Click Confirm to proceed to the System Confirmation mode.

The Confirm Message displays. Click OK to conclude the process.

3. How to Implement a Multi-Page User Action by Wizard

Steps for implementing a user action via the finch-wizard architecture have been described below.

3.1. User Interface configurations

3.1.1. Configure View-Tiles

The tiles definition have to be modified to contain the individual pages as a tab inside the wizard page. The view tiles must extend wizard-page tile and the wizard-page attribute for the tile definition must have the view location as its value.

views.xml

```
<!-- Trade General entry page -->
<definition name="tradeGeneralEntry" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeGeneralEntry.jspx"/>
</definition>

<!-- Trade General detail page -->
<definition name="tradeGeneralDetail" extends="wizard-page">
    <put-attribute name="wizard-page" value="/WEB-INF/views/trd/tradeGeneralDetail.jspx"/>
</definition>
```

3.1.2. Define FINCH\$Wizard\$OnPageLoad javascript Function

When a page is loaded as a tab inside the wizard-page, the FINCH\$Wizard\$OnPageLoad javascript function is called first so that page specific javascript can be loaded. The definition of FINCH\$Wizard\$OnPageLoad is meant for containing the on-page-load function.

FINCH\$Wizard\$OnPageLoad function

```
var FINCH$Wizard$OnPageLoad = function(){
    FINCH.loadLocalizedScript([
        {path: FINCH.context.path + '/scripts/trd/sample-trd-i18n.js', async: false}
    ]);

    FINCH.loadScript([
        {path: FINCH.context.path + '/scripts/inf/utilValidator.js', async: false},
        {path: FINCH.context.path + '/scripts/inf/finch-datevalidation.js', async: false},
        {path: FINCH.context.path + '/scripts/inf/utilCommons.js', async: false},
        {path: FINCH.context.path + '/scripts/inf/finch-popquery-form.js', async: false}
    ]);

    FINCH.loadScript([{path: FINCH.context.path + '/scripts/inf/finch-form-utils.js'}], {
        success: function() {
            $('#input.numVal').numVal();
        }
    });
};
```

3.1.3. Setup the Wizard for Uploading Multipart Data

The page, which needs to upload multipart form data (for example providing file upload feature), need to setup the wizard accordingly, as shown in the following code snippet:

Apply multipart

```
var context = jQuery('#' + '${commandForm.uniqueId}`).parent().parent();
var wizard = jQuery('#' + '${commandForm.uniqueId)').parent().parent().parent().data('FINCH$Wizard');
wizard.apply({
    multipart: true,
    multipartForm: '#content form'
});
```

The result view page returned successfully uploading the multipart data need to include all the script within a block checking if the current window is the top window, as shown in the following code snippet:

```
Checking condition  
if (window == window.top) {  
    //all required script goes here  
}
```

3.1.4. Register Hooks for Wizard Buttons

Various hooks can be registered for particular events, as displayed below:

```
Register Hooks  
var $finch$wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');  
$finch$wizard.register('previous',validateHook);  
$finch$wizard.register('next',validateHook);  
$finch$wizard.register('submit',validateHook);  
$finch$wizard.register('unload',unloadHook);
```

Hooks can also be de-registered from particular events, as needed.

```
Deregister Hooks  
var $finch$wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');  
$finch$wizard.deregister('previous',validateHook);  
$finch$wizard.deregister('next',validateHook);  
$finch$wizard.deregister('submit',validateHook);  
$finch$wizard.deregister('unload',unloadHook);
```

3.1.5. Set the URL for Exiting From the Wizard

The Exit Url of the Wizard has to be set on the page loading of the very first view-tile of the first mode. This Url is used while leaving the Wizard and It is invoked after the successful user interaction, when the user clicks the OK button.

```
Exit URL of Wizard  
// The Exit Url is pointing to the Dashboard  
var $finch$wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');  
$finch$wizard.apply({exitUri: FINCH.context.path});
```

3.1.6. Enabling and Disabling Wizard buttons

If an application page contains elements that takes long time to render, the application can enable the wizard buttons after their elements have been loaded completely. The application needs to call the FINCH\$Wizard\$OnPageComplete on the jsp containing the elements. For example,

```

tradeDetailsEntry.jspx

var FINCH$Wizard$OnPageLoad = function() {
    // to stuffs on page load
}
var FINCH$Wizard$OnPageComplete = function() {
    // checks if tax fee grid is present on the dom
    if($('.grid-canvas','#taxFeeGrid').length > 0) {
        var $finch$Wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');
        $finch$Wizard.renderer.enableActions(); // enable the buttons
        return;
    }
    // else call the function recursively
    else {
        setTimeout(FINCH$Wizard$OnPageComplete,1);
    }
};

```

If FINCH\$Wizard\$OnPageComplete function is not implemented in the application jspx, the buttons will get enabled by default without assuring whether all the elements has been loaded completely.

3.1.7. Allow or Disallow removal of form item

If application wants to decide whether any form item should be allowed to removed or not then application need to implement FINCH\$Wizard\$FormItem\$Remove\$Hook hook in their jspx file. If client implementation returns 'true' then finch will take action to delete the form item, but if it returns 'false' then finch will not delete the form item. An example is shown below

```

var FINCH$Wizard$FormItem$Remove$Hook = function(ev){
    //All the form item is allowed to be removed by user.
    return true;
}

```

3.2. Server Configurations

The following buttons are available for general usage:

Buttons available in FinchWizard

```

public enum FinchWizardButton {
    PREVIOUS, NEXT, RESET, BACK, SUBMIT, CONFIRM, OK, CUSTOM1, CUSTOM2, CUSTOM3
}

```

The following wizard modes are supported:

Modes available in FinchWizard

```

public enum FinchWizardMode {
    EDIT, VIEW, USER_CONFIRMATION, SYSTEM_CONFIRMATION
}

```

Warning !!

The above FinchWizardButton and FinchWizardMode is only look-up information and the application should not modify these.

3.2.1. Configure FinchWizard

Wizard is a container for the various instances of FinchWizardPages along with their relative FinchWizardModes with corresponding Finch WizardButtons. A FinchWizardPage is the most basic element of FinchWizard that represents each and every view-tile along with the mode with which it is associated and the buttons to be shown in that page. There is also provision for specifying the buttons to be excluded while showing a view-tile in a particular mode.

FinchWizardPage

```
Map<FinchWizardMode, String> page0tiles = new HashMap<FinchWizardMode, String>();
page0tiles.put(EDIT, "tradeGeneralEntry");
page0tiles.put(USER_CONFIRMATION, "tradeGeneralDetail");
page0tiles.put(SYSTEM_CONFIRMATION, "tradeGeneralDetail");

Map<FinchWizardMode, FinchWizardButton[]> page0IncludeButtons = ImmutableMap.of(EDIT, new
FinchWizardButton[] {PREVIOUS, NEXT, RESET, SUBMIT});
Map<FinchWizardMode, FinchWizardButton[]> page0ExcludeButtons = ImmutableMap.of(EDIT, new
FinchWizardButton[] {PREVIOUS, SUBMIT});

FinchWizardPage pages0 = new FinchWizardPage("trd.tradeentryaction.label.tradegen", page0tiles,
page0IncludeButtons, page0ExcludeButtons);
```

Buttons can also be added directly to the wizard instead of adding explicitly to each and every page. The FinchWizard needs the followings for proper initialization:

- Wizard Display Name
- FinchWizard view-tile name
- baseUri
- FinchWizardModes
- FinchWizardPages
- FinchWizardButtons.

The wizard-title parameter can be a title-key for i18n support. The Wizard view-tile name has to be defined for wizard-page in tiles-configuration. Code snippet is as follows:

Instance Creation of FinchWizard

```
Map<FinchWizardMode, String> page0tiles = new HashMap<FinchWizardMode, String>();
page0tiles.put(EDIT, "tradeGeneralEntry");
page0tiles.put(USER_CONFIRMATION, "tradeGeneralDetail");
page0tiles.put(SYSTEM_CONFIRMATION, "tradeGeneralDetail");

Map<FinchWizardMode, String> page1tiles = new HashMap<FinchWizardMode, String>();
page1tiles.put(EDIT, "tradeDetailsEntry");
page1tiles.put(USER_CONFIRMATION, "tradeDetailsParticular");
page1tiles.put(SYSTEM_CONFIRMATION, "tradeDetailsParticular");

FinchWizardPage[] pages = new FinchWizardPage[2];
Map<FinchWizardMode, FinchWizardButton[]> page0ExBttns = ImmutableMap.of(EDIT, new
FinchWizardButton[] {PREVIOUS, SUBMIT});
Map<FinchWizardMode, FinchWizardButton[]> page1ExBttns = ImmutableMap.of(EDIT, new
FinchWizardButton[] {NEXT});
pages[0] = new FinchWizardPage("trd.tradeentryaction.label.tradegen", page0tiles, null, page0ExBttns);
pages[1] = new FinchWizardPage("trd.tradeentryaction.label.tradedetails", page1tiles, null, page1ExBttns);

Map<FinchWizardMode, FinchWizardButton[]> buttons = ImmutableMap.of(EDIT, new FinchWizardButton[]
{PREVIOUS, NEXT, RESET, SUBMIT},
USER_CONFIRMATION, new FinchWizardButton[] {BACK, CONFIRM},
SYSTEM_CONFIRMATION, new FinchWizardButton[] {OK});

FinchWizardMode[] modes = new FinchWizardMode[] {EDIT, USER_CONFIRMATION,
SYSTEM_CONFIRMATION};
FinchWizard finchWizard = new FinchWizard("trd.tradeentryaction.label.tradeentry", "wizard", "trd/entry",
modes, pages, buttons);
```

In the above example, first two FinchWizardPages have been created. The FinchWizard will move through three basic modes, which are EDIT, USER_CONFIRMATION and SYSTEM_CONFIRMATION. So the wizard will start in EDIT mode and the respective view-tiles for EDIT mode (page0 -> tradeGeneralEntry, page1 -> tradeDetailsEntry) are fetched from the mode-tile map inside the two FinchWizardPages of the wizard. These are then added as tabs into the wizard-page.

The Previous and Next buttons is for the user to navigate between the two view-tiles (now tabs inside the wizard page) while staying inside the same wizard-mode (EDIT). When user is in the last page of this EDIT mode, on submitting the form, the FinchWizard changes its mode to the next mode in sequence and consequently load the view-tiles for the next mode (USER_CONFIRMATION). It is not necessary to have same number of pages in each mode. The user can have any number of pages in any mode with proper tiles configuration..

3.2.2 Configure CommandForm

The CommandForm is used to collect information and interacts with the user under the control of the FinchWizard must extend FinchAbstractWizardCommandForm class. This superclass carries the FinchWizard instance. The FinchWizard instance can be set to the commandForm right at the initialization of the commandForm through constructor. The FinchWizard can also be injected from outside like the controller after CommandForm initialization, but it has to be there before the execution of the wizard starts.

3.2.3 Configure Controller

FinchAbstractWizardController is an abstract class that is the highest level parent of the controllers that will work with the FinchWizard. This Controller handles the mechanism for CommandForm initialization and page Navigation along with page validation.

FinchAbstractEditWizardController is another abstract class that extends FinchAbstractWizardController and handles the mechanism related to an Editing action through wizard like Reset, Submit and Confirmation.

Thus when a Controller extends FinchAbstractEditWizardController there is scope for defining the method body of the following protected methods, like doInit(), doNavigate(), doValidate(), doReset(), doSubmit() and doConfirm(). These methods are called appropriately through the wizard allowing easy implementation business logic within these few methods only.

The following diagram shows the order in which these protected methods are called when the Next action is done by the user.

The following diagram shows the order in which these protected methods are called when the Previous action is done by the user.

The following diagram shows the order in which these protected methods are called when the Submit action is done by the user.

The following diagram shows the order in which these protected methods are called when the Confirm action is done by the user.

For example:

TradeEntryController.java

```
@Controller
@SessionAttributes("commandForm")
@RequestMapping(value = "/trd/entry/**")
public class TradeEntryController extends FinchAbstractEditWizardController<TradeEntryCommandForm>
implements MessageSourceAware{

    private static final Logger log = LoggerFactory.getLogger(TradeEntryController.class);

    private MessageSource messageSource;

    @Override
    protected void doInit(TradeEntryCommandForm commandForm) throws FinchException {
        try {
            log.debug("Initialising TradeEntryCommandForm ...");
            // do custom stuff

        } catch (Exception e) {

            log.error("Failed to make a view for the trade data", e);

            throw new TrdException("trd.tradeentryaction.error.populate", e);
        }
    }
}
```

The application can override the implementation of the following:

- 1) `doInit(CommandForm form, FinchModelMap map)` - called during initialization of first screen
- 2) `doSubmit(CommandForm form, int index, FinchModelMap map)` - called on click of SUBMIT
- 3) `doReset(CommandForm form, int index)` - called on click of RESET
- 4) `doConfirm(CommandForm form, FinchModelMap map)` - called on click of CONFIRM
- 5) `getQueryReportResult(CommandForm form)` - called for generating report for PDF, XLS, CVS
- 6) `getQueryResult(CommandForm form)` - called for fetching query result.
- 7) `getReportParameters(CommandForm form)` - called for getting the report generation parameters
- 8) `getComponent()` - get the current component
- 9) `getReportName()` - called for providing the report name.

3.2.4 Skipping Wizard Pages / Modes

finch provides a feature for skipping wizard pages or modes according to the application requirement. The following assumptions has been considered while implementing the feature:

- 1) Wizard cannot skip the next mode, if the mode to be skipped is the last mode.

- 2) Wizard cannot skip the previous mode, if the mode to be skipped is the first mode.
- 3) Application has to define whether the skipping is required in the wizard by setting the skipWizard attribute in command form while initializing the controller. For example,

TradeEntryController.java

```
@Override
protected void doInit(TrdBulkCancelCommandForm form, FinchModelMap modelMap)
    throws FinchException {
    // other stuffs
    commandForm.setSkipWizard(true);
}
```

- 4) If the skipWizard attribute in command form is true, the wizard cannot be navigated through carousel(navigation bar).

finch provides the following api's for application to skip pages / modes:

[doSkipNextMode\(FinchModelMap map, CommandForm form\)](#) : called by application for skipping next mode. The abstract command form contains a boolean flag skipNextMode which the application must set to true if skipping is required.

[doSkipPreviousMode\(FinchModelMap map, CommandForm form\)](#) : called by application for skipping previous mode. The abstract command form contains a boolean flag skipPreviousMode which the application must set to true if skipping is required.

[doSkipNextPage\(FinchModelMap map, CommandForm form\)](#) : called by application for skipping next page. The abstract command form contains a boolean flag skipNextPage which the application must set to true if skipping is required.

[doSkipPreviousPage\(FinchModelMap map, CommandForm form\)](#) : called by application for skipping previous page. The abstract command form contains a boolean flag skipPreviousPage which the application must set to true if skipping is required.

Let us consider the following wizard structure:

EDIT	USER CONFIRMATION	SYSTEM CONFIRMATION
Page 0 → Page 1 → Page 2	Page 3 → Page 4	Page 5 → Page 6

- 1) If skipNextPage is true on Page 0, then Page 1 will be skipped and Page 2 will be displayed.
- 2) If skipNextPage is true on Page 1, then Page 2 will be skipped and Page 3 will be displayed. Here mode will be switched from EDIT to USER CONFIRMATION. Page 2 must contain the SUBMIT,CONFIRM button that will trigger the mode transition.
- 3) Clicking PREVIOUS on Page 2, then Page 0 will be displayed.
- 4) If skipNextMode is true on Page 0, then Page 5 will be displayed. Here mode will be switched from EDIT to SYSTEM CONFIRMATION. Page 2 must contain either SUBMIT or CONFIRM button for this transition.
- 5) If skipping previous page is requested from Page 1, the skipping will not occur.
- 6) If skipping next mode is requested from Page 3 or Page 4, the skipping will not occur.
- 7) If skipPreviousMode is requested from Page 5, Page 2 will be displayed. Here mode will be switched from USER CONFIRMATION to EDIT

The following is the code snippet for various scenario's :

TradeEntryController.java

```
@Override  
protected void doSkipNextMode(FinchModelMap modelMap, TradeEntryCommandForm form) throws  
FinchException {  
    // other stuffs  
    FinchWizard wizard = form.getWizard();  
    if(wizard.currentMode().name() == "EDIT" && wizard.getModelIndex() == 0) {  
        form.skipNextMode(true);  
    }  
}
```

3.4.6.2.3 Query/Query Result

Introduction

This section describes how a developer can create a query screen using the finch framework and what all the developer needs to write in both server and client side code.

Some of the sections described in the subsequent sections are:

- Controller
- Command Form
- Result View
- Service
- Criteria Builder
- Screen Data Setup
- Presentation JSPX
- Save Query
- Pre-Fetching Data

Controller

Step: 1

To write a query controller we must write a Spring controller class which would extend the AbstractQueryController class of finch framework. The following code snippet is for writing a query controller class:

```
@Controller  
@RequestMapping(value = "/trd/query")  
@SessionAttributes({"commandForm"})  
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements  
MessageSourceAware {  
  
    public TrdQueryController () {  
        commandFormType = TrdQueryCommandForm.class;  
    }  
    ...  
}
```

The controller class called TrdQueryController is created by extending AbstractQueryController. While creating the controller class, ensure about the followings:

- @Controller annotation is used in the class level
- @RequestMapping annotation is specified for the class level

- commandForm attribute is bound with Session using @SessionAttributes annotation tag
- The class to implement MessageSourceAware interface for resource bunndle
- The command form is specified with the AbstractQueryController. We would see what is command form at a latter stage

Step: 2

Once the above is done, then the necessary methods to be implemented to make the controller work. Firstly declare the initialization method for the controller, which is the doInit().

```

@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

protected void doInit(TrdQueryCommandForm form, FinchModelMap modelMap) throws FinchException {
initTradeQuery(modelMap,form);
}

private void initTradeQuery(FinchModelMap modelMap, TrdQueryCommandForm initCommandForm) {
try
{ // Initialisation code
...
} catch (FinchException e){
modelMap.addError(e);
}

}

}

```

In the initialisation method, all the properties which are required at the time of displaying the query criteria page (such as drop down fields, search fields etc) are initialized.

Step: 3

Once initialization method is created, override the getQueryFormViewName() and getQueryResultViewName() methods, where the tiles view name which is defined in the views.xml file for defining the JSPX files, is specified.

```

@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

.....
@Override
protected String getQueryFormViewName() {
    return "tradeQueryCriteria";
}

@Override
protected String getQueryResultViewName() {
    return "tradeQueryResult";
}

.....
}

```

Step: 4

Entries to be made in the view.xml file

```

<definition name="tradeQueryCriteria" extends="queryCriteria">
<put-attribute name="content">
<definition extends="criteria_and_order">
<put-attribute name="criteria" value="/WEB-INF/views/trd/tradeQueryCriteria.jspx"/>
</definition>
</put-attribute>
</definition>

<definition name="tradeQueryResult" extends="result">
<put-attribute name="content">
<definition extends="queryResult">
<put-attribute name="resultGrid" value="/WEB-INF/views/trd/tradeQueryResult.jspx"/>
</definition>
</put-attribute>
</definition>

```

Step: 5

After the `getQueryFormViewName()` and `getQueryResultViewName()` is overridden and appropriate entries are made in the `views.xml` file, override certain additional methods.

```

@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

    .....

    @Override
    protected String getComponent() {
        // Specify the component under which the query ui is being created
        return "TRD";
    }

    @Override
    public void setMessageSource(MessageSource messageSource) {
        this.messageSource=messageSource;
    }

    @Override
    protected Class<?> getResultViewClass() {
        return null;
    }

    .....
}

```

Step: 6

The date parameters used for the query criteria has to be overridden by the application in order to set the query parameters.

```

@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

    .....

    @Override
    protected String[] getDateParams() throws FinchException {
        return new String[]{
            "tradeDateFrom",
            "tradeDateTo"};
    }

    .....
}

```

The query fetch is taken by the AbstractQueryController.

Step: 7

Finally, in controller class the following methods need to be specified so as to implement the reports (PDF,XLSX) for the query result. To

implement, override the `getReportName()` and `getReportParameters()` methods.

```
@Controller
@RequestMapping(value = "/trd/query")
@SessionAttributes({"commandForm"})
public class TrdQueryController extends AbstractQueryController<TrdQueryCommandForm> implements
MessageSourceAware {

.....
@Override
protected String getReportName(){
 // Specify the report name. This report name with the formaat type of the report is specified in the
jasper-views.xml
 return "trdquery";

}

@Override
protected Map<String, Object> getReportParameters(@ModelAttribute("commandForm")
TrdQueryCommandForm cmdForm) throws FinchException {

Map<String, Object> params = Maps.newHashMap();
Map<String, String> criteriaMap = Maps.newLinkedHashMap();
// Specify the report parameters....
.....
addCriteria(criteriaMap,"Trade Type", cmdForm.getTradeType());
.....
params.put("criteriaMap", criteriaMap);

return params;

}
.....
}
```

Step: 8

To get the support multiple values of a form field, anew method on `AbstractQueryController` has been introduced.To use the functionality the `prepareQueryInput(..)` method is to be overridden, as shown in the following code snippet:

```
protected Map<String, Object> prepareQueryInput(TrdQueryCommandForm queryForm) throws Exception {
 BeanWrapper bw = new BeanWrapperImpl(queryForm);
 Map<String, Object> map = Maps.newHashMap();
 for(PropertyDescriptor pd : bw.getPropertyDescriptors()) {
 map.put(pd.getName(), bw.getPropertyValue(pd.getName()));
 }
 return map;
}
```

Command Form

Command Form class is simple Java POJO which is used to transport data from the client side i.e. JSPX to the server and vice-versa. For query screens, each screen should have a command form class which extends the `QueryCommandForm` class of Finch framework. Apart from the getter-setters for the form properties, the command form class also has a `reset()` method where the contents of the form can be reset. Apart from regular property fields command form class also contains the properties for the sort fields for the query screen.

```
public class TrdQueryCommandForm extends QueryCommandForm {  
....  
    public String tradeType = null;  
  
    public String accountBalanceType = null;  
....  
  
    private String m_sortField1 = null;  
    private String m_sortField2 = null;  
    private String m_sortField3 = null;  
  
    private String m_sortFieldOrder1 = null;  
    private String m_sortFieldOrder2 = null;  
    private String m_sortFieldOrder3 = null;  
  
    public String getTradeType() {  
  
        return tradeType;  
    }  
  
  
    public void setTradeType(String tradeType) {  
  
        this.tradeType = tradeType;  
    }  
  
  
    public String getAccountBalanceType() {  
  
        return accountBalanceType;  
    }  
  
  
    public void setAccountBalanceType(String accountBalanceType ) {  
  
        this.accountBalanceType = accountBalanceType;  
    }  
  
....  
  
    public void reset() {  
  
....  
  
        this.tradeType = null;  
  
        this.accountBalanceType = null;  
  
        m_sortField1 = null;  
        m_sortField2 = null;  
        m_sortField3 = null;  
        m_sortFieldOrder1 = null;  
        m_sortFieldOrder2 = null;  
        m_sortFieldOrder3 = null;
```

```
....  
}  
  
@JsonView(Constants.class)  
public List<LabelValueBean> getSortFieldList() {  
    return sortFieldList;  
}  
  
public void setSortFieldList(List<LabelValueBean> sortFieldList) {  
    this.sortFieldList = sortFieldList;  
}  
  
public String getSortField1() {  
    return m_sortField1;  
}  
  
public void setSortField1(String m_sortField1) {  
    this.m_sortField1 = m_sortField1;  
}  
  
public String getSortField2() {  
    return m_sortField2;  
}  
  
public void setSortField2(String m_sortField2) {  
    this.m_sortField2 = m_sortField2;  
}  
  
public String getSortField3() {  
    return m_sortField3;  
}  
  
public void setSortField3(String m_sortField3) {  
    this.m_sortField3 = m_sortField3;  
}  
  
public String getSortFieldOrder1() {  
    return m_sortFieldOrder1;  
}  
  
public void setSortFieldOrder1(String m_sortFieldOrder1) {  
    this.m_sortFieldOrder1 = m_sortFieldOrder1;  
}  
  
public String getSortFieldOrder2() {  
    return m_sortFieldOrder2;  
}  
  
public void setSortFieldOrder2(String m_sortFieldOrder2) {  
    this.m_sortFieldOrder2 = m_sortFieldOrder2;  
}  
  
public String getSortFieldOrder3() {  
    return m_sortFieldOrder3;  
}  
  
public void setSortFieldOrder3(String m_sortFieldOrder3) {  
    this.m_sortFieldOrder3 = m_sortFieldOrder3;
```

```
}

@JsonView(Constants.class)
public List<String> getSortDetail(){
    List<String> sort = new ArrayList<String>();
    sort.add("sortField1");
    sort.add("sortField2");
    sort.add("sortField3");
    return sort;
}
```

```
}
```

Result View

The Result View class is a POJO which stores the value of the result of the performed query. Sample result view class is shown below:

```
public class TrdQueryResultView implements Serializable{
    private static final long serialVersionUID = 1L;
    private long tradePk;
    private String tradeType;
    private String tradeReferenceNo;
    private String tradeCcy;
    private Date tradeDate;
    private Date valueDate;
    private String buySellFlag;
    private long accountBalanceType;
    private String inventoryAccountNo;
    private String settlementCcy;
    private String instrumentType;
    private String externalReferenceNo;
    private String status;

    public TrdQueryResultView(long tradePk, String tradeType, String tradeReferenceNo,
        String tradeCcy, Date tradeDate, Date valueDate, String buySellFlag, long accountBalanceType, String
        inventoryAccountNo, String settlementCcy, String instrumentType,
        String externalReferenceNo, String status) {
        super();
        this.tradeType = tradeType;
        this.tradeReferenceNo = tradeReferenceNo;
        this.tradeCcy = tradeCcy;
        this.tradeDate = tradeDate;
        this.valueDate = valueDate;
        this.buySellFlag = buySellFlag;
        this.accountBalanceType = accountBalanceType;
        this.inventoryAccountNo = inventoryAccountNo;
        this.settlementCcy = settlementCcy;
        this.instrumentType = instrumentType;
        this.externalReferenceNo = externalReferenceNo;
        this.status = status;
        this.setId(tradeReferenceNo);
        this.tradePk = tradePk;
    }

    public String getTradeType() {
        return tradeType;
    }

    public void setTradeType(String tradeType) {
        this.tradeType = tradeType;
    }

    public String getTradeReferenceNo() {
        return tradeReferenceNo;
    }

    public void setTradeReferenceNo(String tradeReferenceNo) {
        this.tradeReferenceNo = tradeReferenceNo;
    }
}
```

```
}

public String getTradeCcy() {
    return tradeCcy;
}
public void setTradeCcy(String tradeCcy) {
    this.tradeCcy = tradeCcy;
}
public Date getTradeDate() {
    return tradeDate;
}
public void setTradeDate(Date tradeDate) {
    this.tradeDate = tradeDate;
}

public String getBuySellFlag() {
    return buySellFlag;
}
public void setBuySellFlag(String buySellFlag) {
    this.buySellFlag = buySellFlag;
}
...
@Beta
// for grid rendering
private String id;
public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public long getTradePk() {
    return tradePk;
}
public void setTradePk(long tradePk) {
    this.tradePk = tradePk;
}
```

```
}
```

Note: id field in the result view class needs to be set for the pagination to work. The value of the id field must be unique in nature.

Pagination

For pagination implementation, please refer to [Pagination](#)

Service

As mentioned in the Controller section, to get the query result, access the database through different layers. Access the service layer which in turn access the repository layer. This section describes what needs to be created in the service layer and repository layer to get the results for the query.

Step 1:

In the controller, overwrite the getQueryService() method, as shown below:

```
@Override  
protected TradeService getQueryService() {  
    if (tradeService == null) {  
        tradeService = Operation.getInstance().getReference(TradeService.class);  
    }  
    return tradeService;  
}
```

Step 2:

Create a service interface by extending the QuerySupport (Deprecated) or QuerySupportEx of finch framework, as shown below:

```
public interface TradeService extends QuerySupportEx<TrdQueryResultView> {  
    ....  
}
```

Note: The getQueryResult() method is defined in the QuerySupportEx interface

Step 3:

Create the corresponding implementation class for the service, as shown below:

```

@Service
public class TradeServiceImpl implements TradeService{
.....
@.Autowired
private TradeRepository trdRepo;

@Override
@Transactional(value = "sample-tm", readOnly = true)
public List<TrdQueryResultView> getQueryResult(Map<String, Object> queryInput,int startIndex,int maxResult)
throws FinchException {
try {
TrdQueryCriteriaBuilder criteriaBuilder = new TrdQueryCriteriaBuilder(queryInput);
JPAQueryExecutionParams<TrdQueryResultView> queryExParams = new
JPAQueryExecutionParams.Builder<TrdQueryResultView>()
    .withQueryCriteria(criteriaBuilder.build(trdRepo.getEntityManager()))
    .withStartIndex(startIndex)
    .withMaxResult(maxResult).build();
return trdRepo.getQueryResult(queryExParams);
} catch (Exception ex) {
log.error("Failed to execute Trade Query", ex);
throw new TrdException(ex);
}
}
.....
}

```

While creating the implementation class, ensure about the followings:

- Use the @Service annotation to define it as Service class.
- Autowire the repository bean
- Specify the transactional manager using the @Transactional annotation. Do note the value for the transaction manager is either global-tm or sample-tm. In case of INF related objects, specify it as global-tm and in case of sample app objects, specify it as sample-tm.
- Use the appropriate criteria builder class object to be used with JPAQueryExecutionParams

Step 4:

Create the appropriate criteria builder class by implementing the QueryCriteriaBuilder interface of the finch framework.

Step 5:

Create the repository class, as shown below:

```

@Repository
public class TradeRepository extends RepositoryBase<Trade, Long> {

private static final Logger Log = LoggerFactory.getLogger(TradeRepository.class);

@PersistenceContext(unitName="sample-em")
private EntityManager em;
@Override
public EntityManager getEntityManager() {
return em;
}
...
}

```

While creating the repository class, ensure about the followings:

- The class must be annotated as @Repository
- While defining the EntityManager we must use the @PersistenceContext annotation and specify the context. It can be either global-em or sample-em. global-em is used for INF entities and sample-em is used for sample app entities.

Criteria Builder

The criteria builder class is one of the main class where we write the actual query for retrieving the data. Write the code for getting the results in the build() method using JPA API, as shown in the following code snippet.

The build method under QueryCriteriaBuilder has been deprecated. Use instead AbstractQueryCriteriaBuilder and its related API's. For detail implementation, please refer to TrdQueryCriteriaBuilder.

```
public class TrdQueryCriteriaBuilder extends AbstractQueryCriteriaBuilder<TrdQueryResultView, Trade> {
    private static final Logger LOGGER = LoggerFactory.getLogger(TrdQueryCriteriaBuilder.class);

    public TrdQueryCriteriaBuilder(Map<String, Object> queryInput) {
        super(queryInput);
    }

    @Override
    protected void processSelectClause(CriteriaBuilder builder, CriteriaQuery<TrdQueryResultView> cq,
                                       Root<Trade> trdRoot) {
        cq.multiselect(trdRoot.get("tradePk"),
                       trdRoot.get("tradeType"),
                       trdRoot.get("referenceNo"),
                       trdRoot.get("tradeCurrency").get("securityId"),
                       trdRoot.get("tradeDate"),
                       ...
                       trdRoot.get("status"));
    }

    @Override
    protected List<Predicate> getPredicateList(CriteriaBuilder builder, CriteriaQuery<?> cq, Root<Trade> trdRoot)
            throws FinchException {
        AccountService accountService = Operation.getInstance().getReference(AccountService.class);
        InstrumentService instrumentService = Operation.getInstance().getReference(InstrumentService.class);
        FinInstRoleService firService = Operation.getInstance().getReference(FinInstRoleService.class);
        List<Predicate> predList = Lists.newArrayList();

        // Criteria
        processQueryStringField(predList, builder, trdRoot, "tradeType", "tradeType");
        processQueryStringField(predList, builder, trdRoot, "referenceNo", "referenceNo");
        processQueryMultipleStringValueField(predList, builder, trdRoot, "tradeCcy", new String[]{"tradeCurrency",
                "securityId"});
        processQueryStringField(predList, builder, trdRoot, "buySellFlag", "buySellFlag");
        processQueryStringField(predList, builder, trdRoot, "accountBalanceType", "accountBalanceType");
        // inventory account no
        if (!Strings.isNullOrEmpty(getInputParam("inventoryAccountNo", String.class))) {
            Pair<String, String> codePair = parseCode(getInputParam("inventoryAccountNo",
                    String.class).trim().toUpperCase());
            Account account;
            try {
                account = accountService.findAccount(codePair.left(),
                        codePair.right());
            } catch (RefException e) {
                LOGGER.error("", e);
                throw new TrdException("error.field.invalid", "Inventory Account");
            }
        }
    }
}
```

```

}

predList.add(builder.equal(trdRoot.get("inventoryAccount").get("accountPk"), account.getAccountPk()));

}

// account no
if (!Strings.isNullOrEmpty(getInputParam("accountNo", String.class))) {
    Pair<String, String> codePair = parseCode(getInputParam("accountNo", String.class).trim().toUpperCase());
    Account account;
    try {
        account = accountService.findAccount(codePair.left(), codePair.right());
    } catch (RefException e) {
        LOGGER.error("", e);
        throw new TrdException("error.field.invalid", "Account");
    }
    predList.add(builder.equal(trdRoot.get("account").get("accountPk"), account.getAccountPk()));
}
}

...
// Trade Date From
try {
    processQueryDateField(predList, builder, trdRoot, "tradeDateFrom", "tradeDate", DateComparator.GE);
} catch (Exception e) {
    LOGGER.error("", e);
    throw new TrdException("error.field.invalid", "Trade Date From");
}

// Trade Date To
try {
    processQueryDateField(predList, builder, trdRoot, "tradeDateTo", "tradeDate", DateComparator.LE);
} catch (Exception e) {
    LOGGER.error("", e);
    throw new TrdException("error.field.invalid", "Trade Date To");
}

...

processQueryStringField(predList, builder, trdRoot, "status", "status");
Subquery<Short> subQry = cq.subquery(Short.class);
Root<Trade> subRoot = subQry.from(Trade.class);
Path<Short> verNoExp = subRoot.get("versionNo");
subQry.select(builder.max(verNoExp));
Path<Short> varExp1 = trdRoot.get("versionNo");
subQry.where(builder.equal(subRoot.get("referenceNo"), trdRoot.get("referenceNo")));
predList.add(builder.equal(varExp1, subQry));
return predList;
}

@Override
protected Map<String, Path<?>> preparePathMap(Root<Trade> trdRoot) {
    Map<String, Path<?>> pathMap = new HashMap<>();
    pathMap.put("tradePk", trdRoot.get("tradePk"));
    pathMap.put("tradeType", trdRoot.get("tradeType"));
    pathMap.put("referenceNo", trdRoot.get("referenceNo"));
    pathMap.put("tradeCurrency", trdRoot.get("tradeCurrency").get("securityId"));
    pathMap.put("tradeDate", trdRoot.get("tradeDate"));
    pathMap.put("valueDate", trdRoot.get("valueDate"));
    pathMap.put("buySellFlag", trdRoot.get("buySellFlag"));
    pathMap.put("accountBalanceType", trdRoot.get("accountBalanceType"));
    pathMap.put("inventoryAccount", trdRoot.get("inventoryAccount").get("accountNo"));
    pathMap.put("settlementCurrency", trdRoot.get("settlementCurrency").get("securityId"));
    pathMap.put("instrumentType", trdRoot.get("instrument").get("instrumentType"));
    return pathMap;
}

```

```
}
```

```
private Pair<String, String> parseCode(String code) {
    String[] parts = code.split("/");
    if(parts.length ==1) {
        return Pair.pair(code, DEFAULT_CODE_TYPE);
    }
    return Pair.pair(parts[0], parts[1]);
}

@Override
protected Class<TrdQueryResultView> getResultViewClass() {
    return TrdQueryResultView.class;
}

@Override
protected Class<Trade> getRootEntityClass() {
    return Trade.class;
}
```

```
}
```

For Consolidated Action (if applicable)

For consolidated action, create a javascript file containing the consolidated parameters, as shown below:

sample-trd-query-result.js

```
var consolidateParmas = function(e){
    var row = $(e.target).attr('row');
    var gridData = $('.finch-grid').data("gridInstance").getData().getItems();
    var parmas = '&pk=' + gridData[row]['tradePk'];
    $('#a.consolidateActLink').data("params", parmas);
};

// Query Result from Dashboard feed does not show in zebra colour, added for background overlapping
$(document).ready(function(){
    if($('.ui-dialog-buttonpane').length==0){
        $('.ui-resizable').children('.ui-widget-content').css('background','none');
    }
});
```

The result page must contain the button configuration for consolidated action.

Consolidated action can applied across each records in a grid or can be applied on a selected record.

For Consolidated Action on All Records

tradeQueryResult.jspx

```
<jsp:element name="script">
.....
    $('#queryResultForm .finch-grid').finchgrid(grid_result_data, grid_result_columns, grid_result_settings);
</jsp:body>
</jsp:element>

<div id="consolidateAct" style="display:none;">
    <c:forEach items="${navAction}" var="record">
        <a href="${record.actionUrl}">
            <c:choose>
                <c:when test="${fn:containsIgnoreCase(record.actionId, 'TRDTAM')}"></c:when>
                <c:when test="${fn:containsIgnoreCase(record.actionId, 'TRDTCX')}"></c:when>
                <c:otherwise></c:otherwise>
            </c:choose>
        </a>
    </c:forEach>
```

For Consolidated Action on Selected Record

A setting attribute consolidationActionCallback must be defined as described below. The attribute takes the each row as a parameter and returns false for those which do not have consolidated action.

The following example shows consolidated actions for records with trade status = 'NORMAL'.

```
tradeQueryResult.jspx

// other stuffs
var grid_result_settings = {
  enableToolbar:true,
  consolidateActionFlag:true,
  buttons:[
    // do stuffs
  ],
  pagingInfo:{
    //do stuffs
  },
  urls:{
    //do stuffs
  },
  consolidationActionCallback: function(rec) {
    return rec.status !== 'CANCEL'; // returns only false
  }
};
var row_id = 0;
var rec = {};
<c:forEach items="${value}" var="dl">
  rec = {};
  row_id+=1;
  rec.id = "finch_" + row_id;
  rec.status = "<c:out value=\"${dl.status}\" />";
  grid_result_data.push(rec);
</c:forEach>
$('#queryResultForm .finch-grid').finchgrid(grid_result_data, grid_result_columns, grid_result_settings);
```

For navigating back to the query result after performing the consolidated action, the following needs to be incorporated in the exit URI of the wizard. For example, In trade consolidated action- cancel and amend:

```
tradeQuery.jspx

if(tradeMode == "cancel" || tradeMode == "amend")
  $finch$wizard.apply({exitUri: FINCH.context.path + '/trd/query/exitURI'});
else
  $finch$wizard.apply({exitUri: FINCH.context.path});
```

The exit URI for navigating to the query result must be the base url of the query controller + "exitURI".

Screen Data Setup

This section describes what all data setup needs to be performed in the database for query screen (both criteria and result) to work properly.

Step 1:

Screen data setup for the criteria screen and result screen needs to be setup in the INF_UI_SCREEN table. The following database fields

need to be populated:

- SCREEN_KEY
- SCREEN_ID
- SCREEN_NAME
- SHORT_NAME
- DESCRIPTION
- VERSION_NUMBER

Step 2:

Once the data setup for the criteria screen and result screen is done, then update the record for the criteria screen and set the value for the column NEXT_SCREEN_PK to with the SCREEN_KEY value of the result screen.

The recordset is displayed below:

	SCREEN_KEY	SCREEN_ID	SCREEN_NAME	SHORT_NAME	DESCRIPTION	VERSION_NUMBER	NEXT_SCREEN_PK	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE	UPDATED_BY	UPDATE_DATE	CC_CHE
1	10011TRDRS	Trade Query Result	Trade Query Result	Trade Query Result		1	(null)	09-10-13	09-10-13	FINCH-SAMPLE	09-10-13	FINCH-SAMPLE	09-10-13	
2	10002TRDQR	Trade Query	Trade Query	Trade Query		1	10011	09-10-13	09-10-13	FINCH-SAMPLE	09-10-13	FINCH-SAMPLE	09-10-13	

Presentation Layer

In finch, the presentation layer is created using JSPX and jQuery. For Query UI, create two JSPX files - one for the criteria screen and one for the result screen. To see how the JSPX files are created, refer to the following JSPX files in the code base.

- For criteria page – ../src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx
- For result page – ../src/main/web/WEB-INF/views/trd/tradeQueryResult.jspx

Query

finch provides following hook:

- FINCH\$Query\$Reset\$Hook : Using this hook application can do any additional activity required during form RESET. An example is shown below:

tradeQueryCriteria.jspx

```
var FINCH$Query$Reset$Hook = function() {
// custom implementation
console.log('Hello ,this is reset query hook.....');
}
```

- FINCH\$Query\$Submit\$Hook : Using this hook application can do any additional activity required after Submit button press and before sending the request to server. An example is shown below:

userQueryCriteria.jspx

```
var FINCH$Query$Submit$Hook = function() {
console.log('Hello ,this is Submit query hook.....');
}
```

If sort criteria is implemented, then duplicateSortFieldInfoMsg must be passed as an argument. For example,

userQueryCriteria.jspx

```
var FINCH$Query$Submit$Hook = function(duplicateSortFieldInfoMsg) {
//do other stuffs
if(duplicateSortFieldInfoMsg != "") {
validationMessages.push(duplicateSortFieldInfoMsg);
}
//do other stuffs
```

The FINCH\$Query\$Submit\$Hook must return boolean value. If value returned is false, form is not submitted. If no value is returned, true is value is returned by default.

- FINCH\$Save\$Query\$Hook : Using this hook, application can perform any additional activity before saving the query criteria. An example is shown below:

tradeQueryCriteria

```
var FINCH$Save$Query$Hook = function() {
console.log("Do your validation before save query and post notice.....");
return true; //return true for success and false for failure.
}
```

- FINCH\$PreSave\$Query\$Hook : Using this hook, application can perform custom validation before opening the saved query. It must return false to prevent opening the save query prompt. If validation is successful, return true An example is shown below:

tradeQueryCriteria.jspx

```
var FINCH$PreSave$Query$Hook = function() {
//FINCH.postNotice(FINCH.notice.type.error,'Error in opening save query');
return true;
}
```

Validation during save query will be performed using FINCH.preValidateSaveQuery flag. If this flag is true, FINCH\$PreSave\$Query\$Hook will be called if implemented. If this value is false, FINCH\$Save\$Query\$Hook will be called, if implemented.

- FINCH\$Error\$Message\$Hook : Using this hook, application can customize the persistence of the error message that is returned from the server. An example is shown below:
- FINCH\$Error\$Message\$Hook : Using this hook, application can customize the persistence of the error message that is returned from the server. An example is shown below:

tradeQueryCriteria.jspx

```
var FINCH$Error$Message$Hook = function(msg) {
FINCH.postNotice(FINCH.notice.type.error,msg,true);
}
```

- FINCH\$Query\$FormItem\$Remove\$Hook : Using this hook application can decide whether form item that user is about to delete is allowed or not. If this hook returns 'true' then finch will take action to delete form item, but if this hook returns 'false' then finch will not delete the form item. An example is shown below

tradeQueryCriteria.jspx

```
var FINCH$Query$FormItem$Remove$Hook = function(ev){  
    //All the form item is allowed to be removed by user.  
    return true;  
}
```

Save Query

In order to get the date fields date as per the date format specified in the user's preference, the following activity needs to be performed. In the command form - for the getter method of the display date, use the `DisplayDateType` annotation for the method. An example is shown below:

```
@DisplayDateType  
public void getTradeDateFrom() {  
...  
}
```

Note: The setter method name should be same as the getter method name other than 'get'.

Example:

```
@DisplayDateType  
public String getTradeDateFrom() {  
...  
}
```

Setter method

```
public void setTradeDateFrom(String tradeDateFrom) {  
...  
}
```

Pre Fetching Data

In finch, while performing query, data (query result) can be pre-fetched i.e. 'n' of query result pages can be pre fetched so that during pagination, database call will not be made for each pagination request. Database request will be made only when 'n' no pages of data has been traversed. This pre-fetching of data will only work with the new pagination model i.e. where `getQueryService()` method is implemented in the controller. For details see the Service section of this document. This pre fetching of data would work in both normal query pages and wizard based query pages. Do note, that once data is pre fetched, the data is not re-loaded.

Out of the box, finch pre-fetches data for 5 pages and on every 3rd page next set of data is loaded. However this configuration can be configured by the application. The configuration needs to be done in the `keys.properties` and `app-pref.xml` file.

keys.properties

```
pref.key.application.prefetchpagesize=PRE_FETCH_PAGE_SIZE  
pref.key.application.dataloadpageinterval=DATA_LOAD_PAGE_INTERVAL
```

```

app-pref.xml

<ROOT>
...
<PRE_FETCH_PAGE_SIZE>5</PRE_FETCH_PAGE_SIZE>
<DATA_LOAD_PAGE_INTERVAL>3</DATA_LOAD_PAGE_INTERVAL>
...
</ROOT>

```

Note: The tag <PRE_FETCH_PAGE_SIZE> would contain the value for the no of pages to be pre-fetched and <DATA_LOAD_PAGE_INTERVAL> tag would contain the value for loading the next set of data after every nth page.

Also when a user changes the page size value from the preference when the user is in the result screen, the new preference value for the page size would be honored only when the user re submits the query.

- FINCH\$Error\$Message\$Hook : Using this hook, application can customize the persistence of the error message that is returned from the server. An example is shown below:

3.4.6.2.3.1 Personalized Query Criteria

Introduction

finch framework allows the contents (fields) of the query criteria screen to be personalized. User can selectively remove or add fields in the query criteria screen. User also can rearrange the position of the fields on the screen. For all this functionality, HTML code needs to be specified in the query criteria JSPX file.

Configuration

To remove/add fields from the query criteria screen following HTML code needs to be specified while designing the query criteria page.

```

<div>
  <form:label path="fromDate"><spring:message code="access.log.fromto.date.label"
  htmlEscape="false"/></form:label>
  <span><form:input value="" path="fromDate" onchange="checkDate(this);"/></span>
  <span><form:input value="" path="toDate" onchange="checkDate(this);"/></span>
  <span class="remove" title="Remove this field" style="display: none;"><spring:message text=""
  htmlEscape="false"/></span>
  <div><spring:message text="" htmlEscape="false"/></div>
</div>
.....
<div style="display:none;" >
  <!-- Deleted Item goes here -->
  <input type="button" value="Restore All" />
  <div><spring:message text="" htmlEscape="false"/></div>
</div>

```

Usage

For usage of query screen personalization please refer to the [end user guide](#).

3.4.6.2.3.2 Pagination Support

Introduction

In finch, pagination support for query results is in-built into the framework. Pagination in finch can be achieved in the following ways:

- Fetch all data at once – This is the old mechanism to fetch all the data in one shot and bind all of them in the session. This implementation has an disadvantage. If the query result has huge data, the entire data gets stored in the session. As the result the session gets overloaded. To prevent such overhead, the pagination model has been enhanced with fetching data per request as mentioned below.
- Fetch data per request – This technique is used to fetch data based on the upper limit and the starting index. The upper limit defines the total number of records to be fetched per request. The start index defines the position from where the first record will be fetched. The index keeps updating on each request.

Implement Pagination

Fetch all data at once

Step 1:

The service class should extend `QuerySupportEx`, as shown in the following code snippet:

```
public interface EmployeeService extends QuerySupportEx<EmployeeQueryResultView>
```

Step 2:

In controller, you must override `getQueryResult (CommandForm form)`, as shown in the following code snippet:

```
EmployeeQueryController.java

@SuppressWarnings("unchecked")
@Override
protected List<EmployeeQueryResultView> getQueryResult(
    EmployeeQueryCommandForm queryForm) throws FinchException {
    DateExpressionEvaluator.eval(queryForm,
        "INF",
        "startDateFromStr",
        "startDateToStr",
        "employeeOpenDateFromStr",
        "employeeOpenDateToStr");

    try {
        return getService(EmployeeService.class).getQueryResult(BeanUtils.describe(queryForm), 0, 0);
    } catch (IllegalAccessException | InvocationTargetException
        | NoSuchMethodException e) {
        log.error("Failed to execute the query");
        throw new InfException(e);
    }
}
```

Fetch data per request

Step 1:

The service class should extend `QuerySupportEx`, as shown in the following code snippet:

```
public interface TradeService extends QuerySupportEx<TrdQueryResultView>
```

Step 2:

In controller, you must override getQueryService(), as shown in the following code snippet:

```
@Override  
protected TradeService getQueryService() {  
    if (tradeService == null) {  
        tradeService = Operation.getInstance().getReference(TradeService.class);  
    }  
    return tradeService;  
}
```

Step 3:

In controller, getDateParams() must be overridden for providing any date parameters in query, as shown in the following code snippet:

TrdQueryController.java

```
@Override  
protected String[] getDateParams() throws FinchException {  
    return new String[]{  
        "tradeDateFrom",  
        "tradeDateTo",  
        "valueDateFrom",  
        "valueDateTo",  
        "creationDateFrom",  
        "creationDateTo"};  
}
```

Step 4:

In service implementation, override the count method, as shown in the following code snippet:

TradeServiceImpl.java

```
@Override
public long count(Map<String, Object> queryInput) throws FinchException {
    try {
        Identity principal = null;
        if (Operation.getInstance() != null && Operation.getInstance().getContext() != null) {
            principal = ((OperationScope) Operation.getInstance().getContext()).getCallerIdentity();
        }
        TrdQueryCriteriaBuilder criteriaBuilder = new TrdQueryCriteriaBuilder(queryInput);
        JPAQueryExecutionParams<Long> queryExParams =
            new JPAQueryExecutionParams.Builder<Long>()
                .withQueryCriteria(criteriaBuilder.buildCountQuery(trdRepo.getEntityManager()))
                .withMaxResult(Integer.valueOf(principal.getPreferenceBean()
                    .getPreference(PreferenceConstants.MAX_UI_RECORDS_FETCHED))).build();
        return trdRepo.getCount(queryExParams);
    } catch (Exception ex) {
        log.error("Failed to execute Trade Query", ex);
        Throwables.propagateIfInstanceOf(ex, FinchException.class);
        throw new TrdException("error.query.failed", ex);
    }
}
```

How to Use

Please refer to the [end user guide](#).

3.4.6.2.3.3 Saved Template

Introduction

In finch framework, entry screen data can be saved as templates and the same can be used for the data entry at later stage. The saved templates can be created in the following two modes:

- My templates – This type of template is only available to the user who has created it. The owner only can use this type of template, not the other users of the same enterprise.
- Shared templates – All the users of the enterprise along with the template owner can use this type of template.

The saved template feature works on the model of the Saved query. The command form object of the screen is serialized in JSON format and stored in the database.

Implementation

To implement save template feature in a entry screen, the developer should follow the steps mentioned below:

1. In the default constructor of the entry controller class, the command form type associated with the controller need to be specified, as shown in the following code snippet.

```
public TradeEntryController() {
    commandFormType = TradeEntryCommandForm.class;
}
```

2. In the constructor of the command form associated with the entry controller where the wizard page setup is done, specify the enumeration TEMPLATESAVE for edit mode, as shown in the following code snippet.

```
Map<FinchWizardMode, FinchWizardButton[]> buttons = ImmutableMap.of(EDIT, new
FinchWizardButton[] {PREVIOUS, NEXT, RESET, SUBMIT, TEMPLATESAVE},
USER_CONFIRMATION, new FinchWizardButton[] {BACK, CONFIRM},
SYSTEM_CONFIRMATION, new FinchWizardButton[] {OK});
```

- In the command form, it may be required that certain method of the command form is not needed during the serialization of the command form (i.e. converting the command form into JSON). In such case use the `@JsonIgnore` annotation in the command form method, as shown in the following code snippet.

```
@JsonIgnore
public Collection<TradeRrDTO> getRrAmounts() {
if(rrMap !=null){
Collection<TradeRrDTO> values = rrMap.values();
return values;
}
return null;
}
```

One new flag called `isSaveTemplate():boolean` has been introduced in the `FinchAbstractWizardCommandForm` class. With the help of this method developer can figure out whether the entry screen displayed is from save template or entry screen displayed by invoking it from the menu. This is required in case of initialization method of the controller, where sometimes reset method is called. But in case of entry screen opened from saved template mode, the `reset()` method should not be called.

Client Side Hook

For Save Template feature, a client side hook is provided. The developer can perform additional activity when save template is being created. The hook needs to be configured in the JSPX page, as shown in the following code snippet.

- `FINCH$PreSave$Template$Hook`: This hook is called when the SAVE TEMPLATE button is clicked. Application client can use this hook in their jspx and perform proper validation. This method must return false in order to prevent opening of the prompt. For example,

```
tradeGeneralEntry.jspx

var FINCH$PreSave$Template$Hook = function() {
if ((typeof isValidDate !== 'undefined' && !isValidDate($('#commandForm')))) {
return false;
}
};
```

- `FINCH$Save$Template$Hook`: This hook is called when either option from the share template is selected but before the call to save the template is made at server end. Application client can use this hook in their jspx and perform proper validation. This method must return false in order to prevent saving the template. For example,

```
tradeGeneralEntry.jspx

var FINCH$Save$Template$Hook = function() {
if ((typeof isValidDate !== 'undefined' && !isValidDate($('#commandForm')))) {
return false;
}
};
```

Entry Screens provided by finch out-of-the box does not provide any validation fro the above mentioned hooks

Date Field Formatting as Per Preference Date Format

To get the date fields date in the identical date format, as specified in the user's preference – in the command form for the getter method of the display date, use the `DisplayDateType` annotation for the method.

```
@DisplayDateType  
public void getTradeDateFrom() {  
...  
}
```

Note: The setter method name should be same as the getter method name other than 'get'.

Example:

```
@DisplayDateType  
public String getTradeDateFrom() {  
...  
}
```

Setter method
public void setTradeDateFrom(String tradeDateFrom) {
...
}

How to Use?

Please refer to [Save Template](#) manual.

Data Model

Data model for Save Template has been described in the [Data Model](#) section. Database tables used for Save Template are:

- INF_SAVED_TEMPLATE
- DBD_CUSTOM_SAVED_TEMPLATE

3.4.6.2.4 Reference Data Popup

Introduction

finch Popup framework consists of the followings:

- finch-popQuery-Form.js - Contains the logic of firing the right URL to fetch the right popup page. It also holds the logic for submit, reset and re-query.
- Module specific logics

Develop Popup Query Page

Brief description of how a popup query page can be developed has been discussed below.

Modification in Query Page

On the query page where the popup page to be integrated the following modification is required:

In the target div block(say for Inventory account) a <div> element with class "popupBtn" has been introduced. Available properties of the input element are:

- type : button
- class : popupBtnlco
- tgt : Id of the form:input element where the selected data from popup page to be set.
- popType : Popup type (for query or result).
- popFor : Popup type (for Account/Inventory Account/Security etc).
- Any other dependent configuration for specific popup pages.
- also can provide option to load TreeView (i.e. isTreeView: true) and DateValidation (i.e. isDateValidation: true). By default value for this property is false.

Example:

Adding account popup in Trade general entry page.

```
form:label path="commandForm.tradeDTO.accountNo" class="required">><spring:message  
code="trd.tradeentryaction.label.accountno" htmlEscape="false"/></form:label>  
  
<span><form:input id="accountNo" path="commandForm.tradeDTO.accountNo" class="textBox"/></span>  
  
<div class="popupBtn">  
  
    <input type="button" class="popupBtnlco amendReadOnlyPopup" tgt="accountNo" popType="query"  
    popFor="cpAccount" actTypeContext="T|B" actCPTYPEContext="CLIENT|BROKER|INTERNAL"  
    actStatusContext="OPEN" value="" />  
  
</div>  
  
<div class="clear"><spring:message text="" htmlEscape="false"/></div>  
  
</div>
```

Flow of Control in finch-popQuery-Form.js

The finch-popQuery-form.js is responsible for all the following tasks:

- Determine popup object from popType.
- Determine target field using tgt and the selected data from popup page to be set in this target field.
- If required, populate the dependent parameter from the given custom properties popFor of popup button.
- Send an ajax request to load the requested popup in a dialog.
- Contains the logic for submit,reset and requery handling.

To build a new popup we need to add following lines of code in this JS.

```

if(popType== givenPopType){

    title = FINCH.title.popupTitle;
    url = popupUrl;
    var dependentParam = prepareXXXDependentParams(btn,popUpType)
    extUrlParams = dependentParam;
    multiSelectPopup = true;
    multiSelectHandler = customHandler /* Only needed if the multiSelectPopup parameter is true.
        There exists a default handler if none specified which appends all the data of the target column in the
        rows selected in a comma-separated manner*/
    var popSize = {};
    $.extend(defaultSize,popSize);
}

function prepareXXXDependentParams(btnObj,popType)
{
// Some code related to populate dependent parameter &nbsp;&nbsp; return dependent parameter;
}

function customHandler(settings, data)
{
// Here settings will contain the all the settings used in the creation of the popup including the target field,
// and the data will be an array of object, where each object will contain data in the rows selected
}

```

Example:

```

if(popUpType=="invAccount"){
    title = FINCH.title.accPopup;
    url = "/ref/account/popup/query";
    var dependentParam = prepareAccDependentParams(btn,popUpType);
    extUrlParams = dependentParam;
    //sample object to override default dialog size
    var popSize = {};
    $.extend(defaultSize,popSize);
}

```

```

function prepareAccDependentParams(btnObj,popFor) {
    var dStr ="";
    var flag = true;
    if(popFor=='invAccount')
        dArray = new Array('invActTypeContext','invCPTYPEContext','actStatusContext');
    else if(popFor=='cpAccount')
        dArray = new Array('actTypeContext','actCPTYPEContext','actStatusContext');
    else
        dArray = new Array();
    for (var k=0;k<dArray.length ;k++ )
    {
        var objVal=btnObj.attr(dArray[k]);
        if(objVal == undefined){
        }else{
            if(flag){
                dStr+="?dependents["+dArray[k]+"]="+objVal;
                flag = false;
            }else{
                dStr+="&dependents["+dArray[k]+"]="+objVal;
            }
        }
    }
    return dStr;
}

```

Another two files we need to build for each popup and they are:

- Popup query criteria jsp – contains form input item.
- Popup Query result jsp – contains grid result columns

Example:

In the accountPopUpQueryCriteria.jspx file:

```

<div class="popFormItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:spring="http://www.springframework.org/tags"
    xmlns:form="http://www.springframework.org/tags/form"
    xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">

    <jsp:directive.page contentType="text/html;charset=UTF-8" />

    <jsp:output omit-xml-declaration="yes" />

    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:attribute name="src">
            <c:url value="/scripts/ref/finch-counterparty-code.js"/>
        </jsp:attribute>
    </jsp:element>

```

```

</jsp:attribute>

<jsp:body></jsp:body>

</jsp:element>

<div class="popFormItem">

    <form:label path="AccountNo"><spring:message code="trade.label.account.no"
htmlEscape="false"/></form:label>

    <span><form:input value="" path="AccountNo" class="textBox popTgt" /></span>

    <div class="clear"><spring:message text="" htmlEscape="false"/></div>

</div>

<div class="popFormItem">

    <form:label path="defaultShortName"><spring:message code="ref.label.account.nameshort"
htmlEscape="false"/></form:label>

    <span><form:input value="" path="defaultShortName" class="textBox" /></span>

    <div class="clear"><spring:message text="" htmlEscape="false"/></div>

</div>

<div class="popFormItem twoCols">

    <form:label path="counterPartyDetailType"><spring:message code="ref.label.counterparty.type"
htmlEscape="false"/></form:label>

    <span>

        <form:select path="counterPartyDetailType" class="dropdowninput counterPartyCodeType" >
            <form:options items="$
{commandForm.counterPartyDetailTypeMap}
"/>

        </form:select>

        <form:input value="" path="counterPartyCode" class="textBox smallInput marginLeft"
disabled="true"/>

    </span>

    <div class="clear"><spring:message text="" htmlEscape="false"/></div>


```

```
</div>  
</div>
```

In the accountPopUpQueryResult.jspx file:

```
<div xmlns:c="http://java.sun.com/jsp/jstl/core"  
      xmlns:fn="http://java.sun.com/jsp/jstl/functions"  
      xmlns:spring="http://www.springframework.org/tags"  
      xmlns:util="urn:jsptagdir:/WEB-INF/tags/util"  
      xmlns:jspx="http://java.sun.com/JSP/Page" version="2.0">  
  
<jsp:directive.page contentType="text/html;charset=UTF-8" />  
<jsp:output omit-xml-declaration="yes" />  
  
<jsp:element name="script">  
    <jsp:attribute name="type">text/javascript</jsp:attribute>  
    <jsp:attribute name="src">  
        <c:url value="/scripts/inf/finch-formatters.js" />  
    </jsp:attribute>  
    <jsp:body></jsp:body>  
</jsp:element>  
  
<jsp:element name="script">  
    <jsp:attribute name="type">text/javascript</jsp:attribute>  
  
    <jsp:body>  
        var isNext = <c:out value="${isNext}" />;  
        var grid_result_data = [];  
        var grid_result_columns = [  
            {name:"",field:"header",id:"header",width:50,formatter:Slick.Formatters.SelectPopQueryFormater},  
            {name:<spring:message code="ref.label.account.no"  
                htmlEscape="false"/>,targetColumn:true,field:"accountNo",id:"accountNo", width:100,sortable:true},  
            {name:<spring:message code="ref.label.account.nameshort"  
                htmlEscape="false"/>,field:"accountName",id:"accountName", width:160,sortable:true},  
            {name:<spring:message code="ref.label.account.customer.name"  
                htmlEscape="false"/>,field:"customerName",id:"customerName", width:100,sortable:true},  
            {name:<spring:message code="ref.account.label.acstatus" htmlEscape="false"/>, field:"status",id:"status",  
                width:100,sortable:true,formatter:Slick.Formatters.CancelRecordFormatter}  
        ];  
        var grid_result_settings = {  
            enableToolbar:true,  
            forceFitColumns:true,  
            multiSelectPopup:true,  
            pagingInfo:{  
                isNext : isNext  
            },  
            urls:{  
                nextPage : '/ref/account/popup/query/result.json?fetch=next',  
                prevPage : '/ref/account/popup/query/result.json?fetch=previous'  
            }  
        };  
        var row_id = 0;  
        var rec = {};
```

```
<c:forEach items="${value}" var="dl">
    rec = {};
    row_id+=1;
    rec.id = "igv_" + row_id;
    rec.accountNo = "<c:out value='${dl.accountNo}'/>";
    rec.accountName = "<c:out value='${dl.accountName}'/>";
    rec.customerName = "<c:out value='${dl.customerName}'/>";
    rec.status = "<c:out value='${dl.status}'/>";
    grid_result_data.push(rec);
</c:forEach>
</jsp:body>
</jsp:element>
```

```
</div>
```

- targetColumn:true signifies that value of this column is selected when user clicks on Select button
- multiSelect:true signifies that the popup will have the option to select more than one row

Model Specific Logic

Now we have to write a one Controller and Command Form. The Controller should extend com.nrft.fin.ch.inf.web.controller.AbstractQueryController and Command Form should extend com.nrft.fin.ch.inf.web.QueryCommandForm.

Controller should have @Controller at class level. Controller should have @SessionAttributes at class level.

All URL(s) should be unique in a Controller. For example, AccountPopQueryController.java

```
...
@Controller
@RequestMapping(value = "/ref/account/popup/query")
@SessionAttributes({"commandForm"})
public class AccountPopUpQueryController extends
AbstractQueryController<AccountPopUpQueryCommandForm>{

    private static final Logger log = LoggerFactory.getLogger(AccountPopUpQueryController.class);
    Identity principal = null;

    public AccountPopUpQueryController() {
        commandFormType = AccountPopUpQueryCommandForm.class;
    }

    @Override
    protected void doInit(AccountPopUpQueryCommandForm form, FinchModelMap FinchModelMap) throws
    FinchException {
        HttpServletRequest request = ((ServletRequestAttributes)
        RequestContextHolder.getRequestAttributes()).getRequest();
        try{
            initQuery(FinchModelMap,form);
            if(request != null){
                form.setBaseUrl(StringUtils.removeEnd(request.getServletPath(), "/init"));
            }
        }catch(Exception e){
            log.error(e.getMessage());
            FinchModelMap.addError(e);
        }
    }

    private void initQuery(FinchModelMap modelMap, AccountPopUpQueryCommandForm commandForm)
    throws FinchException{
        commandForm.setScreenId("TRDEN");

        Identity principal = null;
        if (Operation.getInstance() != null && Operation.getInstance().getContext() != null) {
            principal = (Operation.getInstance().getContext()).getCallerIdentity();
        }

        commandForm.setCounterPartyDetailTypeMap(principal.getUserConstraint().get("COUNTER_PARTY_TYPE"));
        String locale =
        Operation.getInstance().getContext().getPreferenceBean().getPreference(PreferenceConstants.LOCALE);
```

```

String screenName =
Operation.getInstance().getContext().getReference(ScreenService.class).getScreenName("TRDEN", locale);
    commandForm.setScreenName(screenName);
    prepareContextsFromDependents(commandForm);
    prepareUIForContext(commandForm);

}

private void prepareContextsFromDependents(AccountPopUpQueryCommandForm form) {
    Map<String, String> localDependents = form.getDependents();
    for (String contextKey : localDependents.keySet()) {
        // AccountType Context
        if (StringUtils.contains(contextKey.toLowerCase(),
            PopupUtility.ACCOUNT_CONTEXT_ACCOUNT_TYPE.toLowerCase())))
            form.setCtxAccountType((String) localDependents.get(contextKey));

        // CounterPartyType Context
        if (StringUtils.contains(contextKey.toLowerCase(),
            PopupUtility.ACCOUNT_CONTEXT_COUNTER_PARTY_TYPE.toLowerCase())))
            form.setCtxCounterPartyDetailType((String) localDependents.get(contextKey));

        // Status Context
        if (StringUtils.contains(contextKey.toLowerCase(),
            PopupUtility.ACCOUNT_CONTEXT_STATUS.toLowerCase()))
            form.setCtxStatus((String) localDependents.get(contextKey));

        // Settlement Bank Context
        if (StringUtils.contains(contextKey.toLowerCase(),
            PopupUtility.ACCOUNT_CONTEXT_SETTLEMENT_BANK.toLowerCase())))
            String value = localDependents.get(contextKey);
            form.setCtxSettlementBank(value);
            form.setCounterPartyCode(value);
    }
}

/***
 * This prepare the UI accordingly if any context exists.
 *
 * @param form
 * @param request
 */
private void prepareUIForContext(AccountPopUpQueryCommandForm form) {
    if (!Strings.isNullOrEmpty(form.getCtxCounterPartyDetailType())) {
        List<String> tmpCPLList = PopupUtility.getParsedContextValues(form
            .getCtxCounterPartyDetailType());
        Map<String, String> counterPartyDetailTypeMap = form
            .getCounterPartyDetailTypeMap();
        Map<String, String> cp = new HashMap<>();

        for (String licx : tmpCPLList) {
            if (counterPartyDetailTypeMap.containsKey(licx)) {
                cp.put(licx, counterPartyDetailTypeMap.get(licx));
            }
        }
        form.setCounterPartyDetailTypeMap(cp);
    }
}

```

```
@Override
protected Class<?> getResultViewClass() {
    return AccountQuerySummaryView.class;
}

@Override
protected String getQueryFormViewName() {
    return "accountPopUpQueryCriteria";
}

@Override
protected String getQueryResultViewName() {
    return "accountPopUpQueryResult";
}

@Override
protected List<AccountQuerySummaryView> getQueryResult(AccountPopUpQueryCommandForm queryForm)
throws FinchException {
    AccountService service = Operation.getInstance().getReference(AccountService.class);
    List<AccountQuerySummaryView> list = new ArrayList<AccountQuerySummaryView>();
    try {
        list = service.getQueryResult(queryForm);
    } catch(Exception e) {
        log.error(e.getMessage());
    }
    return list;
}

@Override
protected String getComponent(){
    return "REF";
}
```

```
}
```

```
}
```

In controller we have to override:

- `getQueryFormViewName()` and return the name of query jsp page. For example here it is `accountPopQueryCriteria`.
- `getQueryResultViewName()` and return the name of query result jsp page. For example here it is `accountPopQueryResult`.
- `getQueryResult()`. Here we have to write what the query will do.

Popups

finch supports two types of popup.

- Simple popup – It appears when a button is clicked. It displays some records display.
- Query popup – In which you can give some criteria and submit. Then resulting records display.

Simple popup

Simple popup, extend abstract class `SimplePopupController.java` and implement following methods:

- `getComponent()` – returns module name string.
- `getQueryResultViewName()` - return the view name (by which `xxx.jspx` is resolved for displaying the popup).
- `getQueryResult()` - returns a `List` object(consisting of records). Here user can write their own method to get the list of displaying bean.

Example:

To get the instrument list:

```
@Controller
@RequestMapping(value = "/ref/instrument/query/ccy")
@SessionAttributes({"commandForm"})
public class RefInstrumentQueryController
    extends SimplePopupController<InstrumentQueryCommandForm> {

    private InstrumentService service = null;

    @Override
    protected List<?> getQueryResult() throws FinchException {
        List<Instrument> list = new ArrayList<Instrument>();
        try {
            list = getService().getInstrumentsByInstrumentType(RefConstants.CCY);
        } catch(Exception e) {
            Log.error(e.getMessage());
        }
        return list;
    }

    private InstrumentService getService(){
        if(service == null)
            service = Operation.getInstance().getReference(InstrumentService.class);
        return service;
    }
}
```

Figure: 1

```
public interface InstrumentService {  
  
    Instrument findInstrument(String securityId, String codeType) throws RefException;  
  
    /**  
     * Returns all the Instruments by InstrumentType  
     * */  
    List<Instrument> getInstrumentsByInstrumentType(String instrumentType);
```

```
@Service  
public class InstrumentServiceImpl implements InstrumentService {  
    private static final Log LOGGER = LogFactory.getLog(InstrumentServiceImpl.class);
```

```
    @Autowired  
    private InstrumentRepository repo;
```

```
    public List<Instrument> getInstrumentsByInstrumentType(String instrumentType){  
        return repo.getInstrumentsByInstrumentType(instrumentType);  
    }
```

```
@Repository  
public class InstrumentRepository extends RepositoryBase<Instrument, Long> {  
  
    public List<Instrument> getInstrumentsByInstrumentType(String instrumentType){  
        String finchSql = " select i " +  
            " from Instrument i " +  
            " where i.instrumentType = '" + instrumentType + "' " +  
            " and i.status = 'NORMAL' " +  
            " order by shortName ";  
        return getEntityManager().createQuery(finchSql, Instrument.class).getResultList();  
    }
```

Figure:

```

@Entity
@Table(name="REF_INSTRUMENT")
public class Instrument extends BaseRefEntity implements Serializable {
    private static final long serialVersionUID = 1L;
    private long instrumentPk;
    private String instrumentType;
    private Date issueDate;
    private BigDecimal minTradingUnit;
    private String officialName;
    private String securityId;
    private String shortName;
    private String status;
    private Instrument instrument;
    private Set<Instrument> instruments;
    private FinInstRole finInstRole;
    private Country country;
    private Set<InstrumentCrossRef> instrumentCrossRefs;

    public Instrument() {
    }

    @Id
    @SequenceGenerator(name="REF_INSTRUMENT_INSTRUMENTPK_GENERATOR", sequenceName="SEQ_INSTRUMENT_PK")
    @GeneratedValue(strategy=GenerationType.SEQUENCE, generator="REF_INSTRUMENT_INSTRUMENTPK_GENERATOR")
    @Column(name="INSTRUMENT_PK")
    public long getInstrumentPk() {
        return this.instrumentPk;
    }

    public void setInstrumentPk(long instrumentPk) {
        this.instrumentPk = instrumentPk;
    }

    @Column(name="INSTRUMENT_TYPE")
    public String getInstrumentType() {
        return this.instrumentType;
    }

    public void setInstrumentType(String instrumentType) {
        this.instrumentType = instrumentType;
    }
}

```

Figure: 3

value name will be used to identify the object in xxx.jspx page to render the search result.

```

<div xmlns:c="http://java.sun.com/jsp/jstl/core"
      xmlns:fn="http://java.sun.com/jsp/jstl/functions"
      xmlns:spring="http://www.springframework.org/tags"
      xmlns:util="urn:jsptagdir:/WEB-INF/tags/util"
      xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">

    <jsp:directive.page contentType="text/html;charset=UTF-8" />
    <jsp:output omit-xml-declaration="yes" />

    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:attribute name="src"><c:url value="/scripts/inf/finch-formatters.js"/></jsp:attribute>
        <jsp:body></jsp:body>
    </jsp:element> .

    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:body>
            var grid_result_data = [];
            var grid_result_columns = [
                {name:"",field:"header",id:"header",width:50,formatter:Slick.Formatters.SelectPopQueryFormater},
                {name:<spring:message code="ref.label.popup.securitytype" htmlEscape="false"/>,targetColumn:true, field:"value",width:150,formatter:Slick.Formatters.SelectPopQueryFormater},
                {name:<spring:message code="ref.label.popup.securityname" htmlEscape="false"/>,field:"label",id:"label", width:150,formatter:Slick.Formatters.SelectPopQueryFormater}
            ];
            var grid_result_settings = {
                isPopUpQuery:true,
                pagingInfo:{},
                url:{}}
            ;
            var row_id = 0;
            var rec = {};
            <c:forEach items="${value}" var="dl">
                rec = {};
                row_id+=1;
                rec.id = "finch_" + row_id;
                rec.value = "<c:out value='${dl.securityId}'/>";
                rec.label = "<c:out value='${dl.shortName}'/>";
                grid_result_data.push(rec);
            </c:forEach>
        </jsp:body>
    </jsp:element>
</div>

```

doInit() - this is a void method, and user can do some pre-setting in this method before actually execute the query.

Example:

```

@Override
public void doInit(FinchModelMap modelMap,
                    InstrumentQueryCommandForm form) throws FinchException {
    form.setScreenId("TRDEN");
}

```

Figure: 6

Request Mapping

request url from client side should end with init (e.g. - /...xxx/init). Then user have to map their implemented class for simple popup with annotation @RequestMapping(value = "/...xxx/")

Query Popup

For Query popup, extend abstract class AbstractQueryController.java and implement all methods.

Print Button in Details Popup Query Page

If the user want to add the print button in Details Popup Query Page then used need to add the following code in the corresponding jspx page.(e.g. in our application we can refer tradeDetailView.jspx file)

```
<jsp:element name="script">
<jsp:attribute name="type">text/javascript</jsp:attribute>
<jsp:body>
var FINCH$Dialog$Detail$Hook = function (container, btnContainer) {
    var details$View$print$Handler = function printHandler(e) {
        e.preventDefault();
        $('.ui-dialog.topMost .ui-dialog-content').printArea();
    };
    // show different buttons
    var $btn1 = jQuery('.btn1', btnContainer);
    $btn1.children().first().html("Print");
    $btn1.show();
    $btn1.click(details$View$print$Handler);
};
</jsp:body>
</jsp:element>
```

3.4.6.2.5 Message Notification

Introduction

Message Notification in finch has been implemented using Growl Notify mechanism. It is a JavaScript notification plugin designed to provide an unparalleled level of flexibility as well as very easy to implement and use.

How it Works?

The growl notification in finch is achieved by postNotice function described in finch.js. The function definition is given below:

```
var postNotice = function(type, message, persistent, timeout){
```

- type – refers to the notice type viz warning, error, info. FINCH uses warning as the default type.
- message – refers to the message or a list of messages that needs to be displayed.
- persistent – refers to the Boolean flag which indicates the growl type. If set to true, it will behave as persistent notification, else non-persistent.
- timeout – refers to the duration specified by the user to remove the growl box. If not specified, the default time will be considered.

For displaying an persistent error message :

```
FINCH.postNotice(FINCH.notice.type.error, validationMessages);
return false;
```

Note:

The persistent flag is not required here because finch default persist the error messages.

However, if client wants the non-persistent error message,

```
FINCH.postNotice(FINCH.notice.type.error, validationMessages,false);  
return false;
```

For displaying an info message :

```
FINCH.postNotice(FINCH.notice.type.info, validationMessages);
```

To persist the info message,

```
FINCH.postNotice(FINCH.notice.type.info, validationMessages,true);
```

For displaying an warning message :

```
FINCH.postNotice(FINCH.notice.type.warning, validationMessages);
```

To persist the warning message

```
FINCH.postNotice(FINCH.notice.type.warning, validationMessages,true);
```

For displaying an success message :

```
FINCH.postNotice(FINCH.notice.type.success, validationMessages);
```

To persist the success message

```
FINCH.postNotice(FINCH.notice.type.success, validationMessages,true);
```

The application can configure their own default timeout described under app-pref.xml. Below is the code snippet:

```
app-pref.xml  
<?xml version="1.0" encoding="UTF-8"?>  
<ROOT>  
<!-- other preference information -->  
<GROWL_DEFAULT_TIMEOUT>1500</GROWL_DEFAULT_TIMEOUT>  
</ROOT>
```

Customizing Display of Error Messages Encountered From Server end

finch now facilitates client to customize the display of error messages encountered from server end. By default, error messages encountered from server end are displayed with FINCH.notice.type.error and persistent as true. The client has to implement the FINCH\$Error\$Message\$Hook in proper jspx page to customize the message display. For example,to display server side error messages encountered in submitting trade query

```
tradeQueryCriteria.jspx  
  
var FINCH$Error$Message$Hook = function(msg) {  
    FINCH.postNotice(FINCH.notice.type.error,msg,true);  
}
```

The msg is the exception message thrown from server end.

In case the FINCH\$Error\$Message\$Hook is not defined in the jspx, the default behaviour should be displayed.

3.4.6.2.6 i18n support

Introduction

In the following scenarios there is provision for i18n support:

- The labels, tooltips of input elements in the UI
- The menu names and the screen names
- The client side validation error messages and information messages
- The server side validation messages
- The server side error messages
- The constraint values for the user

Initial Setup

The basic concept behind the i18n support provided is that there is a set of localized key/value pairs in some files. There exist files containing the same keys for each locale, but the values corresponding to those keys vary according to the user's current locale. This is made possible in the server side of the application by using the resource bundle feature of java, for which there exists *.properties file for each locale. In the client side special scripts file containing key/value pairs are loaded according to the user's locale at the very beginning.

Server Side

The properties file are files containing simple key/value pairs. The following is an entry in a properties file :

```
global_language=Language  
pref_welcome_user=Welcome
```

These files are used as the ResourceBundles. The very first step to using ResourceBundle is to register the path of the ResourceBundle files. The path to the ResourceBundle files which will contain the key/value pairs have to be registered in the basenames property of the ReloadableResourceBundleMessageSource bean in —webmvc-config.xml. Ideally each application will have their own set of ResourceBundles.

```

<bean class="org.springframework.context.support.ReloadableResourceBundleMessageSource"
id="messageSource"
    p:fallbackToSystemLocale="false"
    p:defaultEncoding="UTF-8" >
<property name="basenames">
<list>
<value>WEB-INF/i18n/app</value>
<value>WEB-INF/i18n/messages</value>
<value>WEB-INF/i18n/dbd-messages</value>
<value>WEB-INF/i18n/ref-messages</value>
<value>WEB-INF/i18n/trd-messages</value>
</list>
</property>
</bean>

```

The path to the resource bundle files are registered in the <value> tags.

Client Side

There is an utility javascript method FINCH.loadLocalizedScript() which is used to load the localized version of a certain script. At first user's locale is ascertained and then locale-specific script files are loaded in the following manner :

```
FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/inf/finch-i18n.js', async: false}]);
```

Here, the arguments to the method is the path to the js file, but it is not specified whether the English or the Japanese version is to be loaded. The filename previous to the locale specific section (i.e. _en or _ja) is given. The loadLocalizedScript() method appends the locale specific section to the file name depending on the user's locale and finally loads the file. Now wherever a locale-specific value is to be used, we write the corresponding variable name as stated in one of the localized scripts. These scripts contains the same list of variables in an organised manner. For each variable there exists a particular value which varies according to the locale-specific file it is in.

finch-i18n_en.js

```

FINCH.i18n = {
    title: {
        preferences: 'Preferences',
        formatters:{},
        tradeDetails: 'Trade Details',
        employeeDetails:'Employee Details'
    },
    message: {
        access_denied: 'Oops, you do not have permission.',
        your_session_expired: 'Oops, your session has expired. You will be taken to the login page.'
    }
};

```

finch-i18n_ja.js

```
FINCH.i18n = {
    title: {
        preferences: '¥u30D7',
        formatters: {
            tradeDetails: '¥u53D6¥u5F15¥u306E¥u8A73¥u7D30',
            employeeDetails: '¥u793E¥u54E1¥u8A73¥u7D30'
        }
    },
    message: {
        access_denied: '¥u304A¥u3063¥u3068¥u3001¥u3042¥u306A¥u305F¥u306F¥u6A29¥u9650¥u304C¥u3042¥u308A¥u307E¥u305B¥u3093¥u3002',
        your_session_expired: '¥u304A¥u3063¥u3068¥u3001¥u3042¥u306A¥u305F¥u306E¥u30BB¥u30C3¥u30B7¥u30E7¥u30F3¥u304C¥u671F¥u9650¥u5207¥u308C¥u306B¥u306A¥u3063¥u3066¥u3044¥u307E¥u3059¥u3002¥u3042¥u306A¥u305F¥u304C¥u30ED'
    }
};
```

As it can be seen both the files contain the same variables. The value that will be used depends on which of the above file is loaded. For example using 'FINCH.i18n.title.preferences' variable at any place replaces it with the locale-specific value for it.

How it Works?

This section describes how to use the i18n support in the set of scenarios stated at the top:

Labels, Tool-Tips of Input Elements in UI

The labels for the input elements, texts in the UI can be resolved using '<spring:message>' tag which ultimately replaces values against particular message keys as stored in the properties file. At the very top of the file, the URI for the spring tag library descriptor file is to be declared:

```
xmlns:spring=http://www.springframework.org/tags
```

And the tag is used in the following manner:

```
<spring:message code="dbd.label.tsm.dashboard" htmlEscape="false"/>
```

Here the value against the key dbd.label.tsm.dashboard is replaced in the file according to the user's locale.

The tooltips that are shown when mouse is hovered over certain html elements are nothing but the "title" attribute of that element. So showing locale-specific tooltips depends on having locale-specific strings as 'title'. If the title is set directly in the jspx file, then it can be set in the following way

```
<spring:message code="dbd.label.tsm.test" var="title" htmlEscape="false"/>
<div class="toggleMenuArea img" title="$${title}"/>
```

In the above example, the value cannot be directly replaced, thus at first the value against a particular key is stored in a variable and is used at the required place using Expression Language. The way to set a locale-specific value from a script file are shown in the following examples.

Menu and Screen Names

The menu names and the screen names are shown according to the user's current locale and are independent of the values stored as their names in the database. To assign names to menus for a particular locale, separate properties file are maintained for assigning names to menus. The key for the menu is formed by appending .name to the menu_id of a menu and is paired with the particular value for that menu in that particular locale which that properties file is representing. For example :

For English locale the following is an entry to —menu.properties:

```
TRD.name=TRADE
```

For Japanese locale the following is an entry to —menu_ja.properties:

```
TRD.name=¥u53D6¥u5F15 ( The Japanese value is here represented in UTF-8).
```

Here, TRD is the menu_id, '.name' is appended to it to form locale-specific names of it. In a similar manner, the screen names are also assigned through its corresponding properties file. For example:

For English locale the following is an entry to —screen.properties:

```
TRDEN.name=TRADE ENTRY
```

For Japanese locale the following is an entry to —screen_ja—.properties:

```
TRD.name=¥u8CBF¥u6613¥u30A8¥u30F3¥u30C8¥u30EA¥u30FC ( The Japanese value is here represented in UTF-8).
```

Here, *TRDEN *is the screen_id, '.name' is appended to it to form locale-specific names of it.

Client Side Validation Error Messages and Information Messages

In the following example, a client side validation is taking place to see if the value in a certain textfield is empty, and if it is accordingly a validation error message is shown in the growl using FINCH.postNotice :

```
if ($dataStart.val() === ""){
    FINCH.postNotice(FINCH.notice.type.error, FINCH.i18n.upload.data_row_no_absent, true);
    return false;
}
```

Here, the localized value against the variable FINCH.i18n.upload.data_row_no_absent is shown if the above validation fails. Information messages for certain user activities such as on successful completion of an activity, etc. are also resolved through the variables of the localized js files loaded. In the following snippet, the notification for removing a menu shortcut is shown with a locale-specific string

```
FINCH.postNotice('info', FINCH$DBD$i18n.menu_shortcut.shortcut_removed);
```

Here FINCH\$DBD\$i18n.menu_shortcut_shortcut_removed is a variable in one of the files loaded locale-specifically using the utility method FINCH.loadLocalizedScript. The alert, confirm and prompt boxes are also fed locale-specific data that are to be shown in their dialog boxes. In the following example the alert box to be shown when the user has timed out is fed locale-specific string FINCH.i18n.message.your_session_expired.

```
jAlert(FINCH.i18n.message.your_session_expired, null, function() {
    window.location.assign(FINCH.context.path + '/resources/j_spring_security_logout', '_self');
});
```

Server Side Validation Messages

To use the messages from ResourceBundles that are specified in the —webmvc-config.xml, the java class must implement MessageSource Aware. This gives the java class access to the keys specified in the properties file. One has to override the setMessageSource() method. The Resource files are automatically injected by spring during its initialization by calling this setMessageSource() method.

For example :

Here TradeQueryController is implementing MessageSourceAware:

TrdQueryController.java

```
@Override  
public void setMessageSource(MessageSource messageSource){  
    this.messageSource=messageSource;  
}
```

To use a particular value against a particular locale from the ResourceBundle files specified, we extract the value using getMessage() method of the MessageSource object. For example in the following snippet a localised string is added to the list "invalidMessages" which are to be shown to the end-user :

```
invalidMessages.add(messageSource.getMessage("upload.error.cannotRead", null, locale));
```

Server Side Error Messages

The error messages which are generated as a result of certain exceptions while processing in the server side are also resolved through error-specific properties file, the key-value pairs corresponding to error messages are stored in the file error-resources.properties and its locale-specific copies. For example :

```
throw new FinchException("upload.action.error.populate", e);
```

Here upload.action.error.populate is the error key, and 'e' is the exception object. The value against the key for the current locale is resolved.

If the user wants to add any value passed from server side with the error message user can declare the parameter as [%s] or {0},{1} with the message string.

Below is the example:

```
throw new InfException("error.screen.notfound", screenId);
```

and in .properties file user have to declare as below

```
error.screen.notfound=Screen entry not found for screen id = {0}
```

Internationalization support is provided through the ResourceBundle feature of java. Instead of hard-coding text in the view, the text to be shown to the end-user is resolved at runtime using user's locale and according to his preferences using .properties file of ResourceBundle containing key/value pairs.

Constraint values for the user :

finch stores constraint or possible values for field in following two database tables.

- INF_CONSTRAINT_LIST
- INF_CONSTRAINT_VALUES

The INF_CONSTRAINT_VALUES table contains the constraint values is to be populated with different values for different language for the same constraint name.

Steps for Providing Internationalization Support

XML Configuration

At first the path to the ResourceBundle files which will contain the key/value pairs have to be registered in the basenames property of the ReloadableResourceBundleMessageSource bean in —webmvc-config.xml. Ideally each application will have their own set of ResourceBundles.

```
<bean class="org.springframework.context.support.ReloadableResourceBundleMessageSource"  
id="messageSource"  
p:fallbackToSystemLocale="false"  
p:defaultEncoding="UTF-8" >  
<property name="basenames">  
<list>  
<value>WEB-INF/i18n/app</value>  
<value>WEB-INF/i18n/messages</value>  
<value>WEB-INF/i18n/dbd-messages</value>  
<value>WEB-INF/i18n/ref-messages</value>  
<value>WEB-INF/i18n/trd-messages</value>  
</list>  
</property>  
</bean>
```

JAVA Configuration

The ResourceBundle files stated in the above manner will be used in the java files and jspx files for replacing locale-specific values. In case of java files, the ResourceBundle is to be explicitly stated from where the values are to be extracted against particular keys. For example, to assign names to menus for a particular locale, separate properties file are maintained for assigning names to menus. The key for the menu is formed by appending .name to the menu id of a menu and is paired with the particular value for that menu in that particular locale which that properties file is representing.

For example:

For English locale the following is an entry to —menu.properties: TRD.name=TRADE and for Japanese locale the entry to —menu_ja.properties:TRD.name=¥u53D6¥u5F15 (Japanese value is here represented in UTF-8)

In a similar manner, the screen names are also assigned through its corresponding properties file. Another way to use values from ResourceBundles that are specified in the —webmvc-config.xml is for the java class to implement MessageSourceAware. This gives the java class access to the keys specified in the properties file. One has to override the setMessageSource() method. The Resource files are automatically injected by spring during its initialization by calling this setMessageSource() method.

For example:

Here TradeQueryController is implementing MessageSourceAware:

TrdQueryController.java

```
@Override  
public void setMessageSource(MessageSource messageSource){  
this.messageSource=messageSource;  
}
```

To use a particular value against a particular locale from the ResourceBundle files specified, we extract the value using getMessage() method of the MessageSource object.

For example:

```
(messageSource.getMessage("trd.sort.displayvalue.tradedate", null, new Locale(strLocale)))
```

The above method retrieves value against the key trd.sort.displayvalue.tradedate against the locale which is represented in String through the strLocale variable and passes no parameters to the key hence the middle argument is null .

Error messages which are to be shown on the occurrence of an exception also have i18n support, the key-value pairs corresponding to error messages are stored in the file error-resources.properties and its locale-specific copies.

For example:

```
throw new FinchException("upload.action.error.populate", e);
```

Here "upload.action.error.populate" is the error key, and 'e' is the exception object. The value against the key for the current locale is resolved.

JSPX Configuration

As for the jspx files it uses values from the ResourceBundle files using <spring:message> tag.

At the very top of the file, the URI for the spring tag library descriptor file is to be declared:

```
xmlns:spring=http://www.springframework.org/tags
```

The tag is used in the following manner:

```
<spring:message code="dbd.label.tsm.dashboard" htmlEscape="false"/>
```

Here the value against the key dbd.label.tsm.dashboard is replaced in the file according to the user's locale. At places the value cannot be directly replaced, in those cases at first the value against a particular key is stored in a variable and is used at the required place using EL.

```
<spring:message code="dbd.label.tsm.test" var="title" htmlEscape="false"/>
<div class="toggleMenuArea img" title="${title}">
```

i18n Support in the Client Side

All the texts, labels, messages that are generated in the client side through javascript has i18n support. A separate method is in place for internationalization of javascript files. At first user's locale is ascertained and then locale-specific script files are loaded in a special manner. These scripts contains the same list of variables in an organised manner. For each variable there exists a particular value which varies according to the locale-specific file it is in.

finch has two scripts —finch-i18n_en.js and —finch-i18n_ja.js. Both the scripts contain same list of variables in the same manner, but the values against the variable in the first script are in English whereas that in case of the second script are in Japanese. An illustration is shown below,

finch-i18n_en.js

```
FINCH.i18n = {
  title: {
    preferences: 'Preferences',
  formatters:{ 
    tradeDetails: 'Trade Details',
    employeeDetails:'Employee Details'
  }
  },
  message: {
    access_denied: 'Oops, you do not have permission.',
    your_session_expired: 'Oops, your session has expired. You will be taken to the login page.'
  }
};
```

finch-i18n_ja.js

```
FINCH.i18n = {
  title: {
    preferences: '\u30D7',
  formatters:{ 
    tradeDetails: '\u53D6\u5F15\u306E\u8A73\u7D30',
    employeeDetails:'\u793E\u54E1\u8A73\u7D30'
  }
  },
  message: {
    access_denied: '\u304A\u3063\u3068\u3001\u3042\u306A\u305F\u306F\u6A29\u9650\u304C\u3042\u308A\u307E\u305B\u3093\u3002',
    your_session_expired: '\u304A\u3063\u3068\u3001\u3042\u306A\u305F\u306E\u306F\u30BB\u30C3\u30B7\u30E7\u30F3\u304C\u671F\u9650\u5207\u308C\u306B\u306A\u3063\u3066\u3044\u307E\u3059\u3002'
  }
};
```

As it can be seen both the files contain the same variables. The value that will be used depends on which of the above file is loaded. There is an utility javascript method FINCH.loadLocalizedScript() which is used to load the localized version of a certain script.

For example, the localized script for the above example is loaded in the following way:

```
FINCH.loadLocalizedScript([${path: FINCH.context.path + '/scripts/inf/finch-i18n.js', async: false}]);
```

Here, the arguments to the method is the path to the js file, but it is not specified whether the English or the Japanese version is to be loaded. The filename previous to the locale specific section (i.e. _en or _ja) is given. The loadLocalizedScript() method appends the locale specific section to the filename depending on the user's locale and finally loads the file. Now wherever a locale-specific value is to be used, we write the corresponding variable name as stated in one of the localized scripts. For example using 'FINCH.i18n.title.preferences' variable at any place replaces it with the locale-specific value for it.

i18n Support in the Database

i18n support is also provided through the values retrieved from the database. The user constraint is populated with values specific to the user's preference and language. The table containing the constraint values is to be populated with values having different ui_display_value and charset_code for the same constraint value. For example, For the same constraint 'NORMAL', two entries are made having different ui_display_value and charset_code. First entry had as its ui_display_value and charset_code, 'NORMAL' and ENGLISH respectively, whereas the second entry had as its ui_display_value and charset_code, '???' and JAPANESE respectively.

For retrieving non-static values (e.g : saved criteria name) from the database and displaying the same value in the UI a different approach is considered. There is an utility javascript method toUnicode(theString) in the —finch-saved-query-handler.js which is used to convert the localized string to its UTF-8 equivalent. The converted UTF-8 equivalent string is passed as a parameter to the controller where the UTF-8 string is converted back to the original localized format using StringEscapeUtils.unescapeJava(param), where param is the parameter containing UTF-8 equivalent string.

An illustration is given below:

Let us suppose, In Trade Query, a criteria was saved with the name:???

```
finch-saved-query-handler.js
var headerInfo=toUnicode(??? );
```

The above line converts the string ??? to '¥u30A4¥u30F3¥u30C9' and is passed to the controller.

```
TrdQueryController.java
String criteriaName=StringEscapeUtils.unescapeJava(headerInfo);
```

The above line converts the '¥u30A4¥u30F3¥u30C9' back to ??? and is displayed in UI.

Change application locale from UI

Application client can change the application locale through UI by clicking preference icon. Please follow the link [5.2.2.8 User Preference](#) [3.4.6.2.7 Implement Excel/CSV Upload Screen](#)

Introduction

finch provides the generic infrastructure needed for file uploads. This feature is used for uploading .xls,.xlsx,.csv,.tsv,.tab files for any given module as per the requirement. Please refer to [Upload UI Manual](#) for how to use upload UI provided by finch .

Upload Template

For each upload UI, first a template for the upload UI needs to be defined. The template describes the fields those are required for loading data. The uploaded document can be of any layout. There is a mapping UI where user can define mapping between data file layout and template. The finch upload framework converts data from file to a list of bean objects. The bean objects have the properties as per the template definition. The template in the upload module represents a JSON document which needs to be saved in the database table (INF_UPLOAD_TEMPLATE) as a CLOB for each template entry. Here is a sample template:

A sample template JSON document

```
[  
  {  
    "name": "Trade Date",  
    "type": "java.util.Date",  
    "mandatory": true  
  },  
  {  
    "name": "Execution Market",  
    "type": "java.lang.String",  
    "mandatory": true  
  },  
  {  
    "name": "Trade Type",  
    "type": "java.lang.String",  
    "mandatory": true  
  },  
  {  
    "name": "Value Date",  
    "type": "java.util.Date",  
    "mandatory": true  
  },  
  {  
    "name": "Trade Currency",  
    "type": "java.lang.String",  
    "mandatory": true  
  },  
  {  
    "name": "Account No",  
    "type": "java.lang.String",  
    "mandatory": false  
  },  
  {  
    "name": "Price",  
    "type": "java.math.BigDecimal",  
    "mandatory": true  
  },  
  {  
    "name": "Quantity",  
    "type": "java.math.BigDecimal",  
    "mandatory": true  
  }  
]
```

Here each object in the JSON array represents a single tag or field against which the values in the uploaded file is mapped and processed. The following table describes the properties of each objects in the template :

Name	It represents a tag against which a column in the uploaded file will be mapped.
Type	It represents the expected java datatype of the field.
Mandatory	It represents the mandatory nature of the field, whether it is a must for a valid upload or not.

How to Implement

For implementing the file upload feature do the followings:

- make a binding commandForm object and a controller. For example, if the module is Trade, make a TradeUploadCommandForm which will extend the UploadCommandForm object of finch
- make a controller TradeUploadController, which extends finch's UploadController.

Code snippet is as follows:

TradeUploadCommandForm

```
package com.nrft.sample.trd.web.form;
import com.nrft.finч.inf.web.upload.UploadCommandForm;
public class TradeUploadCommandForm extends UploadCommandForm{
    public TradeUploadCommandForm(){
    }

    @Override
    public String getModule() {
        return "TRD";
    }

    @Override
    public String setWizardBaseUri(){
        return "trd/upload";
    }
}
```

In the commandForm object, override the getModule method and return the module name. Set the base URI of the [finch-wizard](#) that is being used in this case. Both the methods must be implemented. Make sure it is the same as the root URI of your controller. For example, the root URI of the controller for this command Form is /trd/upload/**.

In the case of controller, six methods can be overridden to implement the business logic at certain points in the execution. The upload module implicitly uses [finch-wizard](#), so it follows the same pattern of methods that the [finch-wizard](#) requires. The upload module provides six methods for implementing the logic in the associated wizard methods, as listed in the following table:

dolnItAction	The hook in the init action of the wizard	Here the action against which the templates are to be populated in the drop-down is to be set. The status of whether to show the action field or not, is to be set. This is an action hook while transiting to the first screen from the menu.
doSubmitAction	The hook in the submit action of the wizard	Here the main business logic for checking the values are to be implemented. In case of invalid values, they are to be registered with the parent controller. This is an action hook while transiting to the third screen user confirmation from the second screen.
doValidateAction	The hook in the validate action of the wizard	Application specific business logic is implemented here. This action hook is passed an integer value representing the transition state. If it is 0, it signifies the transition is from first to second screen and occurs before the doNavigate event. If it is 1, it signifies the transition is from second to third screen and occurs before the doSubmit event. It must be noted here that the list of beans are not available in the command form at this instance and are only available after the doSubmit action.

doNavigateAction	The hook in the navigate action of the wizard	Application specific business logic while transiting from the first upload screen to the second mapping screen is implemented here.
doConfirmAction	The hook in the confirm action of the wizard	Here the main logic of saving the document is to be implemented using the service method.
doBackAction	The hook in the back action of the wizard	Here the custom logic while implementing the back action from the second screen to the first screen (if any), is implemented.

doInitAction

```
@Override
protected void doInitAction(TradeUploadCommandForm form) {
    form.setAction("Trade"); // This is the action against which the templates are saved in the database.
    form.setShowAction("no");
    form.setSuccessMessage("Trade Uploaded Successfully");
}
```

Setting the **commandForm** to the string **yes** or not setting it at all, results in a drop-down appearing in the first screen. From the drop-down end-user can choose a particular action against which the templates are populated in the second drop-down. Else only a single drop-down is shown in the first screen where the list of templates are already populated. Success message can be set here and can be shown when the file is uploaded successfully.

doSubmitAction

```
@Override
protected void doSubmitAction(TradeUploadCommandForm form) {
    List<DynaBean> beanList = form.getBeanList();
    TradeDTO trdDTO = new TradeDTO();
    AccountService accountService =
        Operation.getInstance().getContext().getReference(AccountService.class);
    FinInstRoleService finInstService =
        Operation.getInstance().getContext().getReference(FinInstRoleService.class);
    InstrumentService instrumentService =
        Operation.getInstance().getContext().getReference(InstrumentService.class);
    int startRow = form.getDataStartRowNo();
    int row;
    String tagName;
    String cellName;
    TemplateMappingDTO tempMapDTO = form.getTemplateMappingDTO();

    int j = 0;
```

```

for (DynaBean dynaBean : beanList) {
    row = startRow+j;
    if(dynaBean.get("Account No")!=null){
        try {
            accountService.resolveAccountNoWithType((String)dynaBean.get("Account No"));
        } catch (RefException e) {
            tagName = "Account No";
            int col = tempMapDTO.getColumnMap().get(tagName).index;
            cellName = XLSUploadReaderUtil.getCellName(row, col);
            UploadParserHelper.addToInvalidMessages(form,
messageSource.getMessage("trd.upload.error.accountno", new Object[]{{(String)dynaBean.get("Account
No")},cellName}, new Locale(getLocaleString()),cellName));
            log.error("Invalid Account No", e);
        }
    }
    if(dynaBean.get("Inventory Account")!=null){
        try {
            accountService.resolveInvAccountNoWithType((String)dynaBean.get("Inventory Account"));
        } catch (RefException e) {
            tagName = "Inventory Account";
            int col = tempMapDTO.getColumnMap().get(tagName).index;
            cellName = XLSUploadReaderUtil.getCellName(row, col);
            UploadParserHelper.addToInvalidMessages(form,
messageSource.getMessage("trd.upload.error.inventoryaccountno", new
Object[]{{(String)dynaBean.get("Inventory Account")},cellName}, new Locale(getLocaleString()),cellName));
            log.error("Invalid Inventory Account", e);
        }
    }
    if(dynaBean.get("Trade Currency")!=null){
        try {
            instrumentService.resolveSecurityIdWithType((String)dynaBean.get("Trade Currency"));
        } catch (RefException e) {
            tagName = "Trade Currency";
            int col = tempMapDTO.getColumnMap().get(tagName).index;
            cellName = XLSUploadReaderUtil.getCellName(row, col);
            UploadParserHelper.addToInvalidMessages(form,
messageSource.getMessage("trd.upload.error.tradecurrency", new Object[]{{(String)dynaBean.get("Trade
Currency")},cellName}, new Locale(getLocaleString()),cellName));
            log.error("Invalid Trade Currency", e);
        }
    }
    if(dynaBean.get("Trade Type")!=null){
        try {
            if(!trdDTO.getTradeTypeValues().containsValue((String)dynaBean.get("Trade Type"))){
                tagName = "Trade Type";
                int col = tempMapDTO.getColumnMap().get(tagName).index;
                cellName = XLSUploadReaderUtil.getCellName(row, col);
                UploadParserHelper.addToInvalidMessages(form,
messageSource.getMessage("trd.upload.error.tradetype", new Object[]{{(String)dynaBean.get("Trade
Type")},cellName}, new Locale(getLocaleString()),cellName));
            }
        } catch (TrdException | NoSuchMessageException e) {
            log.error("Invalid Trade Type", e);
        }
    }
    if(dynaBean.get("Status")!=null){
        if(!dynaBean.get("Status").toString().equals(Globals.NORMAL_STATUS) &&

```

```
!dynaBean.get("Status").toString().equals(Globals.CANCEL_STATUS)){
    tagName = "Status";
    int col = tempMapDTO.getColumnMap().get(tagName).index;
    cellName = XLSUploadReaderUtil.getCellName(row, col);
    UploadParserHelper.addToInvalidMessages(form,
messageSource.getMessage("trd.upload.error.status", new Object[]{(String)dynaBean.get("Status"),cellName},
new Locale(getLocaleString())),cellName);
}
j++;
```

```
    }  
  
}
```

As per the above code snippet, custom business logic is implemented here and in case of errors or exceptions, one needs to register the error message with the help of addTolnvalidMessages method of the UploadParserHelper class. The method takes as its parameter the form object and the error message String. The addTolnvalidMessages is an overloaded method which can either take a cell name (in case of excel file uploads) as one of its arguments or no argument.

doConfirmAction

```
@Override  
protected void doConfirmAction(TradeUploadCommandForm form) throws DataInputException {  
    TradeService tradeService = Operation.getInstance().getContext().getReference(TradeService.class);  
    try {  
        tradeService.uploadTrade(form);  
    } catch (DataInputException e) {  
        throw e;  
    }  
}
```

The main method for finally uploading the trade is implemented in the doConfirmAction method.

Maximum Error limit implementation

There is a concept of maximum error limit in file upload. The application should set in the app-pref.xml file the maximum permissible error limit before the file stops being further processed in the following manner.

app-pref.xml

```
<MAX_UPLOAD_ERRORS>20</MAX_UPLOAD_ERRORS>
```

The above xml snippet shows the error limit being set in the app-pref.xml file. In this case, while processing the uploaded file if at anytime the no of errors found till that moment reaches 20, the file processing will stop at that moment and the end-user should first resolve the errors first before uploading again.

3.4.6.2.8 Customize User Preference

Introduction

Preference in finch takes care of the user's settings with respect to the application. It is a control panel like feature which enables the user to define various settings and store them against that user. The user's preference is loaded every time the user comes back to the application. The preference feature is provided by finch and the application just need to override the implementation, as required.

How to use

Please refer to [User Preference](#) manual to know how to use User Preference.

Configuration

finch provides basic infrastructure for plugging application preference; it can be plugged into finch module-wise. finch expects implementation of certain preference Java API only, client side JSON accumulation and JavaScript rendering all are done by finch as it's offerings.

API

finch discovers the plug-able preferences through `PreferenceService.java`. Specific application need to implement the following API:

finch preference API

```
ComponentPrefs getPreferences() throws FinchException;  
}
```

Specific module of an application need to expose the implementation `PreferenceService` through it's own `ComponentScope`, because finch will get the bean from individual component scope.

How-To

As a reference, let us assume that we need a preference for "sample-trd" module for "sample-app" application. Currently there are two types of preferences supported by finch out of the box: dropdown and zebracolor. Let us consider a drop-down preference for "sample-trd".

Create a component context XML Spring configuration context for this module (`trd-context.xml`):

sample-trd component XML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:aop="http://www.springframework.org/schema/aop"  
       xmlns:context="http://www.springframework.org/schema/context"  
       xmlns:jee="http://www.springframework.org/schema/jee"  
       xmlns:tx="http://www.springframework.org/schema/tx"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xsi:schemaLocation="  
           http://www.springframework.org/schema/aop  
           http://www.springframework.org/schema/aop/spring-aop-3.1.xsd  
           http://www.springframework.org/schema/beans  
           http://www.springframework.org/schema/beans/spring-beans-3.1.xsd  
           http://www.springframework.org/schema/context  
           http://www.springframework.org/schema/context/spring-context-3.1.xsd  
           http://www.springframework.org/schema/jee  
           http://www.springframework.org/schema/jee/spring-jee-3.1.xsd  
           http://www.springframework.org/schema/tx  
           http://www.springframework.org/schema/tx/spring-tx-3.1.xsd">  
  
    <bean id="trd-scope" name="trd-scope" class="com.nrft.sample.trd.TrdScope" init-method="init"  
          lazy-init="true"/>  
    <bean id="trd-prefs" name="trd-prefs" class="com.nrft.sample.trd.pref.TrdPreferenceService" />  
</beans>
```

application Spring configuration file (application-context.xml)

```
...
<import resource="classpath:META-INF/spring/component-ctx/inf-context.xml"/>
<import resource="classpath:META-INF/spring/component-ctx/dbd-context.xml"/>
<import resource="classpath:META-INF/spring/component-ctx/trd-context.xml"/>
...
```

TrdScope.java

```
package com.nrft.sample.trd;

import com.nrft.finch.inf.exception.FinchException;
import com.nrft.finch.inf.startup.ComponentScope;

/**
 * TrdComponentScope
 *
 * @version $Revision$
 * @author gautamr
 */
public class TrdScope extends ComponentScope {

    public TrdScope() {
        super(Component.getComponent("TRD").get());
    }

    @Override
    public void init() throws FinchException {
        loadServiceContext("trd-context.xml");
    }
}
```

TrdPreferenceService.java

```
package com.nrft.sample.trd.pref;

import java.util.ArrayList;
import java.util.List;

import com.google.common.collect.ImmutableMap;
import com.nrft.finch.inf.exception.FinchException;
import com.nrft.finch.inf.pref.ComponentPrefs;
import com.nrft.finch.inf.pref.Pref;
import com.nrft.finch.inf.pref.PreferenceService;

/**
 * TrdPreferenceService
 *
 * @version $Revision$
 * @author gautamr
 */
public class TrdPreferenceService implements PreferenceService {

    @Override
    public ComponentPrefs getPreferences() throws FinchException {
        List<Pref> prefList = new ArrayList<>(2);
        Pref p = new Pref();
        p.setComponent("trd");
        p.setType("dropdown");
        p.setKey("SAMPLE_TRD_PREF0");
        p.setValue("val1");
        p.setLabel("Sample TRD Preference");
        p.setMapItems(ImmutableMap.of("test1", "val1", "test2", "val2"));
        prefList.add(p);

        ComponentPrefs cp = new ComponentPrefs();
        cp.setComponentLabel("Dashboard");
        cp.setPrefs(prefList);

        return cp;
    }
}
```

3.4.6.2.9 UI Online Report

Introduction

Online reports help user to export the query result in different report formats like PDF,XLS,XLSX,CSV etc. This page explains the steps required to configure online reports.

Steps to create Online Report using finch

finch provides default templates for Report (xls and pdf). These default templates include title, page-header and page-footer templates for report. User can override these finch provided default templates and configure them specific to the application.

Followings are path default template files:

- report¥templates¥common¥title-pdf.jrxml
- report¥templates¥common¥header-pdf.jrxml
- report¥templates¥common¥footer-pdf.jrxml

Similar files for excel like report¥templates¥common¥title-pdf.jrxml also provides some styles files in report¥templates¥styles¥header-style.jrtx etc.

User can use finch provided template for there report or user also can create there own report template for there report.

Override finch Common Template

The following steps need to follow to override finch provided report template:

1. create title,header and footer jrxml any where and with any name in your project directory and include these report as a sub-report into your main report in appropriate position
2. compile and create corresponding.jasper file
3. Provide the following parameters
 - a. TITLE
 - b. FOOTER
 - c. COMMON_SUBREPORT_TITLE_PATH
 - d. COMMON_SUBREPORT_HEADER_PATH
 - e. COMMON_SUBREPORT_FOOTER_PATH
 - f. QUERY_SUBREPORT_PATH
4. There is default value provided for the above parameters from finch user can use those values.There is separate methods in finch for pdf and xls. The example code for pdf has been displayed in the following code snippet.

```
protected String getCommonTitleSubreportTemplatePathForPdf() {  
    return "report/templates/common/title-pdf";  
}
```

5. If user wants to create there own file (other than finch provided file), user need to override the above method of AbstractQueryController.java from finch.

```
e.g.  
@Override  
protected String getCommonTitleSubreportTemplatePathForPdf() {  
    return "report/templates/common/title";  
}
```

6. The last part of the relative path is the file name(in the above example 'report/templates/common/title' "title" is the file name used by the application).
7. If the user wants to use the same file name as provided by finch but there own design, user first need to exclude the finch provided file.

```
<!-- Exclude Finch Report Template File if application using the same file name as finch -->  
<exclude>**/WEB-INF/classes/report/templates/common/*.jrxml</exclude>  
<exclude>**/WEB-INF/classes/report/templates/common/*.jasper</exclude>
```

If user create different file name then no need to exclude finch provided jrxml files

8. User can also use the style files provided by finch to design there template.

```

<jasperTemplate>
    <style name="headerStyle" mode="Opaque" forecolor="#000000" backcolor="#CCCCCC" fill="Solid"
    fontName="ARIALUNI" fontSize="14">
        <box topPadding="1" leftPadding="1" bottomPadding="1" rightPadding="1"/>
        <paragraph firstLineIndent="1" leftIndent="1" rightIndent="1" spacingBefore="1" spacingAfter="1"/>
    </style>
</jasperTemplate>

```

9. User also can override finch provided style files (below is the example how to override style files) and can use there own style files into there own template.

```

<jasperTemplate>
    <style name="headerStyle" mode="Opaque" forecolor="#000000" backcolor="#CCCCCC" fill="Solid"
    fontName="ARIALUNI" fontSize="14">
        <box topPadding="1" leftPadding="1" bottomPadding="1" rightPadding="1"/>
        <paragraph firstLineIndent="1" leftIndent="1" rightIndent="1" spacingBefore="1" spacingAfter="1"/>
    </style>
</jasperTemplate>

```

10. finch provides the default TITLE name of the report user always override the name and can display there own name in there report.similar for FOOTER.

```

@Override
protected String getReportTitle() {
    return messageSource.getMessage("trade.label.trade.reporttitle", null, I18NUtils.getLocale());
}

```

After creating report template, user need to call the report from jspx and include or override some methods into there query controller using following steps.

- Create the jspx file which contains the two buttons for pdf and xlsx. For example, refer to the following code snippet used in jspx file:

```
var grid_result_settings = {
    enableToolbar:true,
    consolidateActionFlag:true,
    buttons:{
        print:false,
        xls:true,
        pdf:true,
        columnPicker:true,
        save:true
    },
    pagingInfo:{
        isNext : isNext,
        url: 'trd/query/count.json'
    },
    urls:{
        nextPage : '/trd/query/result.json?fetch=next',
        prevPage : '/trd/query/result.json?fetch=previous',
        pdfReport: '/trd/query/report/trade.pdf?filetype=pdf',
        xlsReport: '/trd/query/report/trade.xlsx?filetype=xlsx'
    }
};
```

- Configure webmvc-config.xml, as shown below:

```
<bean class="org.springframework.web.servlet.view.XmlViewResolver">
    <property name="order" value="2" />
    <property name="location" value="/WEB-INF/report/jasper-views.xml"/>
</bean>
```

- Create corresponding jasper-views.xml and write the code as shown below.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">

    <!-- <bean id="trdquery" class="com.nrft.sample.trd.report.MultiFormatReportView"> -->
    <bean id="trdquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="trdqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="trdquerycsv" class="com.nrft.fin.ch.inf.web.report.JasperReportsCsvView">
        <property name="url" value="/WEB-INF/classes/report/templates/trd/TrdSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="accesslogquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/inf/AccessLogSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="accesslogqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/AccessLogSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="empquerypdf"
          class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
        <property name="url" value="/WEB-INF/classes/report/templates/inf/UserSummary.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="empqueryxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/UserSummaryInExcel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>

    <bean id="holidayreportxlsx" class="com.nrft.fin.ch.inf.web.report.JasperReportsXlsxView">
        <property name="url"
                  value="/WEB-INF/classes/report/templates/inf/Holiday-Calendar-Report-Excel.jrxml"/>
        <property name="reportDataKey" value="data"/>
    </bean>
</beans>
```

- Create controller which extends finch AbstractQueryController now override getReportParameters() method of AbstractQueryController class, as shown below.

```

@Override
protected Map<String, Object> getReportParameters(
    @ModelAttribute("commandForm") TrdQueryCommandForm cmdForm)
throws FinchException {
Map<String, Object> params = Maps.newHashMap();
Map<String, String> criteriaMap = Maps.newLinkedHashMap();

addCriteria(criteriaMap, "Trade Type", cmdForm.getTradeType());
addCriteria(criteriaMap, "Trade Reference No", cmdForm.getReferenceNo());
addCriteria(criteriaMap, "Account No", cmdForm.getAccountNo());
addCriteria(criteriaMap, "Inventory Account No", cmdForm.getInventoryAccountNo());
addCriteria(criteriaMap, "Security", cmdForm.getSecurityCode());
addCriteria(criteriaMap, "Trade Currency", cmdForm.getTradeCcy());
addCriteria(criteriaMap, "Settlement Currency", cmdForm.getSettlementCcy());
addCriteria(criteriaMap, "Trade Date From", cmdForm.getTradeDateFrom());
addCriteria(criteriaMap, "Trade Date To", cmdForm.getTradeDateTo());
addCriteria(criteriaMap, "Value Date From", cmdForm.getValueDateFrom());
addCriteria(criteriaMap, "Value Date To", cmdForm.getValueDateTo());
addCriteria(criteriaMap, "Instrument Type", cmdForm.getInstrumentType());
addCriteria(criteriaMap, "Execution Market", cmdForm.getExecutionMarket());

addCriteria(criteriaMap, "Buy/Sell Flag", cmdForm.getBuySellFlag());
addCriteria(criteriaMap, "Principal/Agent Flag", cmdForm.getPrincipalAgentFlag());
addCriteria(criteriaMap, "Entry Date Form", cmdForm.getCreationDateFrom());
addCriteria(criteriaMap, "Entry Date To", cmdForm.getCreationDateTo());
addCriteria(criteriaMap, "Last Entry Date From", cmdForm.getUpdateDateFrom());
addCriteria(criteriaMap, "Last Entry Date To", cmdForm.getUpdateDateTo());
addCriteria(criteriaMap, "External Reference No", cmdForm.getExternalReferenceNo());
addCriteria(criteriaMap, "Cancel Reference No", cmdForm.getCancelReferenceNo());
addCriteria(criteriaMap, "Account Balance Type", cmdForm.getAccountBalanceType());
addCriteria(criteriaMap, "Trade Status", cmdForm.getStatus());
addCriteria(criteriaMap, "Data Source", cmdForm.getDataSource());

params.put("criteriaMap", criteriaMap);
return params;
}

```

- Create another method addCriteria into the controller to put the value.

```

private void addCriteria(Map<String, String> map, String key, Object value) {
    if((value != null) && (!Strings.isNullOrEmpty(value.toString()))) {
        map.put(key, value.toString());
    }
}

```

3.4.6.2.10 finch Theme Support

Introduction

Theme feature of finch allows application to change user interface look-n-feel at run-time.

How To Use

To add a theme with an application, following steps to be performed sequentially:

1. Add the following snippet of code inside its web.xml file:

```
<filter>
    <filter-name>finchThemeFilter</filter-name>
    <filter-class>com.nrift.finch.inf.web.filter.FinchThemeFilter</filter-class>
</filter>

<!-- Finch Theme filter mapping -->
<filter-mapping>
    <filter-name>finchThemeFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Important Note:

Insert the mapping entry after all other filter-mapping entries in the **web.xml** file, especially after the mapping of **spring-security** filter.

2. Select a name for your theme, for example ZANGO.

3. Create the following folder structure inside dist module:

```
src/main
  |_src/main/styles
    |_src/main/styles/theme
      |_src/main/styles/theme/ZANGO
```

4. Create all the .css files inside this directory and then create the main wrapper .css file inside this directory and call other css files from it.

5. Add the following lines of code snippet inside maven assembly file:

```
<!-- style related copy operation -->
<fileSet>
    <directory>src/main/styles/theme/${theme_name}</directory>
    <outputDirectory>styles/theme/${theme_name}</outputDirectory>
</fileSet>
```

6. Add a row inside the INF_CONSTRAINT_VALUES table in DB, where the CONSTRAINT_VALUE column represents the actual theme name (here it is ZANGO). The display name, which will be shown in the UI, should be inserted in the UI_DISPLAY_VALUE column.

7. Create a properties file with the same name of the theme name (here it is ZANGO.properties) inside the directory src/main/resources/web.

8. Put the following code snippet inside the properties file:

```
finch.inf.style = "put the path of your inf style class" e.g
"resources/styles/theme/ZANGO/ZANGO-inf-main.css"
finch.dbd.style = "put the path of your dbd style class" e.g
"resources/styles/theme/ZANGO/ZANGO-dbd-main.css"
```

9. The application log-in page css can also be changed by writing the following code snippet:

```
<bean class="org.springframework.web.servlet.theme.SessionThemeResolver" id="themeResolver"
  p:defaultThemeName="${theme_name}" />
```

where theme_name is the newly created theme.

Note:

If any application doesn't want to use any theme and not even wants to display the theme drop-down inside the user preference screen, it is to be ensured that there is no entry related with theme in META-INF/pref/app-pref.xml file of the application code base.

3.4.6.2.11 Bulk Action Wizard

Introduction

Bulk Action Wizard in finch is designed to perform bulk actions based on criteria as specified by the user. This is a hybrid implementation of [query](#) and [wizard](#) design flow. User can perform query related actions like providing criteria, personalization of criteria screen, save the query, modifying the result preference, filtering the result columns etc. User can also get the flavor of wizard during the entire flow while navigating between the screens.

How to Implement

It is not a part of finch. The following sections describe the implementation process.

User Interface Configuration

views.xml

```
views.xml

<definition name="tradeCancelQueryCriteria" extends="wizard-query-page">
    <put-attribute name="wizard-query-page" value="/WEB-INF/views/trd/tradeCancelQueryCriteria.jspx"/>
</definition>

<definition name="tradeCancelQueryResult" extends="wizard-result">
    <put-attribute name="wizard-query-page">
        <definition extends="wizardQueryResult">
            <put-attribute name="resultGrid" value="/WEB-INF/views/trd/tradeCancelQueryResult.jspx"/>
        </definition>
    </put-attribute>
</definition>

<definition name="tradeCancelQueryUserConfirm" extends="wizard-result">
    <put-attribute name="wizard-query-page">
        <definition extends="wizardQueryResult">
            <put-attribute name="resultGrid" value="/WEB-INF/views/trd/tradeBulkCxIUserConfirm.jspx"/>
        </definition>
    </put-attribute>
</definition>

<definition name="tradeCancelQuerySystemConfirm" extends="wizard-result">
    <put-attribute name="wizard-query-page">
        <definition extends="wizardQueryResult">
            <put-attribute name="resultGrid" value="/WEB-INF/views/trd/tradeBulkCxISystemConfirm.jspx"/>
        </definition>
    </put-attribute>
</definition>
```

For query criteria page

tradeCancelQueryCriteria.jspx

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:spring="http://www.springframework.org/tags"
    xmlns:form="http://www.springframework.org/tags/form"
    xmlns:tiles="http://tiles.apache.org/tags-tiles"
    xmlns:util="urn:jsptagdir:/WEB-INF/tags/util"
    xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">

    <!-- load required scripts -->

    <spring:eval expression="T(java.lang.System.currentTimeMillis())" var="myId"/>
    <input type="hidden" id="${myId}" />

    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:body>
            var FINCH$Wizard$onPageLoad = function(){

                <!-- load localized scripts -->

                <!-- load scripts -->

                <!-- do stuff for successful scripts loading -->

                <!-- perform client side validation -->

                <!-- register hooks for button -->
                var $finch$wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');
                $finch$wizard.apply({exitUri: FINCH.context.path});

            };
        </jsp:body>
    </jsp:element>

    <!-- form content -->

    <!-- load scripts-->

    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:attribute name="src">
            <c:url value="/scripts/inf/finch-wizard-form.js"/>
        </jsp:attribute>
        <jsp:body></jsp:body>
    </jsp:element>
```

For redirecting from the last wizard page to the query result, please comment or delete the above:

```
$finch$wizard.apply({exitUri: FINCH.context.path});
```

and in system confirmation page add the following:

tradeBulkCxISystemConfirm.jspx

```
var FINCH$Wizard$onPageLoad = function(){
    // do stuffs
    var $finch$Wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');
    $finch$Wizard.apply({ok: '/trd/bulkCancel'});
};
```

Note:

Please remember the URL for OK must be the base url for the wizard.

For query result page

tradeCancelQueryResult.jspx

```
<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:fn="http://java.sun.com/jsp/jstl/functions"
    xmlns:spring="http://www.springframework.org/tags"
    xmlns:util="urn:jsptagdir:/WEB-INF/tags/util"
    xmlns:tiles="http://tiles.apache.org/tags-tiles"
    xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">

    <jsp:directive.page contentType="text/html;charset=UTF-8" />
    <jsp:output omit-xml-declaration="yes" />
    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:body>
            var FINCH$Wizard$onPageLoad = function(){

                <!-- load localized scripts --&gt;

                <!-- load scripts --&gt;

                <!-- on successful loading of scripts, render grid --&gt;
                <!-- change the custom button name for next page --&gt;
                $('div.wizCustom1 input').attr('value',SAMPLE$TRD$i18n.label.cancel);
                <!-- perform client side validation if any --&gt;

                <!-- register hooks for buttons --&gt;
                var $finch$Wizard = jQuery('#' + ${commandForm.uniqueId}).parent().parent().data('FINCH$Wizard');
                $finch$Wizard.register('custom/1',authorize);

                <!-- deregister hooks --&gt;
                var unloadHook = function(){
                    $finch$Wizard.deregister('custom/1',authorize);
                }
                <!-- define mappings for custom buttons --&gt;

                $finch$Wizard.apply({customAction1: 'custom/1'});
                // $finch$Wizard.apply({customAction2: 'custom/2'});
                // $finch$Wizard.apply({customAction3: 'custom/3'});
            &lt;/jsp:body&gt;
        &lt;/jsp:element&gt;
    &lt;/div&gt;</pre>
```

Server Side Configuration

Command Form

The command form must extend the FinchWizardQueryCommandForm :

```
public class TrdBulkCancelCommandForm extends FinchWizardQueryCommandForm
```

The constructor must contain the wizard related configurations:

```

public TrdBulkCancelCommandForm(){
    Map<FinchWizardMode, String> page0tiles = new HashMap<FinchWizardMode, String>();
    page0tiles.put(EDIT, "tradeCancelQueryCriteria");
    page0tiles.put(VIEW,"tradeCancelQueryResult");
    page0tiles.put(USER_CONFIRMATION, "tradeCancelQueryUserConfirm");
    page0tiles.put(SYSTEM_CONFIRMATION, "tradeCancelQuerySystemConfirm");

    FinchWizardPage[] pages = new FinchWizardPage[1];
    Map<FinchWizardMode, FinchWizardButton[]> page0ExBttns = ImmutableMap.of(VIEW, new
    FinchWizardButton[] {OK});
    pages[0] = new FinchWizardPage("trd.tradequeryaction.label.tradeqry", page0tiles, null, page0ExBttns);

    Map<FinchWizardMode, FinchWizardButton[]> buttons = ImmutableMap.of(EDIT, new FinchWizardButton[]
    {RESET,SUBMIT},
        VIEW, new FinchWizardButton[] {BACK, RESET, CUSTOM1,OK},
        USER_CONFIRMATION, new FinchWizardButton[] {BACK,CONFIRM},
        SYSTEM_CONFIRMATION, new FinchWizardButton[] {OK}
    );
}

FinchWizardMode[] modes = new FinchWizardMode[]
{EDIT,VIEW,USER_CONFIRMATION,SYSTEM_CONFIRMATION};

FinchWizard finchWizard = new FinchWizard("trd.tradequeryaction.label.tradeqry", "wizard-query",
"trd/bulkCancel", modes, pages, buttons);
setWizard(finchWizard);
}

```

Note:

The above four modes and buttons must be defined as described above in order to achieve proper implementation of the feature.

In addition to the above configurations, the various attributes along with getter and setter must be present in the application,

Controller

The Controller must extend the FinchQueryWizardController<C> and must implement the MessageSourceAware

TrdBulkCancelController.java

```

@Controller
@RequestMapping(value = "/trd/bulkCancel/**")
@SessionAttributes({"commandForm"})
public class TrdBulkCancelController extends FinchQueryWizardController<TrdBulkCancelCommandForm>
implements MessageSourceAware

```

The application must override the implementation of the following:

- 1) **doInit(CommandForm form, FinchModelMap map)** - called during initialization of first screen
- 2) **doSubmit(CommandForm form, int index, FinchModelMap map)** - called on click of SUBMIT
- 3) **doReset(CommandForm form, int index)** - called on click of RESET
- 4) **doConfirm(CommandForm form, FinchModelMap map)** - called on click of CONFIRM
- 5) **getReportParameters(CommandForm form)** - called for getting the report generation parameters
- 6) **getComponent()** - get the current component

- 7) `getReportName()` - called for providing the report name.
- 8) `doCustomAction(CommandForm form, FinchModelMap map, int buttonNo)` - called for providing custom implementation for specific button
- 9) `getDateParams()` - called for passing the date parameters in query criteria

Sample implementation of `doCustomAction()` for custom button-1 is given below :

```
TrdBulkCancelController.java

@Override
protected void doCustomAction(TrdBulkCancelCommandForm form,
    FinchModelMap map,int buttonNo) throws FinchException {
switch(buttonNo) {
    case 1:
        cancelAction(form,map); // action for custom button 1
        break;
    case 2:
        // action for custom button 2
        break;
    case 3:
        // action for custom button 3
        break;
    default:
        break;
}
}
```

Please refer to the above files for detailed implementations.

3.4.6.2.12 UI Data Exchange

Introduction

finch UI data exchange framework provides infrastructure for exchanging UI data from one UI component to another in flexible way. finch provides cached bus internally to provides this functionality; but developer need not to know the internal working for using it in their application.

How to Use

finch provides abstraction of UI data exchange through `finch-exchange.js`. Developer gets a particular bus form `FINCH$Exchange` like:

```
finch data exchange bus

var dataExchange = FINCH$Exchange.get('excahnge_id');
```

Currently basic one target UI element is supported and finch is using this functionality for every popup query internally. With this new data exchange framework, now onward, options argument in `openPopUpForm` method is enhanced with id and tgt properties like:

popup options

```
var setting = {'id': 'accountNo',
    'title' : SAMPLE$REF$i18n.title.accPopup,
    'url' : '/ref/account/popup/query',
    'extUrlParams' : prepareAccDependentParams($accountNoBtnObj),
    'tgt': '#accountNo'};
```

Where tgt is the target to transfer the data and id is the key for the bus. Developer need not to put any tgt attribute in any HTML DOM element in their form

3.4.6.2.13 Referer Filter

Introduction

Referer Filter of finch application prevents application's internal URI's to get invoked when the same is being accessed from the browser's URL field. By default, when a user tries to access the URL by typing the same in the browser's URL field, the user would be redirected to the landing page. However this redirection is configurable, also the application can configure so that specific requests wont be redirected i.e. the request when invoked from the browser's URL field, the request would be honored.

Configuration

To setup the Referer Filter, specify the filter in the web.xml file of the application:

```
<filter>
<filter-name>referrerFilter</filter-name>
<filter-class>com.nrft.finch.inf.web.filter.ReferrerFilter</filter-class>
<init-param>
<param-name>exclude</param-name>
<param-value>.htm</param-value>
</init-param>
<init-param>
<param-name>redirect</param-name>
<param-value>/redirect-page</param-value>
</init-param>
</filter>

<filter-mapping>
<filter-name>referrerFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

Note:

In the above code, there are two configuration optional parameters as listed in the following table:

Parameter Name	Description
exclude	User can specify the resources which the Referer Filter will ignore, for example specifying .htm , /push, /report, /exportHoliday as the value. The filter would check whether the URL contains the specified values. If they are found, then the request would not get blocked when the user tries to access the same from the browser's URL field. If more than one value is specified then they need to be comma separated.

redirect	The user can specify the redirection path. This path can be a relative path or a fully qualified URL. To specify a relative path, one needs to prefix the path by '/'.
----------	--

Note:

1. In order to use notification when Referer Filter is enabled, one needs to exclude the path '/push' by specifying it in the value for the exclude parameter.
2. In order to use reports when Referer Filter is enabled, one needs to exclude the path '/report' by specifying it in the value for the exclude parameter.

3.4.6.2.14 Office Closing

Office Closing for UI Screen

Introduction

Firstly, finch would search for the closing status based on the screen id and component id. If the closing status is OPEN then the screen would be accessible, if the closing status is CLOSE then the screen would not be accessible, if the closing status value is neither OPEN or CLOSE then the screen would be accessible. However if there is no record, then the closing status based on the component id would be evaluated. If the closing status is OPEN then the screens of the component would be accessible, if the closing status is CLOSE then the screens of the component would not be accessible, if the closing status value is neither OPEN or CLOSE then the screens of the component would be accessible.

API Change

API's in ClosingStatusService has been refactored in ApplicationDateService class

List of API's:

```
public Date getApplicationDate(String componentId) throws FinchException
public String getSystemDate() throws FinchException
public String getUserFormattedApplicationDate(String componentId) throws FinchException
```

Interceptor Configuration

To configure the OnlineAccessRestrictionInterceptor one needs to do the following in webmvc-config.xml file

```
<mvc:interceptor>
<mvc:mapping path="/**"/>
<mvc:exclude-mapping path="/inf/personalizationTransfer/export**" />
<bean class="com.nrft.fin.ch.inf.web.interceptor.OnlineAccessRestrictionInterceptor" />
</mvc:interceptor>

<bean class="com.nrft.fin.ch.inf.web.security.FinchMappingExceptionResolver"
p:defaultErrorView="uncaughtException">
<property name="exceptionMappings">
<props>
<prop key=".HttpSessionRequiredException">index</prop>
<prop key="java.lang.Exception">uncaughtException</prop>
<prop key="com.nrft.fin.ch.inf.OfficeCloseException">officeCloseException</prop>
</props>
</property>
<property name="order" value="1"/>
</bean>
```

Note: Where there is direct download of data (as save file triggered by browser), those actions are not considered as candidate for screen closing for office closing functionality. Hence for this reason, those kind of resource need to be exclude during the

interceptor configuration. For example, in case of personalisation export this needs to be excluded from the interceptor configuration.

```
<mvc:exclude-mapping path="/inf/personalizationTransfer/export***" />
```

Database Changes

INF_UI_SCREEN

A new field called COMPONENT_ID has been introduced. Since this field is not nullable, the initial value is set as INF. Application needs to update this field as accordingly.

INF_APPLICATION_DATE

This is a new table which has been introduced for handling application date. Data needs to be setup for this table by the application. Note the values for the field APP_MODE are as follows:

OL = Online

BT = Batch

SV = Service (This is not handled in finch)

ENTERPRISE_ID , COMPONENT_ID and APP_MODE field acts as composite primary key.

The value for ENTERPRISE_ID and COMPONENT_ID should be a valid value as defined in INF_ENTERPRISE, INF_COMPONENT respectively

INF_CLOSING_STATUS

This is a new table (existing table has been droped from the schema) which has been introduced for handling closing status. Data needs to be setup for this table by the application.

In case of SCREEN_ID is not NULL then ENTERPRISE_ID , COMPONENT_ID, APP_MODE and SCREEN_ID is unique

The value for ENTERPRISE_ID and COMPONENT_ID should be a valid value as defined in INF_ENTERPRISE, INF_COMPONENT respectively

The values specified in the STATUS field are:

OPEN = Status is open

CLOSE = Status is closed

If someother value is mentioned in the STATUS field it would be considered as OPEN. However this is not recommended during data setup.

Note the values for the field APP_MODE are as follows

OL = Online

BT = Batch

SV = Service (This is not handled in finch)

INF_ACTION

A new field called COMPONENT_ID has been introduced. Since this field is not nullable, the initial value is set as INF. Application needs to update this field as accordingly. Currently closing status does not consider this field

Office Closing for Console Batch Process

INF_BATCH_MASTER

A new field called COMPONENT_NAME has been introduced. The field is nullable. This field should have a valid component name when the data needs to be filled by the application.

INF_CLOSING_STATUS

Finch will search the INF_CLOSING_STATUS table with the COMPONENT_NAME specified in INF_BATCH_MASTER and with app_mode as 'BT'. If no data is found the batch will be allowed to run, if data is present and STATUS is 'OPEN' then it will still be allowed to run. However if the STATUS is 'CLOSE' then the batch will be stopped from executing. Currently office closing is implemented for batch process as per component and not per batch.

3.4.6.2.15 Batch with Web UI

Introduction

This page describes how to connect batch with web UI

Steps to Connect Batch from Web UI

Following are the steps

1. Need to setup database tables as described in [Batch Process](#) in step 4 Data setup for Batch Job.
 - a. Insert Values in the following tables
 - i. INF_BATCH_MASTER - Contains the batch name and status
 - ii. INF_BATCH_ROLE_PCPT - Contains the role for the particular batch.
2. Include **batch-context.xml** in application-context.xml.
3. Create an xml file to run a batch job(e.g sample-job.xml in sample-app) which will be available to Spring Context at run-time.
4. Put the job id in the above xml file and the job id in xml should be exactly the same as the id attribute specified in INF_BATCH_MASTER table as well as in Batch annotation. For More please refer to [Batch Process](#) page.
5. Create a java file which extends ConsoleMetaData of finch and also put @Batch annotation on top of the class name and put the id name same as job id name declared in xml file as step 4. Below is the example

```
package com.nrft.sample.trd.batch;
import org.kohsuke.args4j.Option;
import com.nrft.finch.inf.batch.metadata.Batch;
import com.nrft.finch.inf.console.batch.ConsoleMetaData;
@Batch(id="sampleDBtoCVSBatch")
public class SampleDBtoCVSBatch extends ConsoleMetaData {
    @Option(name="-f", usage="console.usage.filename.fn", required=true)
    private String file;
    public String getFile() {
        return file;
    }
    public void setFile(String file) {
        this.file = file;
    }
}
```

6. Put @Option annotation on top of the property name of your java file for sample batch. You can mention "name", "usage" and "required" in this annotation.

```
@Option(name="-f", usage="console.usage.filename.fn", required=true)
```

7. You can also put @UiMetadata annotations on top of the field name as below. Here you can mention the displayName, Type etc.

```
@UiMetadata(displayName="File", displayType=DisplayTypes.Dropdown, isDisplayAble=true)
```

Description of UiMetadata annotation

Following is the description of UIMetadata annotations:

User can put the following fields in @UiMetadata annotations

isDisplayAble by default false user can put true to display the field in UI

displayName by default "" user need to put some value to display the name

displayOrder by default 0 user need to put the value to maintain the order of the fields

displayType by default TEXTBOX user can declare any other displaytype(CHECKBOX, DROPDOWN, TEXTBOX, FILE, DATE)

userConstraintName by default "" will be used in future to generate dynamic dropdown value

```
generator by default DropDownDataGenerator.class  
dateFormat by default "yyyyMMdd" user can put any other type of date format
```

8. Run the BatchRestServer before run batch from UI.To run BatchRestServer refer to Running a Batch Job using UI section of [Batch Process](#) page

3.4.6.2.16 Form field focus on error

Introduction

finch provides client the feature to place the focus on error field once an error has been encountered. By default, in finch when an error is encountered, the focus is shifted to the first input field except for errors which occurs as a result of blur event on date fields. For errors on blur event in date fields, the focus remains on the respective date field. Focus on error field will be placed on the following:

- For more than one error, the focus will be placed on the first error field as described by the client while preparing the validation list.
- For one error, the focus will be placed on the respective error field.

How to configure?

- 1) Application client has to set the flag whether to set the focus on error field in app-init-config.js.

If the application-developer wants that the focus should be placed on the first error field, then:

```
app-init-config.js  
FINCH.focusFirstError = true;
```

If the application-developer wants that the focus should be placed on the first field, then:

```
FINCH.focusFirstError = false;
```

- 2) Build the client side validation map with name attribute as key and message as value. For example, for the below mentioned input:

tradeGeneralEntry.jspx

```
<div class="formItem clientAccountItem">
    <form:label path="commandForm.tradeDTO.accountNo" class="required"><spring:message
code="trd.tradeentryaction.label.accountno" htmlEscape="false"/></form:label>
    <span><form:input id="accountNo" path="commandForm.tradeDTO.accountNo" class="textBox"/></span>
    <div class="popupBtn">
        <input id="accountPopupBtn" type="button" class="popupBtnCo amendReadOnlyPopup" popType="query"
popFor="cpAccount" actTypeContext="T|B" actCPTYPEContext="CLIENT|BROKER|INTERNAL"
actStatusContext="OPEN" value="" />
    </div>
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
```

the validation map must be build as:

tradeGeneralEntry.jspx

```
var FINCH$Wizard$onPageLoad = function(){
    FINCH.loadScript([
        // paths for the scripts
    ],{
        success : function() {
            function validate() {
                var messageFieldMap = new Object();
                if(VALIDATOR.isNullValue($.trim(accountNo))){
                    messageFieldMap['tradeDTO.accountNo'] =
SAMPLE$TRD$i18n.tradeEntry.generalinfo.accountno_empty;
                }
                // other validations
                if ($.isEmptyObject(messageFieldMap) !== true){
                    validationMessages.push(messageFieldMap);
                    FINCH.postNotice(FINCH.notice.type.error, validationMessages, true);
                    return false;
                }
                return true;
            }
        }
    })
}
```

3) Build the server side validation map with path attribute as key and exception key as message in controller. In controller, the field that needs to be focused on error, exception must be added to the FormFieldValidationException as given below :

TradeEntryController.java

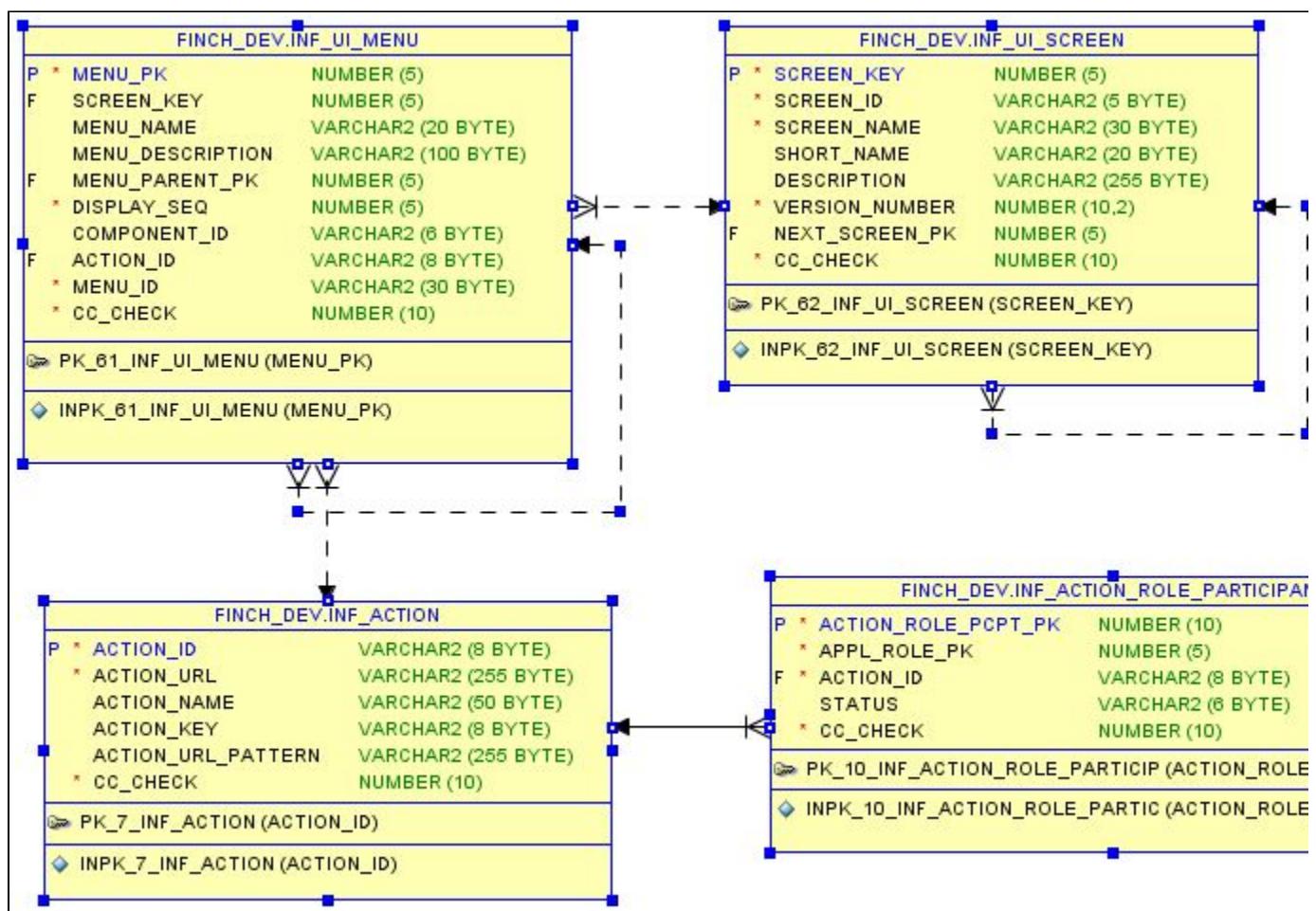
```
private void validatePage1(TradeEntryCommandForm form)
    throws DataInputException {
    try {
        // Account No. Validation
        Account account = accountService.resolveAccountNoWithType(tradeDTO.getAccountNo());
        form.setAccountPk(account.getAccountPk());
    } catch (RefException e) {
        log.error("Account validation failed", e);
        validAccounts = false;
        exceptions.add(new FormFieldValidationException(e, "tradeDTO.accountNo"));
    }
    // other exceptions
    if (exceptions.size() > 0) {
        throw new DataInputException("", exceptions);
    }
}
```

3.4.6.2.17 Add new Menu/Action

Introduction

Menu in **finch** is a hierarchical tree structure, provides a platform to perform several operations in the application. Structure of menu has the parent-child relationship. Parent menu has associated anchors for each child menu. Parent menu uses this associated anchor in order to navigate to the child menu. The following diagram displays the finch Menus with Access Control:

finch Menu Access Control:



Setup

Menu Definition

In finch, the menu definition is stored in a database table named INF_UI_MENU. The following table displays the example:

MENU_PK	SCREEN_KEY	MENU_NAME	MENU_DESCRIPTION	MENU_PARENT_PK	DISPLAY_SEQ	COMPONENT_ID	ACTION_ID	MENU_ID	CC_CHECK
1	(null)	Main Menu	Main Menu	(null)	1	(null)	(null)	ROOT	1
2	(null)	Trade	Parent of Trade	1	1	TRD	(null)	TRD	1
3	1	Entry	Trade Entry	2	1	TRD	TRDTEN	TRDEN	1
4	2	Query	Trade Query	2	2	TRD	TRDTQR	TRDQR	1

- The 1st row is the root menu. Since it is the root menu, it does not have the MENU_PARENT_PK and its DISPLAY_SEQ as 1. Since it will not navigate to any screen, so ACTION_ID and SCREEN_KEY is null.
- The 2nd row is the Trade menu. Since it is the child of Main menu, its MENU_PARENT_PK is 1. Its DISPLAY_SEQ = 1 implies that it is the first child of Main menu. Its null value in ACTION_ID and SCREEN_KEY implies that it does not navigate to any screen.
- The 3rd row is the Trade Entry menu. Since it is the child of Trade menu, its MENU_PARENT_PK is 2. Its DISPLAY_SEQ = 1 implies that it is the first child of Trade menu. Its corresponding value in ACTION_ID and SCREEN_KEY signifies that this menu navigates to a particular screen.
- The 4th row is the Trade Query menu. Since it is the child of Trade menu , its MENU_PARENT_PK is 2. Its DISPLAY_SEQ = 2 implies that it is the second child of Trade menu. Its corresponding value in ACTION_ID and SCREEN_KEY signifies that this menu navigates to a particular screen.

Important Note:

The 1st row in the above example must be created in the application database before creating any new menu.

Screen Key Definition

In finch, the screen key definition is stored in a database table named INF_UI_SCREEN. The following table displays the example:

SCREEN_KEY	SCREEN_ID	SCREEN_NAME	SHORT_NAME	DESCRIPTION	VERSION_NUMBER	NEXT_SCREEN_PK	CC_CHECK
1	TRDEN	Trade Entry	Trade Entry	Trade Entry	2	4	1
2	TRDQR	Trade Query	Trade Query	Trade Query	2	3	1
3	TRDRS	Trade Query Result	Trade Query Result	Trade Query Result	2	null	1
4	TENUC	Trade Entry User Confirmation	Trade Entry UC	Trade Entry User Confirmation	2	null	1

- The 1st row contains the definition of Trade Entry screen. The NEXT_SCREEN_PK = 4 signifies that the next screen to be fetched is the Trade Entry User Confirmation.
- The 2nd row contains the definition of Trade Query screen. The NEXT_SCREEN_PK = 3 signifies that the next screen to be fetched is the Trade Query Result.

Basic Action Definition

In finch, the basic action definition is stored in a database table named INF_ACTION. The following table displays the example:

ACTION_ID	ACTION_URL	ACTION_NAME	ACTION_KEY	ACTION_URL_PATTERN	CC_CHECK
TRDTEN	trd/entry/init	Trade Entry	1	/trd/entry/.*	1
TRDTQR	trd/query/init	Trade Query	2	/trd/query/.*	1

- The 1st row defines the action related details for Trade Entry.
- The 2nd row defines the action related details for Trade Query.

Application Role and Action Mapping

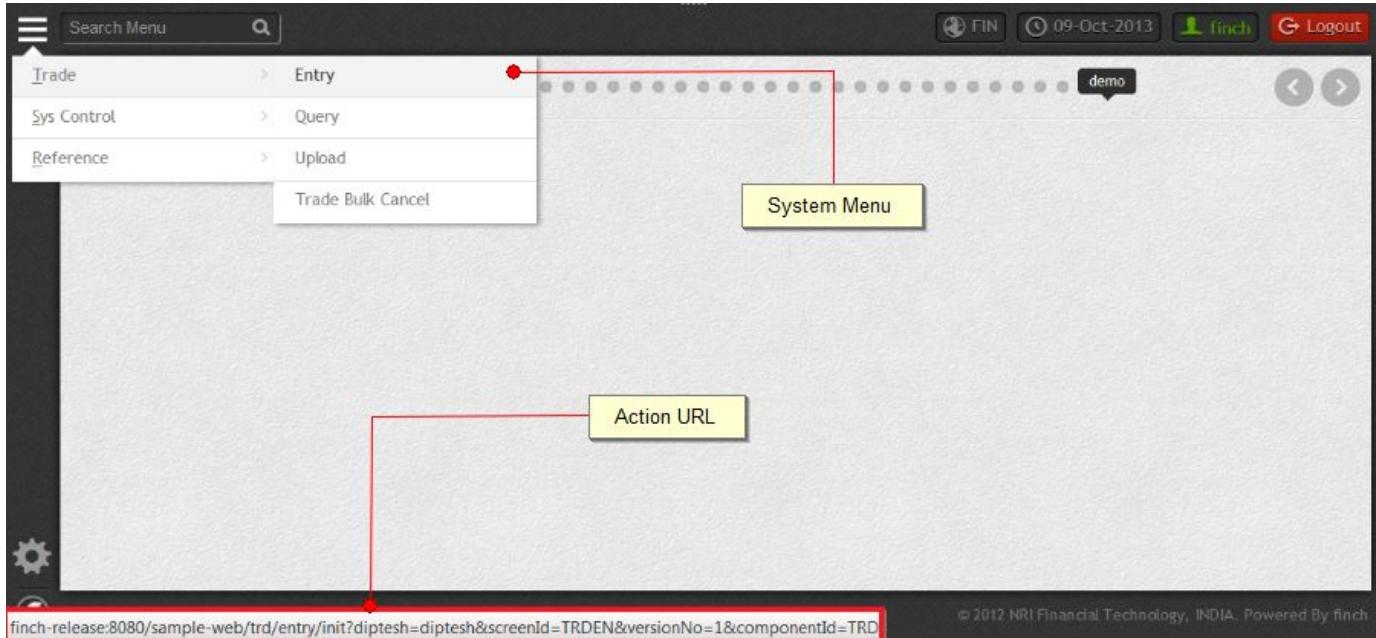
In finch, the mapping between the application role and action is stored in a database table named INF_ACTION_ROLE_PARTICIPANT. The following table displays the example:

ACTION_ROLE_PARTICIPANT_PK	APPL_ROLE_PK	ACTION_ID	STATUS	CC_CHECK
1	1	TRDTEN	NORMAL	1
2	2	TRDTQR	NORMAL	1

- The 1st row defines that it's the Main Menu, Trade Menu and Trade Entry is visible to the user whose application role is 1.
- The 2nd row defines that the Trade Query is visible to user whose application role is 2.

Access to Menu

User having Application Role 1, only Trade Entry screen is accessible as shown below:



The action URL fetched from INF_ACTION is mapped with a controller which does the business processing.

Important Note:

The first level menu can not have a leaf menu (clicking on that menu opens a screen). It can have a menu that have one or more child menu.

Example to set up Menu

Now we will see an example to set up menu for sample application. Sample application do no need to insert the Main Menu in INF_UI_MENU table, as finch provide this entry in INF_UI_MENU table. Sample application only need to

insert the menu entry they want.

INF_UI_MENU table

```
insert into INF_UI_MENU
(
  MENU_PK,SCREEN_KEY,MENU_NAME,MENU_DESCRIPTION,
  MENU_PARENT_PK,DISPLAY_SEQ,COMPONENT_ID,ACTION_ID,
  MENU_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,
  CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK
)
values (
<VALUE_FOR_MENU_PK>,<VALUE_FOR_SCREEN_KEY>,<VALUE_FOR_MENU_NAME>,<VALUE_FOR_MENU_DESCRIPTION>,
<VALUE_FOR_MENU_PARENT_PK>,<VALUE_FOR_DISPLAY_SEQ>,<VALUE_FOR_COMPONENT_ID>,<VALUE_FOR_ACTION_ID>,
<VALUE_FOR_MENU_ID>,<VALUE_FOR_APP_REGI_DATE>,<VALUE_FOR_APP_UPD_DATE>,<VALUE_FOR_CREATED_BY>,
<VALUE_FOR_CREATION_DATE>,<VALUE_FOR_UPDATED_BY>,<VALUE_FOR_UPDATE_DATE>,<VALUE_FOR_CC_CHECK
);
);
```

For example - If application has a menu 'xyz' and under 'xyz' there are two child menus 'abc' and 'pqr' then following scripts needed to run.

In INF_ACTION table leaf menu(clicking on that menu opens a screen) value is to be inserted. For this example need to insert two values for abc and pqr.

we have assigned ACABC action id for abc and ACPQR for pqr.

INF_ACTION table

```
INSERT INTO INF_ACTION
(ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,
APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,
UPDATED_BY,UPDATE_DATE,ACTION_URL_PATTERN,CC_CHECK,COMPONENT_ID
)
VALUES
('ACABC','/trd/entry/init','Trade Entry','1',
NOW(),NOW(),'FINCH-SAMPLE',NOW(),
'FINCH-SAMPLE',NOW().'/trd/entry/.*',1,'TRD'
);
INSERT INTO INF_ACTION
(ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,
APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,
UPDATED_BY,UPDATE_DATE,ACTION_URL_PATTERN,CC_CHECK,COMPONENT_ID
)
VALUES
('ACPQR','/trd/entry/init','Trade Entry','1',
NOW(),NOW(),'FINCH-SAMPLE',NOW(),
'FINCH-SAMPLE',NOW().'/trd/entry/.*',1,'TRD'
);
```

After executing the above script Screen Key definition is to be inserted in INF_UI_SCREEN table.

For this example 1789 is defined for screen abc and 2789 for screen pqr

```
INSERT INTO INF_UI_SCREEN
(SCREEN_KEY,SCREEN_ID,SCREEN_NAME,SHORT_NAME,
DESCRIPTION,VERSION_NUMBER,NEXT_SCREEN_PK,APP_REGI_DATE,
APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
UPDATE_DATE,CC_CHECK,COMPONENT_ID
)
VALUES
(1789,'ABCID','Menu 1','Menu 1',
'Menu 1 desc',1,null,NOW(),
NOW(),'FINCH-SAMPLE',NOW(),'FINCH-SAMPLE',
NOW(),1,'TRD'
);
INSERT INTO INF_UI_SCREEN
(SCREEN_KEY,SCREEN_ID,SCREEN_NAME,SHORT_NAME,
DESCRIPTION,VERSION_NUMBER,NEXT_SCREEN_PK,APP_REGI_DATE,
APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,
UPDATE_DATE,CC_CHECK,COMPONENT_ID
)
VALUES
(2789,'PQRID','Menu 2','Menu 2',
'Menu 2 desc',1,null,NOW(),
NOW(),'FINCH-SAMPLE',NOW(),'FINCH-SAMPLE',
NOW(),1,'TRD'
);
```

Now insert value for xyz menu.

Insert value for 'xyz' menu

```
INSERT INTO INF_UI_MENU
(MENU_PK,SCREEN_KEY,MENU_NAME,MENU_DESCRIPTION,
MENU_PARENT_PK,DISPLAY_SEQ,COMPONENT_ID,ACTION_ID,
MENU_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,
CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK
)
VALUE
(987,null,'xyz','xyz menu drscription',
1,2,'INF',null,
'SYS', NOW(),NOW(),'FINCH-SAMPLE',
NOW(), 'FINCH-SAMPLE', NOW(),1
);
```

In the above script SCREEN_KEY and ACTION_ID is set to null because it will not navigate to any screen instead it will have some two child menu abc and pqr.

To have these two child menu under 'xyz' menu the following script need to run

```

INSERT INTO INF_UI_MENU
    (menu_pk, screen_key, menu_name, menu_description,
     menu_parent_pk, display_seq, component_id, action_id,
     menu_id, app_regi_date, app_upd_date, created_by,
     creation_date, updated_by, update_date, cc_check
    )
VALUES (9871, 1789, 'abc', 'abc screen',
        987, 1, 'INF', 'ACABC',
        'ABC', CURDATE(), CURDATE(), 'FDBA-60',
        NOW(), 'FDBA-60', NOW(), 1
       );
INSERT INTO INF_UI_MENU
    (menu_pk, screen_key, menu_name, menu_description,
     menu_parent_pk, display_seq, component_id, action_id,
     menu_id, app_regi_date, app_upd_date, created_by,
     creation_date, updated_by, update_date, cc_check
    )
VALUES (9872, 2789, 'pqr', 'pqr screen',
        987, 2, 'INF', 'ACPQR',
        'PQR', CURDATE(), CURDATE(), 'FDBA-60',
        NOW(), 'FDBA-60', NOW(), 1
       );

```

In INF_ACTION_ROLE_PARTICIPANT table following entries is to be made

INF_ACTION_ROLE_PARTICIPANT table	
INSERT INTO INF_ACTION_ROLE_PARTICIPANT (ACTION_ROLE_PCPT_PK,APPL_ROLE_PK,ACTION_ID,APP_REGI_DATE, APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY, UPDATE_DATE,STATUS,CC_CHECK) VALUES (3789,1,'ACABC',NOW(), NOW(),'FINCH-SAMPLE',NOW(),'FINCH-SAMPLE', NOW(),'NORMAL',1);	INSERT INTO INF_ACTION_ROLE_PARTICIPANT (ACTION_ROLE_PCPT_PK,APPL_ROLE_PK,ACTION_ID,APP_REGI_DATE, APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY, UPDATE_DATE,STATUS,CC_CHECK) VALUES (4789,1,'ACPQR',NOW(), NOW(),'FINCH-SAMPLE',NOW(),'FINCH-SAMPLE', NOW(),'NORMAL',1);

Following is an example with sample Data to setup menu.

The below code demonstrates how to create simple parent menu and its related sub menu and its actions in MySQL. **Application need to change data according to there uses**

```

## INF_UI_MENU
Insert into INF_UI_MENU
(MENU_PK,SCREEN_KEY,MENU_NAME,MENU_DESCRIPTION,MENU_PARENT_PK,DISPLAY_SEQ,COMPONENT_ID,ACTION_ID,MENU_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK) values
(10014,null,'Test','Test',1,7,'INF',null,'TST',NOW(),NOW(),'FINCH-SAMPLE',NOW(),'FINCH-SAMPLE',NOW(),1);

## INF_ACTION
Insert into INF_ACTION
(ACTION_ID,ACTION_URL,ACTION_NAME,ACTION_KEY,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,ACTION_URL_PATTERN,CC_CHECK) values
('XXXUI','/inf/sample/init','Calendar
Initialisation','19',now(),now(),'FINCH-SAMPLE',now(),'FINCH-SAMPLE',now().'/inf/sample/*!',1);

## INF_ACTION_ROLE_PARTICIPANT
Insert into INF_ACTION_ROLE_PARTICIPANT
(ACTION_ROLE_PCPT_PK,APPL_ROLE_PK,ACTION_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,STATUS,CC_CHECK) values
(10032,1,'XXXUI',now(),now(),'FINCH-SAMPLE',now(),'FINCH-SAMPLE',now(),'NORMAL',1);

## INF_UI_SCREEN
Insert into INF_UI_SCREEN
(SCREEN_KEY,SCREEN_ID,SCREEN_NAME,SHORT_NAME,DESCRIPTION,VERSION_NUMBER,NEXT_SCREEN_PK,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK,COMPONENT_ID) values
(20,'XXXSI','Sample UI Test','Sample UI Test','Sample UI
Test',1,null,now(),now(),'FINCH-SAMPLE',now(),'FINCH-SAMPLE',now(),1,'INF');

## INF_UI_MENU
Insert into INF_UI_MENU
(MENU_PK,SCREEN_KEY,MENU_NAME,MENU_ID,MENU_DESCRIPTION,MENU_PARENT_PK,DISPLAY_SEQ,COMPONENT_ID,ACTION_ID,APP_REGI_DATE,APP_UPD_DATE,CREATED_BY,CREATION_DATE,UPDATED_BY,UPDATE_DATE,CC_CHECK) values
(19,20,'Sample Menu','XXXSI','Sample
UI',10014,1,'INF','XXXUI',now(),now(),'FINCH-SAMPLE',now(),'FINCH-SAMPLE',now(),1);
COMMIT;

```

After setup the menu data application needs to include full controller package name into the component scan path in webmvc-config.xml file as follows:

```
<context:component-scan base-package="<Include full controller package name>" />
```

Also include there service and repository full package name in application-context.xml file as follows

```
<context:component-scan
base-package="<Include full package name of service and repository>" />
```

Again include all the entity full package name in datasource-context.xml as follows

```

<bean id="global-em" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
    p:packagesToScan="<Include full Entity package name>"
    p:dataSource-ref="GLOBAL"
    p:jpaVendorAdapter-ref="jpaVendorAdapter" />
</beans>

```

3.4.6.2.18 Authentication

Introduction

Authentication process authorizes the user and ensures that an unauthorized user cannot access the application and its related resources. finch authorizes users with the help of [Spring security](#). Spring security is highly configurable provides support of multiple authentication provider. In the application, finch framework completely handles the authentication process.

Configuration

finch manages the authentication process by checking for a valid username and password from INF_EMPLOYEE database table.

Authentication Configuration Details

The configuration details for the authentication are described in the applicationContext-security.xml. The application may configure the authentication mechanism by modifying the configuration in the above mentioned XML file. Below is the brief snapshot of the configuration xml file.

```

applicationContext-security.xml

<form-login login-page="/login" login-processing-url="/resources/j_spring_security_check"
default-target-url="/" always-use-default-target="true"
authentication-success-handler-ref="authenticationSuccessHandler"
authentication-failure-url="/login?login_error=t"/>

```

Note:

The /resources/j_spring_security_check is the spring mechanism for login processing. The authenticationSuccessHandler is the bean to be called on authentication success. The authentication-failure-url is the url to trigger on authentication failure.

Grant and Deny Access

XML code snippet defines the URL patterns to grant access and deny access.

```

<intercept-url pattern="^/login.*$" access="permitAll" /> <intercept-url pattern="^/resources.*$"
access="permitAll" /> <intercept-url pattern="^/images/.*$" access="permitAll" /> <intercept-url
pattern="^/scripts/.*$" access="permitAll" /> <intercept-url pattern="^/styles/.*$" access="permitAll" />
<intercept-url pattern="^/WEB-INF/[^w*.xml$]" access="permitAll" /> <intercept-url
pattern="^/WEB-INF/modules/[a-z|a-zA-Z][a-z|a-zA-Z][a-z|a-zA-Z]/[^w*.xml$]" access="permitAll" /> <intercept-url
pattern="^.*$" access="denyAll" />

```

Log-out Mechanism

XML code snippet defines the log-out mechanism which is the default spring log-out mechanism used in finch.

```
<logout logout-url="/resources/j_spring_security_logout"/>
```

Session Management

XML code snippet defines the session management. It detects the invalid session and redirects to the login page for an invalid session.

```
<session-management invalid-session-url="/login" /> <session-management> <concurrency-control max-sessions="1" error-if-maximum-exceeded="true" /> </session-management>
```

Authentication Success Handler

XML code snippet contains the bean definition for authenticationSuccessHandler.

```
<beans:bean id="authenticationSuccessHandler" class="com.nrft.finch.web.security.FinchAuthenticationSuccessHandler"/>
```

Authentication Mechanism

XML code snippet defines the authentication mechanism.

The userDetailsService bean id contains the property name usersByUsernameQuery which contains the query to verify the username and password from INF_EMPLOYEE database table.

```
<!-- Configure authentication mechanism -->
<authentication-manager>
<authentication-provider user-service-ref='userDetailsService' />
</authentication-manager>

<!-- Configure user detail service -->
<beans:bean id="userDetailsService"
class="org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl">
<beans:property name="dataSource" ref="dataSource"/>
<beans:property name="usersByUsernameQuery"
value="select user_id, appl_passwd, 1 as locked from inf_employee where user_id = ?"/>
<beans:property name="authoritiesByUsernameQuery"
value="select e.user_id, r.application_role_name from inf_employee e, inf_emp_appln_role_participant erp,
inf_application_role r where e.employee_pk = erp.employee_pk and erp.appl_role_pk = r.appl_role_pk and
e.user_id = ?"/>
</beans:bean>
```

Note

The above authentication mechanism is the default implementation of finch. The application may configure the mechanism as per requirement.

3.4.6.2.19 Access Control

Introduction

Access Control in finch is the mechanism that asks the user " Do you have permission to use this resource ? ". It checks whether the logged in user has the access to different resources (like menu items, URL's) and let the user perform operations to the accessible resources. This mechanism has been introduced to classify different users based on their roles and responsibilities. finch provides the entire mechanism to control the access and the application does not need to modify the implementation.

Set Up

Each user in finch is assigned a set of application role. The application role defines what are the roles and responsibilities associated with that role. For more details on application role, refer to [Application Role](#). The access control mechanism looks up the **INF_ACTION** and **INF_ACTION_ROLE_PARTICIPANT** database table, as displayed below.

INF_ACTION

ACTION_ID	ACTION_URL	ACTION_NAME	ACTION_KEY	ACTION_URL_PATTERN
TRDTEN	/trd/entry/init	Trade Entry	1	/trd/entry/.*
TRDTUP	/trd/upload/init	Trade Upload	5	/trd/upload/.*

INF_ACTION_ROLE_PARTICIPANT

ACTION_ROLE_PCPT_PK	APPL_ROLE_PK	ACTION_ID
1	1	TRDTEN
2	2	TRDTUP

The above settings illustrates the following:

- An user with APPL_ROLE_PK = 1 – cannot access the menu item TRADE > UPLOAD and URL that contains the string " /trd/upload ".
- An user with APPL_ROLE_PK = 2 – cannot access the menu item TRADE > ENTRY and URL that contains the string " /trd/entry/ ".

Each and every action must be mapped with application role in order to gain access for the resource.

3.4.6.2.20 Application Role management

Introduction

finch provides Application Role management UI screen to manage application role. Currently application role management consists of application role name, menu management and widget feed management. Application can have their own custom widget, which may consists of feeds. If custom widgets consists of feed and application wants to control access of these feed using application role, then they need to implement com.nrft.finch.inf.domain.service.ApplicationRoleFeedService interface. This interface defines two public methods these are

- List<ApplicationRoleFeedMetadata> getFeedList(String widgetType) throws FinchException
This method will be called from finch to get list of feeds correspond to specific custom widget type. Value for 'widgetType' parameter will be same as defined in DBD_WIDGET_TYPE_NAME column in DBD_WIDGET_TYPE table. Application should return feeds corresponding to custom widget type passed as parameter.
- void saveFeedRole(long applicationRolePk, ApplicationRoleWidgetMetadata widget) throws FinchException
This method will be called from finch to allow application to save feed related information associated with application role. Application should use 'applicationRolePk' to identified unique application role and 'widget' to know about the widget that END USER has selected.
- void putSelectedFeedsForRole(long applicationRolePk, ApplicationRoleWidgetMetadata widget) throws FinchException
This method will be called from finch to allow application to populate information about feeds which are currently available for a particular application role. The ApplicationRoleWidgetMetadata will contain information about a particular widget including a complete list of all feeds (a List of ApplicationRoleFeedMetadata objects) for that widget that is returned by the getFeedList() method described above. The application has to iterate over the complete list and mark the feeds for a particular application role as selected by calling the setSelected(boolean b) method of ApplicationRoleFeedMetadata class.

- void updateFeedsForRole(long applicationRolePk, ApplicationRoleWidgetMetadata widget) throws FinchException
This method will be called from finch to allow the application to update the list of feeds available for a particular application role. The ApplicationRoleFeedMetadata object passed as argument will contain a List of feeds which need to be made available to a particular application role. The application can check which feeds are currently available to a particular application role and if the currently selected feeds is different from this new list of feeds, then the application can make the necessary updates to the database.
- void cancelFeedsForAppRole(long applRolePk) throws FinchException
This method will be called by finch when an Application Role is canceled. The argument applRolePk can be used to identify the application role being canceled. If any feed of a custom widget with feeds, is available to this application role, then the corresponding record must be canceled (or hard deleted) from the database.

3.4.6.2.21 Default Dropdown Values

Introduction

finch provides a method for setting the default value (i.e. the option that is chosen when the dropdown is first displayed) for a dropdown. It is possible to configure default value for each dropdown field individually. It is also possible to configure one application-wide default value which will apply to all dropdowns throughout the UI of the application. Localization is also supported.

Setting default value for individual fields

FINCH\$Default\$Dropdown function

A function named FINCH\$Default\$Dropdown needs to be defined in a JavaScript file and that file must be loaded on login of a user. This function should return an object containing key-value pairs for each dropdown for which a default value is desired.

FINCH\$Default\$Dropdown example

```
var FINCH$Default$Dropdown = function() {
    var defaultValue = {
        defaultBranch:'Select default Branch',
        applicationRole:'Select an Application Role',
        branchId:'Select a Branch',
        screenId:'Select a Screen',
        batchId:'Select a Batch',
        defaultBatchDropdown:'Choose a value',
        status:"",
        exitCode:"",
        taskGrp:"",
        timezone:"",
        sortCriteria:"",
        branch: "",
        applicationRoleName: ""
    };
    return defaultValue;
}
```

Loading the FINCH\$Default\$Dropdown function

The JavaScript file in which the FINCH\$Default\$Dropdown function is defined, must be loaded on user login. Say the code in the above example is written in the file app-dropdown-list-i18n_en.js. The code for loading this file should be written in the file app-init-config.js.

app-init-config.js

```
FINCH$Exchange.get('scriptLoaderBus').onValue(function(value) {  
    if(value === 'finchInitComplete') {  
        // other code  
        FINCH.loadLocalizedScript([{path: FINCH.context.path + '/scripts/app-dropdown-list-i18n.js', async: false}]);  
        // other code  
    }  
});
```

The above example shows how to load localized versions of the JavaScript file. But localization may not be required. Say the name of the file defining the FINCH\$Default\$Dropdown function is app-dropdown-list-i18n.js. If localization is not required then the JavaScript file may be loaded as shown below.

app-init-config.js

```
FINCH$Exchange.get('scriptLoaderBus').onValue(function(value) {  
    if(value === 'finchInitComplete') {  
        // other code  
        var scripts = [  
            {path: FINCH.context.path + '/scripts/app-dropdown-list-i18n.js', async: false},  
            // other scripts (if any)  
        ];  
        FINCH.loadScript(scripts, {  
            ordered: true,  
            success: function() {  
                FINCH$Exchange.get('scriptLoaderBus').push('appInitComplete');  
            }  
        });  
        // other code  
    }  
});
```

The defaultValue attribute

To get the values in the key-value pairs returned by the FINCH\$Default\$Dropdown function, the defaultValue attribute must be added to a dropdown.

defaultValue attribute example

```
<form:select id="defaultBranch" path="commandForm.employeeDTO.defaultBranchPk" class="dropdowninput"  
defaultValue="defaultBranch">  
    <form:option value="" />  
    <form:options items="${commandForm.branches}" />  
</form:select>
```

In the above example the value of defaultValue attribute is defaultBranch. When the screen having this example dropdown is loaded then the value defaultBranch will be used as a key to look for a value in the object returned by FINCH\$Default\$Dropdown function. The value that is found, will form the inner text of the `<option>` element with empty value. If no value is found then an empty option will become the default chosen option.

If the defaultValue attribute is given but there is no `<option>` element with empty value then the attribute will have no effect.

Example: Say the value for the key defaultBranch is 'Select..'. Then the dropdown will look like the following images.

Default Branch	<input type="button" value="Select.."/>
----------------	---

Default Branch	<input type="button" value="Select.."/>
End Date (UTC)	<input type="button" value="Select.."/> FBR

```
<select id="defaultBranch" name="employeeDTO.defaultBranchPk" class="dropdowninput" defaultvalue="de
  <option value="" title="Select..">Select..</option>
  <option value="2" title="FBR">FBR</option>
  <option value="3" title="TBR">TBR</option>
</select>
```

Example: Say the value for the key defaultBranch is ". Then the dropdown will look like the following images.

Default Branch	<input type="button" value="Select.."/>
End Date (UTC)	<input type="button" value="Select.."/> FBR

```
<select id="defaultBranch" name="employeeDTO.defaultBranchPk" class="dropdowninput" defaultvalue="defa
  <option value="" title=""></option>
  <option value="2" title="FBR">FBR</option>
  <option value="3" title="TBR">TBR</option>
</select>
```

Setting defaultValue by function call

It may happen that some dropdown is populated with values on some user action. For such dropdowns merely adding the defaultValue attribute and an <option> element with empty value is not sufficient. These two steps can ensure default value on initial load of the screen. But when the dropdown is repopulated on some user action then the default value set initially will be removed. For such dropdowns default value can be set by calling the defaultDropdownValue() function defined in finch-util-commons.js. The syntax for this function is

defaultDropdownValue(<jQuery object>);

Example: defaultDropdownValue(\$('#role'));

In this example, role is the id of a dropdown.

This function must be called each time the dropdown is repopulated. For the function to work defaultValue attribute and empty valued <option> must be present.

Setting Application-wide default value

For setting an application-wide default value the files keys.properties and app-pref.xml are used.

keys.properties
pref.key.application.defaultdropdown=DEFAULT_DROPDOWN

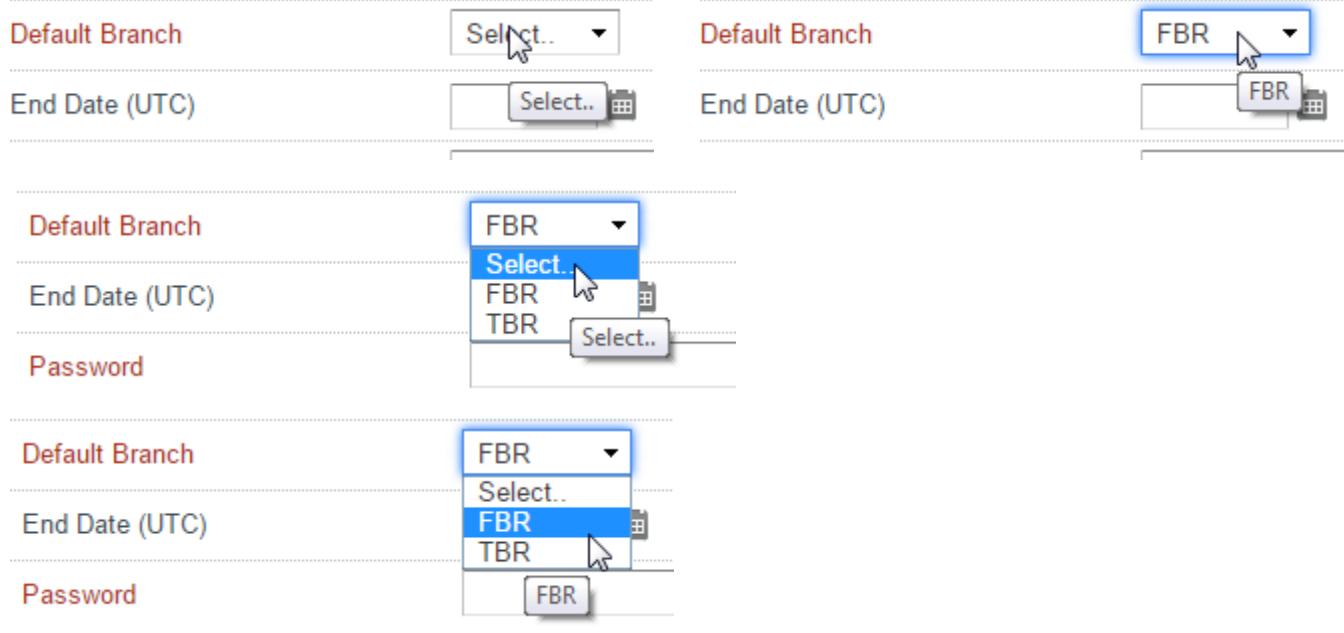
app-pref.xml
<DEFAULT_DROPDOWN>Select..</DEFAULT_DROPDOWN>

In this method localization is not possible.

If the FINCH\$Default\$Dropdown function is defined and loaded, then the value set in app-pref.xml is used if and only if the key given with defaultValue attribute is not found in the object returned by FINCH\$Default\$Dropdown function.

Tool tips for the dropdown and its options

Adding the defaultValue attribute has an additional effect besides setting the default value. It adds tool tips to the dropdown and each of the options. For the dropdown the tool tip shows the same text as the currently selected option. When the drop down is opened, then on hovering over any option the text of the option is displayed in the tool tip. This helps in reading options which are very long and get truncated by the border of the dropdown. On page load, event handler is attached to the dropdown which ensures that whenever a different option is selected the tool tip of the dropdown changes to match the option.



When the value of the dropdown is changed using JavaScript function call, then the handler for changing tool tips is not called. In such a situation, the tool tip needs to be changed manually.

Currently it is not possible to identify the instant when the effect of defaultValue attribute is completed. If there is a requirement to make any modifications to the dropdown or to fetch the value of the dropdown after this instant, then it is recommended to add the defaultValue attribute using JavaScript code and then call the defaultDropdown() function. In this way the application developer can be sure that the defaultValue and tool tips have been added before accessing or modifying the dropdown.

Example use case:

Say the application developer wants to use the selectmenu plugin of jQuery UI. This plugin masks the dropdown with a div which contains the options of the dropdown. The div looks and behaves like a dropdown. When the plugin is used on the dropdown it uses the options that are currently present in the dropdown.

If the plugin is used before the effect of defaultValue attribute has completed then the div created by selectmenu plugin will have stale options. Specifically, the topmost option will have an empty string instead of the expected default value string.

To avoid this problem its best if the defaultValue attribute is added using JavaScript code, and then the defaultDropdown() function is called, and then selectmenu plugin is used. This allows the application developer to have more control over the order in which modifications are made to the dropdown.

Entry Screen Customization

1. Scope

In iGV application, there are some entry based screens like “Trade Entry” where certain non-mandatory fields may be unnecessary in the context of different enterprises. Those fields can be removed by configuring it at the enterprise level by specifying it in the preference

store of the server where the application is deployed. In addition to it the order in which the fields must be displayed can also be configured through it. The customization is done solely on the client side, and relation between any dependent fields must be handled by the application. This document gives a detailed account of the implementation of the customization of an entry based screen.

2. Prerequisites

Refer to [Web Component](#) for an overall idea about the iGV web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Wizard Based Entry](#) for understanding the entry mechanism and screen layouts of the iGV wizard.

Refer to [Single Page Entry](#) for details on how to create an entry page.

3. Assumptions & Limitations

1. The customization feature is only applicable for screens based on “igv-wizard.js”.
2. The implementation will work with screens based only on the traditional form based layout and not on any table layout. Hence anywhere where the customization needs to be implemented the layout block needs to be in a form items based layout except for a particular scenario in case of a details page which is discussed later in the document.
3. Every page must have a designated unique screenId, versionNo and pageIndex(for entry based wizard screens, not for single page details view) for uniquely identifying it while applying the customization.
4. The screens for amend, cancel, copy all needs to be in the similar expected layout as that of an entry screen for the customization to work uniformly across all these screens.
5. Only non-mandatory fields should be considered for customization.
6. Each of the fields in the form should have an unique name attribute assigned to them.
7. Dynamic List components like “RR Number” are not to be considered for customization.
8. Composite component are not considered for customization. For example if a form item html element is removed from a block, then the corresponding field in a grid representing the element would not be removed simultaneously.

4. Steps for customization

4.1 Change the entry screens page layout to make it a form based layout

The customization is based on the principle that each form item block(div) having a class 'formItem' and containing within it an html element will either be removed or persisted on the page, depending upon the declaration of the value of the name attribute of the html element in the configuration entry file. A set of form item blocks will be inside a block(div) having a class 'formItemBlock'. The input elements of the 'formItem' block is uniquely identified by the value of its 'name' attribute. Hence it is an absolute must that every input element within a 'formItemBlock' must have an unique name attribute assigned to it. Only the html elements with the value of the name attribute declared in the configuration file for the current page will be rendered finally. The image below shows the changes required to be done for changing a table layout to a form based layout.

```

<div class="detailBlock entrySingleCol">
    <table class="tableStyle sixCol">
        <tr>
            <td><label><spring:message code="trd.tradeentryaction.label.tradetype" href="#" /></label></td>
            <td><span class="detailTxtVal">${commandForm.tradeScreenData.tradeType}</span></td>
            .....
            .....
        </tr>
        <tr>
            <td><label><spring:message code="trd.tradeentryaction.label.valuedate" href="#" /></label></td>
            <td><span class="detailTxtVal">${commandForm.tradeScreenData.valueDateStr}</span></td>
            .....
            .....
        </tr>
    </table>
</div>

```

The above image shows the difference in layout between a page designed with a table layout and the same content displayed with a form based layout. The block on the left is the initial table based layout and that on the right is the form based layout. Following are the points of difference that needs to be made as can be seen in the image above :

1. The main wrapper block for the items will not be a table, instead it will be a div. The div should have two classes "formItemBlock" and "dottedBg".
2. Here a row of the table is transformed to a corresponding form based layout. Each column or 'td' element of a row in the table is transformed individually to a formItem block.
3. There are two 'td' blocks for each field, one for the label of the field and the another for displaying the value of the field. The 'td' blocks are to be replaced by a div having the class 'formItem'. Inside the formItem block the label element is to be placed, as it was there in the first 'td' element of the table layout.
4. It should be followed by a span having the class 'detailDisplay'. The name attribute assigned to the span must have an unique value in the context of the current page. The actual text to be displayed is to be placed in this span element.
5. The last statement inside a 'formItem' block should be a div element with the class 'clear' applied to it.
6. In the similar manner as stated in the above points all 'td' elements are to be replaced by a 'formItem' one after another belonging to different rows ('tr') of the initial table layout.
7. Before closing the 'formItemBlock' block, another div having the class 'clear' applied to it must be inserted there.
8. Before closing the div which is wrapping the 'formItemBlock', another div having the class 'linehide' applied to it must be inserted there.

4.2 Change the detail screen page layout to make it a form based layout

This customization is to be done for those details screen which doesn't use the entry confirmation pages as its detail view pages. The

detail view pages are displayed when an user clicks on any hyperlink in a row of the query result grid which would result in the opening of a detail dialog. One example of it is the file tradeDetailView.jspx The layout of the pages in this case is a set of nested tables. Each row of the concerned table block is to be replaced by divs having class 'detailItem'. Initially each table element signifying a block or a logical section has a class 'detailSmlBlk'. This is to be replaced by 'divs' having class 'detailItemBlock'. The current layout of the detail view is such that the sections of the pages are all either rows or columns of the parent table which in itself is a row or column of it's parent table. The table layout needs to be replaced by a form based layout only in the sections where the actual displayable data lies. The image below shows the area of the page which needs to be replaced by a form based layout:

[T20110624085911-2] Trade Details

Counterparty Account		Table with class 'detailSmlBlk' should be replaced by div having class 'detailItemBlock'	
Customer Code		Inventory Account	Security Information
Account No	BRK034007/NCS	Inventory Account No	FRM000001/NCS
Short Name	SMBC NIKKO SECURITIES	Inventory A/C Short Name	FIRM ACCOUNT FOR JP STOCK
Sales Code		Basic Trade Attributes	
Agent Code		Trade Type	EQUITY
Trade Reference		Sub Trade Type	
Reference Number	T20110624085911-2	Buy/Sell Orientation	AB
Linked Trade Ref. No		Principal/Agent Flag	A
Trade Status	CANCEL	Short Sell Flag	
External Reference No		Account Balance Type	
Data Source	SCREEN	Price	
Currency		Input Price	11.00000000
Trade Currency	JPY/ISO3A	Input Price Format	UNIT PRICE
Settlement Currency	JPY/ISO3A	Price	11.00000000
Exchange Rate		Prefigured Principal Flag	N
Issue Currency	JPY/ISO3A	Gross/Net Type	GROSS
Issue Pricing Exchange Rate		Price Conv Type	
Execution Market	JT/NCS	Volume	
		Quantity	11
		Principal Amount	121
		Accrued Interest	
		Net Amount	121
		Principal Amount In Issue Currency	121
		Net Amount In Trading Currency	121

So the concerned area for layout change is every nested table having the class 'detailSmlBlk' and the contents within it. Here also like in the case of other entry based screens, the element which actually holds the displayable value for a field should have a 'name' attribute with an unique value. This 'name' value will be used to uniquely identify a field and depending on the declaration of the name in the configuration the corresponding field would either be removed or persisted in the page. The image below shows the changes required to be done for changing a table layout to a form based layout.

```


|                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <form:label path=""><spring:message code="trade.label.customer.code" htmlEscape="false"/> | <c:choose> <c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}> <div class = "hyperlink-relative-div"><span class="detailTxtVal detail-view-hyperlink old" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;">\${oldScreenData.customerCode}</span></div>  <div class = "hyperlink-relative-div"><span class="detailTxtVal detail-view-hyperlink new" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;">\${newScreenData.customerCode}</span></div><c:when> <c:otherwise> <div class = "hyperlink-relative-div"><span class="detailTxtVal"> <a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink">\${newScreenData.customerCode}</a></span></div> </c:otherwise> </c:choose> |
|-------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|


```

```

<tr>
    <td><form:label path=""><spring:message code="trade.label.sales.code" htmlEscape="false"/></form:label>
    <td><valueDifference:valueDifferenceId oldValue="${oldScreenData.salesCode}" newValue="${newScreenData.salesCode}" diffEnableFlag="${commandForm.diffEnableFlag}" /></td>
</tr>

<br>

<tr class="noBdr">
    <td><form:label path=""><spring:message code="trade.label.agent.code" htmlEscape="false"/></form:label>
    <td><valueDifference:valueDifferenceId oldValue="${oldScreenData.agentCode}" newValue="${newScreenData.agentCode}" diffEnableFlag="${commandForm.diffEnableFlag}" /></td>
</tr>

</table>

```

The above image shows the difference in layout between a detail view page designed with a table layout and the same content displayed with a form based layout. The block on the left is the initial table based layout and that on the right is the form based layout. Following are the points of difference that needs to be made as can be seen in the image above :

1. The changes needs to be done in the table block which is closest to the actual row of data, for example here in the table with the class 'detailSmlBlk'. The table element needs to be replaced with a div having class 'detailItemBlock'. There are other table blocks wrapping this table block, but since it is closest to the actual row of data, hence the customization is commenced from here.
2. Each row(tr) of the table needs to be replaced with a div having class 'detailItem'.
3. The two enclosing <td> elements inside the tr element needs to replaced directly by the the html block wrapped inside the td elements.
4. Instead of using 'form:label' one may also choose to use the native html 'label' tag.
5. The div having the class 'hyperlink-relative-div' which encloses a span element should be removed.
6. As discussed previously every element should have an unique 'name' attribute, so here also the span should have a name attribute with an unique value
7. The sections where the custom tag 'valueDifference:valueDifferenceId' is used, that must be replaced by the custom tag 'valueDifference:valueDifferenceDetailsId'.
8. For setting the unique name attribute the custom attribute 'fieldName' must be used.
9. The last row(tr) of the current table block has a class 'noBdr' applied to it, no such classes needs to be applied to the last detailItem div block.

Beside the above mentioned block, one also needs to modify those table blocks which has six column style appearance, i.e the total block is a single table with each row having three columns of data. The image below shows the changes that needs to be done :

<pre> <table class="tableStyle sixCol more printTableStyle" style="display:none;"> <tr> <td><form:label path=""><spring:message code="trade.label.order.refno" htmlEscape="false"/></form:label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.orderReferenceNo}" newValue="\${newScreenData.orderReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> </tr> <tr> <td><form:label path=""><spring:message code="trade.label.etc.refno" htmlEscape="false"/></form:label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.etcReferenceNo}" newValue="\${newScreenData.etcReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> <td><form:label path=""><spring:message code="trade.label.sender.refno" htmlEscape="false"/></form:label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.senderReferenceNo}" newValue="\${newScreenData.senderReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> <td>.....</td> <td>.....</td> </tr> </table> </pre>	<pre> <tr> <td><label name="orderrefno"><spring:message code="trade.label.order.refno" htmlEscape="false"/></label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.orderReferenceNo}" newValue="\${newScreenData.orderReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> <td><label name="etcrefno"><spring:message code="trade.label.etc.refno" htmlEscape="false"/></label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.etcReferenceNo}" newValue="\${newScreenData.etcReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> <td><label name="senderrefno"><spring:message code="trade.label.sender.refno" htmlEscape="false"/></label></td> <td><valueDifference:valueDifferenceId oldValue="\${oldScreenData.senderReferenceNo}" newValue="\${newScreenData.senderReferenceNo}" diffEnableFlag="*(commandForm.diffEnableFlag)" /></td> </tr> <tr> <td>.....</td> <td>.....</td> </tr> </table> </pre>
---	---

1. A class 'tableBox' needs to applied to the table to signify that the table must be considered for customization.
2. Instead of using 'form:label', the simple html <label> element must be used. In the 'td' of the label element a name attribute must be specified with an unique value which will be used to identify the column. If the given name attribute is not present in the configuration for customization, then the corresponding label and the following td holding the value against it will not be displayed in the table.
3. The elements are placed in the table in the order in which the name of the elements are stated in the configuration. If the colspan of a particular element needs to be more than 1 such as 3 or 5, then the element is either placed in the current row or the next row as per the available space in the row and the required colspan of the element.

One thing that needs to be considered in the case of detail dialogs like this(detail dialogs in which a different view to that of the entry detail/confirmation pages are used) is that in these screens the 'screenId' and the 'versionNo' fields are not populated by the same process as is done in the case of entry, amend, cancel or copy screens. The 'screenId' and 'versionNo' of a screen are read from the hidden fields within the displayed page with the name 'screenId' and 'versionNo' respectively. The value assigned to those fields are interpreted as the corresponding values of screenId and versionNo. Hence if in the detail page that you are modifying has no such fields or either any one of them are missing, then they must be added to the page, and the value should be populated from the server side.

Markup for adding a hidden 'versionNo' field

```
<input type="hidden" name="versionNo" value="${detailsVersionNo}" />
```

The value can be populated by any means as the application may want, but the easiest way is to put the value for screenId or versionNo in the modelMap itself and to retrieve it directly in the jspx page by using EL.

Adding the value for the field versionNo in the model Map

```
xenosModelMap.getModelMap().addAttribute("detailsVersionNo", "1.2");
```

4.3 Specify the entry screen configuration in a json file :

A sample json configuration for Trade related screens is depicted below :

Sample-Trade-Entry-Screen-Config.json

```

/* The number specified within the comments relates to the point described below */

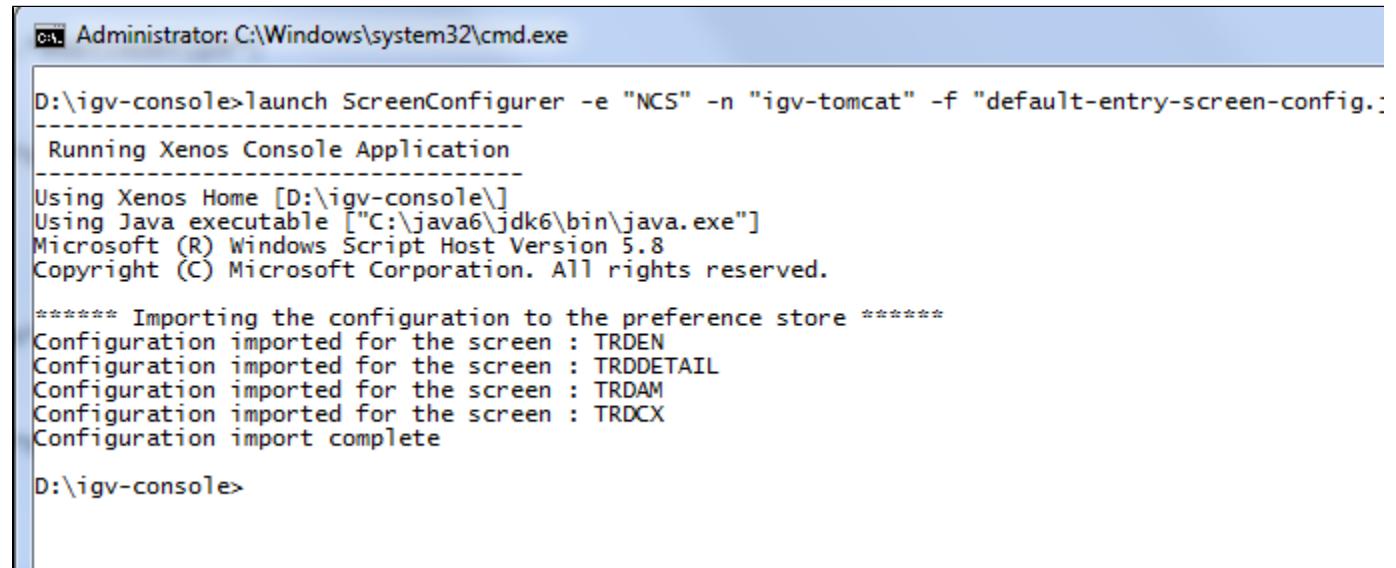
/* 1 */
{
  "TRDEN": {           /* 2,3 */
    "1.2": {           /* 4 */
      "0": {           /* 4 */
        "ScreenConfig": ["tradeType", "subTradeType"] /* 4 */
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    },
    "2": {
      "0": {
        "ScreenConfig": ["accountType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"] /*6,7,8*/
      }
    }
  },
  "TRDDETAIL": {
    "1.2": {
      "NA": {           /* 5 */
        "ScreenConfig": ["tradeType", "subTradeType"]
      }
    }
  },
  "TRDAM": {
    "1.2": {
      "0": {
        "ScreenConfig": ["tradeType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    },
    "2": {
      "0": {
        "ScreenConfig": ["accountType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    }
  },
  "TRDCX": {
    "1.2": {
      "0": {
        "ScreenConfig": ["tradeType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    }
  }
}

```

The following are the set of observations that must be noted in the above mentioned JSON config file :

1. The whole configuration should be a valid JSON object.
2. Inside the JSON object the configuration related to each screens should be described. It should be noted here that for each screens belonging to a particular module distinct set of configurations needs to be described for each of it's concerned actions, i.e for amend, cancel, copy, screen configuration needs to be described separately even though they may be same as that to the entry screen and uses the same jspx pages. The purpose of this design is to be as verbose as possible about configurations of screens with different screenIds and to allow custom screen configuration of the same jspx pages in the context of different actions if any such requirement arises.
3. The top level entry in the JSON configuration for a screen is specifying it's screenId. For example the configuration for a screenId 'TRDEN' should be an object with the key being the screenId, and the value being another object containing the configuration details.
4. The nesting of the JSON configuration for a screen should start with the screenId followed by the versionNo, pageIndex and a key "ScreenConfig". The versionNo is the version no of the page, pageIndex is basically the tab order of the page in the wizard for which the configuration is being described.
5. If "NA" is given in the pageIndex section, then it signifies that pageIndex is not applicable for the current screen, i.e, the screen doesn't have multiple pages. Hence the preference would be saved directly under the versionNo key in the preference store. This is applicable in cases of single page details screen.
6. The actual configuration for the screen is described inside an array which serves as the value for the key 'ScreenConfig'. The array should be populated with the value of the 'name' attribute of the fields which you want displayed in the screen. The fields from the jspx whose names are absent in this configuration will not be displayed in that page for that combination of screenId, versionNo and pageIndex.
7. It should be noted here that keeping in sync with the behavior of query screen configuration, here also the order in which the names are stated inside the array is significant. The fields would be displayed in the page in the same order as the names are stated in the configuration array.
8. Another thing that must be noted here is that though the order of fields displayed in the page is according to the order in which the names are stated in the configuration file, the order is only preserved within the same 'formItemBlock' or 'detailItemBlock'. For example, lets say there are five 'formItems' with names "A", "B", "C", "D" and "E". The 'formItems' "A", "B" and "C" belongs to the 'formItemBlock' "Block1" and 'formItem' "D" and "E" belongs to the 'formItemBlock' "Block2". Now let us suppose that there is a configuration array like this ["B", "C", "E", "A", "D"], then though "E" is stated before "A", "E" will not be displayed before "A" in the page. This is because "E" belongs to a different 'formItemBlock' than "A", "B" and "C". In the 'formItemBlock' "Block 1" the order of items would be "B", "C" followed by "A", and in the 'formItemBlock' "Block2" the order of items would be "E" followed by "D". The same behavior is true for 'detailItems' and 'detailItemBlock'.
9. The configuration does not take into consideration the 'mode' (such as 'EDIT', 'VIEW', 'CONFIRMATION' etc.) in which the page is. So in cases where there are additional fields in a mode for the same combination of screenId, versionNo and pageIndex in respect to that of in some other mode, the configuration should take care of that extra fields and must state them in the configuration array if they want them to be displayed. The fields are searched with the names in the given array and are then persisted in the page accordingly, so if there is any name stated in that configuration for which no such field exists in the jspx for the current mode it would cause no unanticipated behavior and the name would simply be ignored. But in a different mode for the same combination of screenId, versionNo and pageIndex if a field exists with that name then that field will be acknowledged and displayed accordingly.

4.3 Run a tool to import the required configuration to the preference store of the server :



```
D:\igv-console>launch ScreenConfigurer -e "NCS" -n "igv-tomcat" -f "default-entry-screen-config.:
-----
Running Xenos Console Application
-----
Using Xenos Home [D:\igv-console\]
Using Java executable ["C:\java6\jdk6\bin\java.exe"]
Microsoft (R) Windows Script Host Version 5.8
Copyright (C) Microsoft Corporation. All rights reserved.

***** Importing the configuration to the preference store *****
Configuration imported for the screen : TRDEN
Configuration imported for the screen : TRDDETAIL
Configuration imported for the screen : TRDAM
Configuration imported for the screen : TRDCX
Configuration import complete

D:\igv-console>
```

A CLI tool is provided to import the entry screen configuration to the preference store of the server in which the application is to be deployed. The tool should be run in the following way :

Tool for importing entry based screen preferences

```
launch ScreenConfigurer -e "NCS" -n "igv-tomcat" -f "default-entry-screen-config.json"
```

The tool is run with the command 'launch ScreenConfigurer', the enterprise Id for which the preference is to be imported has to be provided with the "-e" parameter. The name of the app(context-root) is to be supplied with the argument "-n" and the filepath is to be supplied with the argument "-f". The file path is resolved according to the specifications as described in the [PathMatchingResourcePatternResolver](#) of the spring core. Please refer to the link for more details. In the above mentioned case as only the name of the file is specified, the scope of the search for the file is restricted to the classpath of the application.

5. Screen configuration JSON creation helper:

A Chrome extension is also being provided to help in the creation of the configuration JSON for any entry based screen and details page. The following steps must be followed for using the extension:

1. Install [this Chrome extension](#) locally to your machine. The Chrome version should be above 23.
2. Go to the first page of the entry screen or details screen. Click on the chrome extension to start the creation of the JSON configuration file.
3. Navigate to every next screen one after another until you reach to the final page of the wizard.
4. After reaching to the end, click the extension icon again to indicate the end of the recording. A new tab will open with the JSON configuration as recorded from each of the screens which were navigated.
5. In the case of a detail screen being present on the screen at the time of the recording, the configuration relating to it will also be recorded simultaneously.
6. For a single page detail screen, as there is no need of navigation hence the chrome extension button can be clicked twice simultaneously to indicate the start and the stop of the recording.
7. Whenever the recording will be stopped, the configuration recorded till that moment will be displayed in the new tab.
8. Only a single screen(for example only TRDEN/TRDCP/TRDAM, but not all) must be considered for recording at a time. The configuration related to a screen must be recorded in a single flow without escaping from it to any other screen while the configuration is being recorded.
9. The JSON configuration created consists of the names of all the relevant fields in a screen which can be considered for customization. The user may remove fields from the list in the JSON as per his requirement and feed the final JSON to the command line tool for saving the configuration.

The images below shows the flow while recording the configuration of trade copy screen(TRDCP) :

localhost:8081/gv-tomcat/#

Reference Mst ▾ Trade ▾ Settlement ▾ Position & Balances ▾ Derivatives ▾ Authorization ▾ More ▾ Search Menu[G]

NCS 20140527

TRDCP:Trade Copy [20140527]

1 Trade General → **2 Trade Details**

Click on the screen configuration creator chrome extension

Trade Type	EQUITY	Sub Trade Type	
Trade Date	20110624	Value Date	20110624
Buy/Sell Orientation	Client Buy	Account No	BRK034007/NCS
Inventory Account No	FRM000001/NCS	Principal Type	Agency
Delivery Method	BOOK ENTRY	Gross/Net Type	GROSS
Input Price Format	UNIT PRICE	Execution Market	JT/NCS
Clearing Party		Settlement Currency	JPY//ISO3A
As Customer Flag		Exchange Rate	
NSCC Report Eligible		Account Balance Type	0: DVP/RVP
Compressed Trade Flag		DVP Eligible	
		Match Eligible	

The user is notified that the configuration creation has started

Screen Configurator
Recording has started

© 2013 Nomura Research Institute, Ltd. Ver 2.3 11.0.0

localhost:8081/igv-tomcat/#

Reference Mst ▾ Trade ▾ Settlement ▾ Position & Balances ▾ Derivatives ▾ Authorization ▾ More ▾ Search Menu ▾   NCS 20140527

TRDCP:Trade Copy [20140527]

1 Trade General > **2 Trade Details**

Trade Type	EQUITY	Sub Trade Type	Trade Date	20110624	Value Date	20110624	
Buy/Sell Orientation	AB	Short Selling Flag	Short Sell Exempt Flag		Account No	BRK034007/NCS	
Short Name	SMBC NIKKO SECURITIES	Inventory Account No	FRM000001/NCS	Inventory A/C Short Name	FIRM ACCOUNT FOR JP STOCK	Principal Type	Agency
Instrument Type	RT	Security Code	1234	Security Name	ABC	Alternate Security Code	JP3257200023/ISIN
Delivery Method	BOOK ENTRY	Input Price	11.00000000	Quantity	11	Gross/Net Type	GROSS
Sales Code		Input Price Format	UNIT PRICE	Price Conv Type		Trade Currency	JPY/ISO3A
Execution Market	JT/NCS	Execution Method	MARKET	Clearing Party		Settlement Currency	JPY/ISO3A
Exchange Rate		Calculation Type	Divide	Issue Currency	JPY/ISO3A	Issue Pricing Exchange Rate	
Issue Pricing Calculation Type		As Customer Flag		Step In/Out Flag		Step Out Account	
Account Balance Type	0: DVP/RVP	NSC Report Eligible		Compressed Trade Flag		DVP Eligible	
Match Eligible		Per Value					
Price	11 00000000	Principal Amount	<input type="text" value="121"/>	Prefigured Principal Flag	No	Principal Amount In Issue Curre...	121
Net Amount	121	Net Amount in Trading Currency	121				

Tax Fee

Tax Fee ID(*)	<input type="text"/>	Commission ID(*)	<input type="text"/>		
Rate	<input type="text"/>	Rate Type	<input type="text"/>		
Amount	<input type="text"/>	Add 			
Actions	ID	Rate	Rate Type	Amount	

RR

RR Role	RR Number	Commission Amount
RR	<input type="text"/>	
Trader	<input type="text"/>	
Executing RR	<input type="text"/>	
Split RR1	<input type="text"/>	
Split RR2	<input type="text"/>	

Buttons: Back, Previous, Next, Reset, Submit

© 2013 Nomura Research Institute, Ltd. Ver.3.3 11.0.0

localhost:8081/lgv-tomcat/#

Reference Mst ▾ Trade ▾ Settlement ▾ Position & Balances ▾ Derivatives ▾ Authorization ▾ More ▾ Search Menu Q

NCS 20140527

TRDCP:User Confirmation - Trade Copy

Trade General		Trade Details			
Trade Type	EQUITY	Sub Trade Type		Trade Date	20110624
Buy/Sell Orientation	AB	Account No	BRX034007/NCS	Short Name	SMBCNIKKO SECURITIES
Inventory A/C Short Name	FIRM ACCOUNT FOR JP STOCK	Principal Type	Agency	Instrument Type	RT
Security Name	ABC	Alternate Security Code	JP3257200023/ISIN	Quantity	11
Input Price	11.00000000	Gross/Net Type	GROSS	Sales Code	Input Price Format
Price Conv Type		Trade Currency	JPY/ISO3A	Execution Market	JT/NCS
Clearing Party		Settlement Currency	JPY/ISO3A	Exchange Rate	Execution Method
Issue Currency	JPY/ISO3A	Issue Pricing Exchange Rate		Issue Pricing Calculation Type	Market
Step In/Out Flag		Step Out Account		Account Balance Type	As Customer Flag
Compressed Trade Flag		DVP Eligible		Match Eligible	NSCC Report Eligible
					Per Value
<input type="button" value="Back"/> <input type="button" value="Confirm"/>					

© 2013 Nomura Research Institute, Ltd. Ver.3.3 11.0.0

localhost:8081/gv-tomcat/#

Reference Mst ▾ Trade ▾ Settlement ▾ Position & Balances ▾ Derivatives ▾ Authorization ▾ More ▾ Search Menu ▾

istar ▾ NCS 20140527

TRDCP:User Confirmation - Trade Copy

Trade General		Trade Details			
Price	11.00000000	Principal Amount	121	Prefigured Principal Flag	N
Net Amount	121	Net Amount in Trading Currency	121	Principal Amount In Issue	121
				Currency	

Tax Fee				RR				
ID	Rate	Rate Type	Amount	RR Name	RR Role	RR Number	Commission Category	Commission Amount
No Records Found				No Records Found				

Detail Information

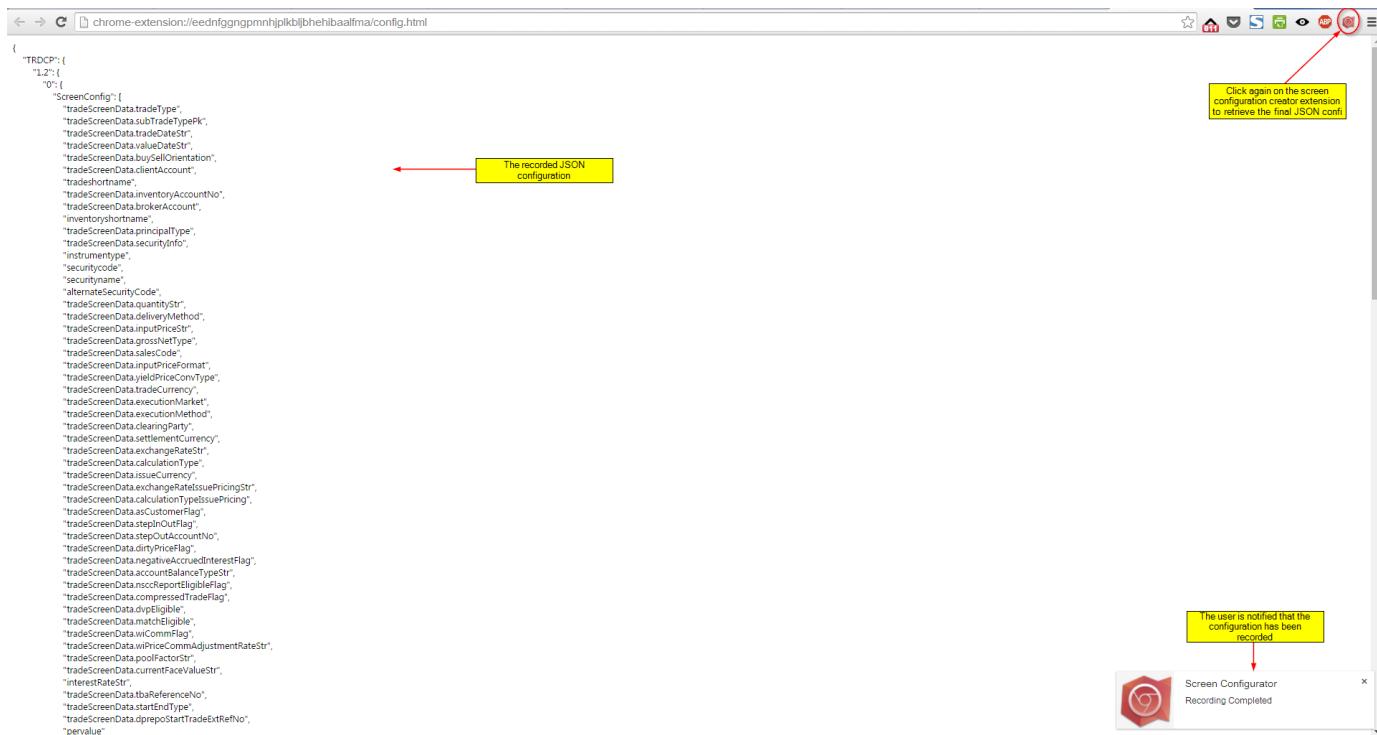
Remarks	Remarks for Reports	Order Reference No	Broker Reference No
Etc Reference No	External Reference No	MSBCC Flag	Trace Eligible
Form T	Soft Dollar Flag	Stipulation	Sender Reference No
Verbiage Name	STL Location For Security	STL Location For Cash	MSBCC Firm Clearing A/C

SSI

Cash Security Flag	Acronym	Instrument Type	Settlement Currency	Settlement Type	Bank Name	Contra Id	DTC Participant Number	CP External Settlement A/C	Priority

Back Confirm

© 2013 Nomura Research Institute, Ltd. Ver.3.3.11.0.0



6. Sample implementation:

The new code blocks for supporting customization in trade module screens are shown below.

6.1 Entry based wizard pages:

tradeDetailsEntry.jspx
Expand

```

<div class="detailBlock entrySingleCol">
  <div class="formItemBlock dottedBg">
    <div class="formItem">
      <label><spring:message code="trd.tradeentryaction.label.tradetype" htmlEscape="false"/></label>

```

source

```

<span class="detailDisplay" name="tradetype">${commandForm.tradeScreenData.tradeType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.subtradetype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="subtradetype">${commandForm.tradeScreenData.subTradeTypeDisp}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.tradedate" htmlEscape="false"/></label>
<span class="detailDisplay" name="tradedate">${commandForm.tradeScreenData.tradeDateStr}
${commandForm.tradeScreenData.tradeTime}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.valuedate" htmlEscape="false"/></label>
<span class="detailDisplay" name="valuedate">${commandForm.tradeScreenData.valueDateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.buysellorientation" htmlEscape="false"/></label>
<span class="detailDisplay"
name="buysellorientation">${commandForm.tradeScreenData.buySellOrientation}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.shortsellingflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="shortsellingflag">${commandForm.tradeScreenData.shortSellingFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.shortsellexemptflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="shortsellexemptflag">${commandForm.tradeScreenData.shortSellExemptFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.accountno" htmlEscape="false"/></label>
<span class="detailDisplay" name="accountno">${commandForm.tradeScreenData.accountNo}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.short.name" htmlEscape="false"/></label>
<span class="detailDisplay" name="tradeshortname">${commandForm.tradeScreenData.shortName}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inventoryaccountno" htmlEscape="false"/></label>
<span class="detailDisplay"
name="inventoryaccountno">${commandForm.tradeScreenData.inventoryAccountNo}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.inventory.shortname" htmlEscape="false"/></label>
<span class="detailDisplay"
name="inventoryshortname">${commandForm.tradeScreenData.invAccShortName}</span>
<div class="clear"></div>

```

```
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.principaltytype" htmlEscape="false"/></label>
<span class="detailDisplay" name="principaltytype">${commandForm.tradeScreenData.principalType}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.instrument.type" htmlEscape="false"/></label>
<span class="detailDisplay"
name="instrumenttype">${commandForm.tradeScreenData.instrumentType}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.security.code" htmlEscape="false"/></label>
<span class="detailDisplay" name="securityId">${commandForm.tradeScreenData.securityId}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.security.name" htmlEscape="false"/></label>
<span class="detailDisplay" name="securityname">${commandForm.tradeScreenData.officialName}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.alternative.security.code" htmlEscape="false"/></label>
<span class="detailDisplay"
name="alternateSecurityCode">${commandForm.tradeScreenData.alternateSecurityCode}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.deliverymethod" htmlEscape="false"/></label>
<span class="detailDisplay"
name="deliveryMethod">${commandForm.tradeScreenData.deliveryMethod}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inputprice" htmlEscape="false"/></label>
<span class="detailDisplay" name="inputprice">${commandForm.tradeScreenData.inputPriceStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.quantity" htmlEscape="false"/></label>
<span class="detailDisplay" name="quantity">${commandForm.tradeScreenData.quantityStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.grossnettype" htmlEscape="false"/></label>
<span class="detailDisplay" name="grossnettype">${commandForm.tradeScreenData.grossNetType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.salescode" htmlEscape="false"/></label>
<span class="detailDisplay" name="salescode">${commandForm.tradeScreenData.salesCode}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inputpriceformat" htmlEscape="false"/></label>
<span class="detailDisplay"
name="inputpriceformat">${commandForm.tradeScreenData.inputPriceFormat}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.yieldpriceconvtype" htmlEscape="false"/></label>
```

```

<span class="detailDisplay"
name="yieldpriceconvtype">${commandForm.tradeScreenData.yieldPriceConvType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.tradecurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradecurrency">${commandForm.tradeScreenData.tradeCurrency}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.executionmarket" htmlEscape="false"/></label>
<span class="detailDisplay"
name="executionmarket">${commandForm.tradeScreenData.executionMarket}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.executionmethod" htmlEscape="false"/></label>
<span class="detailDisplay"
name="executionmethod">${commandForm.tradeScreenData.executionMethod}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.clearingParty" htmlEscape="false"/></label>
<span class="detailDisplay" name="clearingParty">${commandForm.tradeScreenData.clearingParty}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.settlementcurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="settlementcurrency">${commandForm.tradeScreenData.settlementCurrency}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.exchangerate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="exchangerate">${commandForm.tradeScreenData.exchangeRateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.calculationtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="calculationtype">${commandForm.tradeScreenData.calculationTypeDisp}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issuecurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="issuemoney">${commandForm.tradeScreenData.issueCurrency}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issueexchangerate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="issueexchangerate">${commandForm.tradeScreenData.exchangeRateIssuePricingStr}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issuecalculationtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="issuecalculationtype">${commandForm.tradeScreenData.calculationTypeIssuePricing}</span>
<div class="clear"></div>
</div>

```

```

<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.ascustomerflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="ascustomerflag">${commandForm.tradeScreenData.asCustomerFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.stepinoutflag" htmlEscape="false"/></label>
<span class="detailDisplay" name="stepinoutflag">${commandForm.tradeScreenData.stepInOutFlag}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.stepoutaccount" htmlEscape="false"/></label>
<span class="detailDisplay"
name="stepoutaccount">${commandForm.tradeScreenData.stepOutAccountNo}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.excouponflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="exCouponFlag">${commandForm.tradeScreenData.exCouponFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.dirtypriceflag" htmlEscape="false"/></label>
<span class="detailDisplay" name="dirtyPriceFlag">${commandForm.tradeScreenData.dirtyPriceFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.negativeaccruedinterestflag"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="negativeAccruedInterestFlag">${commandForm.tradeScreenData.negativeAccruedInterestFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Mbstbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.whenissueindicator" htmlEscape="false"/></label>
<span class="detailDisplay"
name="whenissueindicator">${commandForm.tradeScreenData.wiCommFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Mbstbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.wipriceconrate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="wipriceconrate">${commandForm.tradeScreenData.wiPriceCommAdjustmentRateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Tbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.tbareferenceNo" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tbareferenceNo">${commandForm.tradeScreenData.tbaReferenceNo}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond4Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.poolfactor" htmlEscape="false"/></label>
<span class="detailDisplay" name="poolfactor">${commandForm.tradeScreenData.poolFactorStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond4Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.currentfacevalue" htmlEscape="false"/></label>

```

```
<span class="detailDisplay"
name="currentfacevalue">${commandForm.tradeScreenData.currentFaceValueStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Dprepoltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.startendtype" htmlEscape="false"/></label>
<span class="detailDisplay" name="startendtype">${commandForm.tradeScreenData.startEndType}</span>
<div class="clear"></div>
</div>
<div class="formItem Dprepoltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.dpreposttradeextref" htmlEscape="false"/></label>
<span class="detailDisplay"
name="dpreposttradeextref">${commandForm.tradeScreenData.drepoStartTradeExtRefNo}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.accbalancetype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="accbalancetype">${commandForm.tradeScreenData.accountBalanceTypeDisplay}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.nsccrepeligible" htmlEscape="false"/></label>
<span class="detailDisplay"
name="nsccrepeligible">${commandForm.tradeScreenData.nsccReportEligibleFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.compressedtradeflag"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="compressedtradeflag">${commandForm.tradeScreenData.compressedTradeFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trd.tradeentryaction.label.dvpeligible" htmlEscape="false"/></label>
<span class="detailDisplay" name="dvpeligible">${commandForm.tradeScreenData.dvpEligible}</span>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trd.tradeentryaction.label.matcheligible" htmlEscape="false"/></label>
<span class="detailDisplay" name="matcheligible">${commandForm.tradeScreenData.matchEligible}</span>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trade.label.per.value" htmlEscape="false"/></label>
<span class="detailDisplay" name="pervalue">${commandForm.tradeScreenData.perValue}</span>
<div class="clear"></div>
</div>
```

```
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<div class="lineHide"><spring:message text="" htmlEscape="false"/></div>
```

tradeGeneralDetail.jspx

› Expand

source

```
<div class="detailBlock entrySingleCol">
<div class="formItemBlock dottedBg">
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.tradetype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.tradeType">${commandForm.tradeScreenData.tradeType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.subtradetype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.subTradeTypePk">${commandForm.tradeScreenData.subTradeTypeDisp}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.tradedate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.tradeDateStr">${commandForm.tradeScreenData.tradeDateStr}
${commandForm.tradeScreenData.tradeTime}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.valuedate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.valueDateStr">${commandForm.tradeScreenData.valueDateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.buysellorientation" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.buySellOrientation">${commandForm.tradeScreenData.buySellOrientation}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.shortsellingflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.shortSellingFlag">${commandForm.tradeScreenData.shortSellingFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.shortsellexemptflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.shortSellExemptFlag">${commandForm.tradeScreenData.shortSellExemptFlag}</span
>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.accountno" htmlEscape="false"/></label>
<div class = "hyperlink-relative-div"><span class="detailDisplay" name="tradeScreenData.clientAccount"><a href="/secure/ref/account/details/${commandForm.tradeScreenData.accountPk}?popup=true"
```

```

view="accountDetailView" dialogTitle="[${commandForm.tradeScreenData.accountNo}] ${accountDetailTitle}"
style="outline:0;"
class="detail-view-hyperlink">>${commandForm.tradeScreenData.accountNo}</a></span></div>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.short.name" htmlEscape="false"/></label>
<span class="detailDisplay" name="tradeshortname">${commandForm.tradeScreenData.shortName}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inventoryaccountno" htmlEscape="false"/></label>
<div class = "hyperlink-relative-div"><span class="detailDisplay"
name="tradeScreenData.inventoryAccountNo"><a
href="/secure/ref/account/details/${commandForm.tradeScreenData.inventoryAccountPk}?popup=true"
view="accountDetailView" dialogTitle="[${commandForm.tradeScreenData.inventoryAccountNo}]
${accountDetailTitle}" style="outline:0;">${commandForm.tradeScreenData.inventoryAccountNo}</a></span></div>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.inventory.shortname" htmlEscape="false"/></label>
<span class="detailDisplay"
name="inventoryshortname">${commandForm.tradeScreenData.invAccShortName}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.principaltype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.principalType">${commandForm.tradeScreenData.principalType}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.instrument.type" htmlEscape="false"/></label>
<span class="detailDisplay"
name="instrumenttype">${commandForm.tradeScreenData.instrumentType}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.security.code" htmlEscape="false"/></label>
<div class = "hyperlink-relative-div"><span class="detailDisplay" name="securitycode"><a
href="/secure/ref/instrument/details/${commandForm.tradeScreenData.instrumentPk}?popup=true"
view="instrumentDetailView" dialogTitle="[${commandForm.tradeScreenData.securityId}]
${instrumentDetailTitle}" style="outline:0;">${commandForm.tradeScreenData.securityId}</a></span></div>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trade.label.security.name" htmlEscape="false"/></label>
<span class="detailDisplay" name="securityname">${commandForm.tradeScreenData.officialName}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trade.label.alternative.security.code" htmlEscape="false"/></label>
<span class="detailDisplay"
name="alternateSecurityCode">${commandForm.tradeScreenData.alternateSecurityCode}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.delivermethod" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.deliveryMethod">${commandForm.tradeScreenData.deliveryMethod}</span>

```

```
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inputprice" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.inputPriceStr">${commandForm.tradeScreenData.inputPriceStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.quantity" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.quantityStr">${commandForm.tradeScreenData.quantityStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.grossnettype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.grossNetType">${commandForm.tradeScreenData.grossNetType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.salescode" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.salesCode">${commandForm.tradeScreenData.salesCode}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inputpriceformat" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.inputPriceFormat">${commandForm.tradeScreenData.inputPriceFormat}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.yieldpriceconvtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.yieldPriceConvType">${commandForm.tradeScreenData.yieldPriceConvType}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.tradecurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.tradeCurrency">${commandForm.tradeScreenData.tradeCurrency}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.executionmarket" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.executionMarket">${commandForm.tradeScreenData.executionMarket}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.executionmethod" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.executionMethod">${commandForm.tradeScreenData.executionMethod}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.clearingParty" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.clearingParty">${commandForm.tradeScreenData.clearingParty}</span>
<div class="clear"></div>
</div>
```

```

<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.settlementcurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.settlementCurrency">${commandForm.tradeScreenData.settlementCurrency}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.exchangerate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.exchangeRateStr">${commandForm.tradeScreenData.exchangeRateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.calculationtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.calculationType">${commandForm.tradeScreenData.calculationTypeDisp}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issuecurrency" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.issueCurrency">${commandForm.tradeScreenData.issueCurrency}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issueexchangerate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.exchangeRateIssuePricingStr">${commandForm.tradeScreenData.exchangeRateIssue
PricingStr}</span>
<div class="clear"></div>
</div>
<div class="formItem IssueItem">
<label><spring:message code="trd.tradeentryaction.label.issuecalculationtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.calculationTypeIssuePricing">${commandForm.tradeScreenData.calculationTypeIssue
ePricing}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.ascustomerflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.asCustomerFlag">${commandForm.tradeScreenData.asCustomerFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.stepinoutflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.stepInOutFlag">${commandForm.tradeScreenData.stepInOutFlag}</span>
<div class="clear"></div>
</div><div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.stepoutaccount" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.stepOutAccountNo">${commandForm.tradeScreenData.stepOutAccountNo}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.excouponflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.exCouponFlag">${commandForm.tradeScreenData.exCouponFlag}</span>
<div class="clear"></div>

```

```

</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.dirtypriceflag" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.dirtyPriceFlag">${commandForm.tradeScreenData.dirtyPriceFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond1Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.negativeaccruedinterestflag"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.negativeAccruedInterestFlag">${commandForm.tradeScreenData.negativeAccruedInt
erestFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Mbstbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.whenissueindicator" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.wiCommFlag">${commandForm.tradeScreenData.wiCommFlag}</span>
<div class="clear"></div>
</div>
<div class="formItem Mbstbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.wipriceconvrate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.wiPriceCommAdjustmentRateStr">${commandForm.tradeScreenData.wiPriceCommA
djustmentRateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Tbaltem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.tbareferenceNo" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.tbaReferenceNo">${commandForm.tradeScreenData.tbaReferenceNo}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond4Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.poolfactor" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.poolFactorStr">${commandForm.tradeScreenData.poolFactorStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Bond4Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.currentfacevalue" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.currentFaceValueStr">${commandForm.tradeScreenData.currentFaceValueStr}</span
>
<div class="clear"></div>
</div>
<div class="formItem Bond4Item" style="display:none;">
<label><spring:message code="trade.label.interest.rate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="interestRateStr">${commandForm.tradeScreenData.interestRateStr}</span>
<div class="clear"></div>
</div>
<div class="formItem Dprepoltitem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.startendtype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.startEndType">${commandForm.tradeScreenData.startEndType}</span>
<div class="clear"></div>
</div>

```

```
<div class="formItem DprepoItem" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.dpreposttradeextref" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.dprepoStartTradeExtRefNo">${commandForm.tradeScreenData.dprepoStartTradeExt
RefNo}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.accbalancetype" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.accountBalanceTypeStr">${commandForm.tradeScreenData.accountBalanceTypeDis
play}</span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.nsccrepeligible" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.nsccReportEligibleFlag">${commandForm.tradeScreenData.nsccReportEligibleFlag}</
span>
<div class="clear"></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.compressedtradeflag"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.compressedTradeFlag">${commandForm.tradeScreenData.compressedTradeFlag}</s
pan>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trd.tradeentryaction.label.dvpeligible" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.dvpEligible">${commandForm.tradeScreenData.dvpEligible}</span>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trd.tradeentryaction.label.matcheligible" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.matchEligible">${commandForm.tradeScreenData.matchEligible}</span>
<div class="clear"></div>
</div>
<div class="formItem noBdr">
<label><spring:message code="trade.label.per.value" htmlEscape="false"/></label>
<span class="detailDisplay" name="pervalue">${commandForm.tradeScreenData.perValue}</span>
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
```

```
</div>
<div class="lineHide"><spring:message text="" htmlEscape="false"/></div>
</div>
```

tradeDetailsParticular.jspx

› [Expand](#)

[source](#)

```
<div class="detailBlock entrySingleCol">
<div class="formItemBlock dottedBg">
<!-- Accrued Interest Amount -->
<div class="formItem Bond2Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.accruedinterestamount"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.accruedInterestAmountStr">${commandForm.tradeScreenData.accruedInterestAmou
ntStr}</span>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Accrued Days -->
<div class="formItem Bond2Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.accrueddays" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.accruedDaysStr">${commandForm.tradeScreenData.accruedDaysStr}</span>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Accrued Start Date -->
<div class="formItem Bond2Item" style="display:none;">
<label><spring:message code="trd.tradeentryaction.label.accruedstartdate" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.accruedStartDateStr">${commandForm.tradeScreenData.accruedStartDateStr}</span
>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Price -->
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.price" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.priceStr">${commandForm.tradeScreenData.priceStr}</span>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Principal Amount -->
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.principalamount" htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.principalAmountStr">${commandForm.tradeScreenData.principalAmountStr}</span>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.principalamountissuocurrency"
htmlEscape="false"/></label>
<span class="detailDisplay"
name="tradeScreenData.principalAmountInIssueStr">${commandForm.tradeScreenData.principalAmountInIss
ueStr}</span>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Prefigured Principal Flag -->
```

```
<div class="formItem">
    <label><spring:message code="trd.tradeentryaction.label.prefiguredprincipalflag"
htmlEscape="false"/></label>
    <span class="detailDisplay"
name="tradeScreenData.prefiguredPrincipalFlag">${commandForm.tradeScreenData.prefiguredPrincipalFlag}<
/span>
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Net Amount -->
<div class="formItem">
    <label><spring:message code="trd.tradeentryaction.label.netamount" htmlEscape="false"/></label>
    <span class="detailDisplay"
name="tradeScreenData.netAmountStr">${commandForm.tradeScreenData.netAmountStr}</span>
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<!-- Net Amount in Trading Currency -->
<div class="formItem">
    <label><spring:message code="trd.tradeentryaction.label.netamountintradingcurrency"
htmlEscape="false"/></label>
    <span class="detailDisplay"
name="tradeScreenData.netAmountInTradingCurrency">${commandForm.tradeScreenData.netAmountInTrad
ingCurrency}</span>
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
<div class="clear clear-block"><spring:message text="" htmlEscape="false"/></div>
```

```

</div>
<div class="lineHide"><spring:message text="" htmlEscape="false"/></div>
</div>

```

6.2 Details page:

tradeDetailView.jspx

Expand

source

```


|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <table class="tableStyle"> <tbody> <tr> <td> <table class="detailBlock detailSmlBlkWidth"> <tbody> <tr> <td class="detailSmlBrdPad"> <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div></td></tr></tbody></table></td></tr></tbody></table> | <table class="detailBlock detailSmlBlkWidth"> <tbody> <tr> <td class="detailSmlBrdPad"> <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div></td></tr></tbody></table> | <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div> |
| <table class="detailBlock detailSmlBlkWidth"> <tbody> <tr> <td class="detailSmlBrdPad"> <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div></td></tr></tbody></table>                                                                        | <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div>                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <h1>&lt;spring:message code="trade.label.counterpartyaccount" htmlEscape="false"/&gt;</h1> <div class="detailItemBlock"> <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.customer.code" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.customerCode != oldScreenData.customerCode)}"&gt; &lt;span class="detail-view-hyperlink old" name="customerCode" href="/secure/ref/customer/details/\${oldScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${oldScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${oldScreenData.customerCode}&lt;/span&gt; &lt;span class="hyperlink-old-ico"&gt;&lt;/span&gt; &lt;span class="detail-view-hyperlink new" name="customerCode" href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;"&gt;\${newScreenData.customerCode}&lt;/span&gt; &lt;/c:when&gt;  &lt;c:otherwise&gt; &lt;span name="customerCode"&gt;&lt;a href="/secure/ref/customer/details/\${newScreenData.customerPk}?popup=true" view="customerDetailView" dialogTitle="[ \${newScreenData.customerCode} ] \${customerDetailTitle}" style="outline:0;" class="detail-view-hyperlink"&gt;\${newScreenData.customerCode}&lt;/a&gt;&lt;/span&gt; &lt;/c:otherwise&gt; &lt;/c:choose&gt; &lt;div class="clear"&gt;&lt;/div&gt; &lt;/div&gt; <div class="detailItem"> &lt;label&gt;&lt;spring:message code="trade.label.account.no" htmlEscape="false"/&gt;&lt;/label&gt; &lt;c:choose&gt; &lt;c:when test="\${(commandForm.diffEnableFlag == true) and (newScreenData.accountNo != oldScreenData.accountNo)}"&gt; &lt;span class="detail-view-hyperlink old" name="accountNo" href="/secure/ref/account/details/\${oldScreenData.accountPk}?popup=true" view="accountDetailView" dialogTitle="[ \${oldScreenData.accountNo} ] \${accountDetailTitle}" style="outline:0;"&gt;\${oldScreenData.accountNo}&lt;/span&gt; </div></div></div>                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |


```

```

<span class="hyperlink-old-ico"></span>
<span class="detail-view-hyperlink new" name="accountNo"
href="/secure/ref/account/details/${newScreenData.accountPk}?popup=true" view="accountDetailView"
dialogTitle="[${newScreenData.accountNo}] ${accountDetailTitle}"
style="outline:0;">${newScreenData.accountNo}</span>
</c:when>
<c:otherwise>
<span name="accountNo"><a href="/secure/ref/account/details/${newScreenData.accountPk}?popup=true"
view="accountDetailView" dialogTitle="[${newScreenData.accountNo}] ${accountDetailTitle}" style="outline:0;"
class="detail-view-hyperlink">${newScreenData.accountNo}</a></span>
</c:otherwise>
</c:choose>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.short.name" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="shortName" oldValue="${oldScreenData.shortName}"
newValue="${newScreenData.shortName}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.sales.code" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="salesCode" oldValue="${oldScreenData.salesCode}"
newValue="${newScreenData.salesCode}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.agent.code" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="agentCode" oldValue="${oldScreenData.agentCode}"
newValue="${newScreenData.agentCode}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.traderefERENCE" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message code="trade.label.reference.no" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="referenceNo"
oldValue="${oldScreenData.referenceNo}-${oldScreenData.versionNoStr}"
newValue="${newScreenData.referenceNo}-${newScreenData.versionNoStr}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.include.linked.reference.no" htmlEscape="false"/></label>
<span name="linkedReferenceNo"><a

```

```

href="/secure/trd/query/details/${newScreenData.linkedTradePk}/${newScreenData.linkedTradeReferenceNo}
}diffEnableFlag=false" view="tradeDetailTitle" dialogTitle="[${newScreenData.linkedTradeReferenceNo}]
${tradeDetailTitle}" style="outline:0;
class="detail-view-hyperlink">${newScreenData.linkedTradeReferenceNo}</a></span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.trade.status" htmlEscape="false"/></label>
<c:choose>
<c:when test="${commandForm.txnPkForAuthHistory != -1}">
<span name="status"></span>
</c:when>
<c:otherwise>
<valueDifference:valueDifferenceDetailsId fieldName="status" oldValue="${oldScreenData.status}"
newValue="${newScreenData.status}" diffEnableFlag="${commandForm.diffEnableFlag}">
</c:otherwise>
</c:choose>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.external.Refno" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="externalReferenceNo"
oldValue="${oldScreenData.externalReferenceNo}" newValue="${newScreenData.externalReferenceNo}"
diffEnableFlag="${commandForm.diffEnableFlag}">
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.data.source" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="dataSource" oldValue="${oldScreenData.dataSource}"
newValue="${newScreenData.dataSource}" diffEnableFlag="${commandForm.diffEnableFlag}">
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.currency" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message code="trade.label.trade.ccy" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="tradeCurrency"
oldValue="${oldScreenData.tradeCurrency}" newValue="${newScreenData.tradeCurrency}"
diffEnableFlag="${commandForm.diffEnableFlag}">
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.settlement.ccy" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="settlementCurrency"
oldValue="${oldScreenData.settlementCurrency}" newValue="${newScreenData.settlementCurrency}">
</div>

```

```

diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.exchange.rate" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="exchangeRate"
oldValue="${oldScreenData.exchangeRateStr}" newValue="${newScreenData.exchangeRateStr}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.issue.ccy" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="issueCurrency"
oldValue="${oldScreenData.issueCurrency}" newValue="${newScreenData.issueCurrency}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>

<div class="detailItem">
<label><spring:message code="trade.label.issuepricing.exchangerate" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="exchangeRateIssuePricing"
oldValue="${oldScreenData.exchangeRateIssuePricingStr}"
newValue="${newScreenData.exchangeRateIssuePricingStr}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>

<div class="detailItem">
<label><spring:message code="trade.label.execution.market" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="executionMarket"
oldValue="${oldScreenData.executionMarket}" newValue="${newScreenData.executionMarket}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>

<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
<td class="detailBlockMiddlePad">
<table class="tableStyle">
<tbody>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.inventoryaccount" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message code="trade.label.inventory.accno" htmlEscape="false"/></label>

```

```

<c:choose>
<c:when test="${(commandForm.diffEnableFlag == true) and (newScreenData.inventoryAccountNo != oldScreenData.inventoryAccountNo)}">
    <span class="detail-view-hyperlink old" name="inventoryAccountNo"
        href="/secure/ref/account/details/${oldScreenData.inventoryAccountPk}?popup=true"
        view="accountDetailView" dialogTitle="[ ${oldScreenData.inventoryAccountNo} ] ${accountDetailTitle}"
        style="outline:0;">${oldScreenData.inventoryAccountNo}</span><span class="hyperlink-old-ico"></span>
    <span class="detail-view-hyperlink new" name="inventoryAccountNo"
        href="/secure/ref/account/details/${newScreenData.inventoryAccountPk}?popup=true"
        view="accountDetailView" dialogTitle="[ ${newScreenData.inventoryAccountNo} ] ${accountDetailTitle}"
        style="outline:0;">${newScreenData.inventoryAccountNo}</span>
</c:when>

<c:otherwise>
    <span name="inventoryAccountNo"><a
        href="/secure/ref/account/details/${newScreenData.inventoryAccountPk}?popup=true"
        view="accountDetailView" dialogTitle="[ ${newScreenData.inventoryAccountNo} ] ${accountDetailTitle}"
        style="outline:0;" class="detail-view-hyperlink">${newScreenData.inventoryAccountNo}</a></span>
</c:otherwise>
</c:choose>
<div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.inventory.shortname" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="invAccShortName"
        oldValue="${oldScreenData.invAccShortName}" newValue="${newScreenData.invAccShortName}"
        diffEnableFlag="${commandForm.diffEnableFlag}" />
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
    <tbody>
        <tr>
            <td class="detailSmlBrdPad">
                <h1><spring:message code="trade.label.basictradeattributes" htmlEscape="false"/></h1>
                <div class="detailItemBlock">
                    <div class="detailItem">
                        <label><spring:message code="trade.label.trade.type" htmlEscape="false"/></label>
                        <valueDifference:valueDifferenceDetailsId fieldName="tradeType" oldValue="${oldScreenData.tradeType}"
                            newValue="${newScreenData.tradeType}" diffEnableFlag="${commandForm.diffEnableFlag}" />
                        <div class="clear"></div>
                    </div>
                    <div class="detailItem">
                        <label><spring:message code="trade.label.sub.trade.type" htmlEscape="false"/></label>
                        <valueDifference:valueDifferenceDetailsId fieldName="subTradeType"
                            oldValue="${oldScreenData.subTradeTypeDisp}" newValue="${newScreenData.subTradeTypeDisp}"
                            diffEnableFlag="${commandForm.diffEnableFlag}" />
                        <div class="clear"></div>
                    </div>
                    <div class="detailItem">
                        <label><spring:message code="trade.label.buySell.orientation" htmlEscape="false"/></label>

```

```

<valueDifference:valueDifferenceDetailsId fieldName="buySellOrientation"
oldValue="${oldScreenData.buySellOrientation}" newValue="${newScreenData.buySellOrientation}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.principleagent.flag" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="principalAgentFlag"
oldValue="${oldScreenData.principalAgentFlag}" newValue="${newScreenData.principalAgentFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.shortsell.flag" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="shortSellingFlag"
oldValue="${oldScreenData.shortSellingFlag}" newValue="${newScreenData.shortSellingFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.account.balancetype" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="accountBalanceType"
oldValue="${oldScreenData.accountBalanceTypeDisplay}"
newValue="${newScreenData.accountBalanceTypeDisplay}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.price" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message code="trade.label.input.price" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="inputPrice" oldValue="${oldScreenData.inputPriceStr}"
newValue="${newScreenData.inputPriceStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.input.priceformat" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="inputPriceFormat"
oldValue="${oldScreenData.inputPriceFormat}" newValue="${newScreenData.inputPriceFormat}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.price" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="price" oldValue="${oldScreenData.priceStr}"
```

```

newValue="${newScreenData.priceStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.prefigured.principalflag" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="prefiguredPrincipalFlag"
oldValue="${oldScreenData.prefiguredPrincipalFlag}" newValue="${newScreenData.prefiguredPrincipalFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.gross.nettype" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="grossNetType"
oldValue="${oldScreenData.grossNetType}" newValue="${newScreenData.grossNetType}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
<td>
<table class="tableStyle">
<tbody>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.securityinformation" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message code="trade.label.instrument.type" htmlEscape="false"/></label>
<valueDifference:valueDifferenceDetailsId fieldName="instrumentType"
oldValue="${oldScreenData.instrumentType}" newValue="${newScreenData.instrumentType}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message code="trade.label.security.code" htmlEscape="false"/></label>
<c:choose>
<c:when test="${(commandForm.diffEnableFlag == true) and (newScreenData.securityId != oldScreenData.securityId)}">
<span class="detail-view-hyperlink old" name="securityCode"
href="/secure/ref/instrument/details/${oldScreenData.instrumentPk}?popup=true"
view="InstrumentDetailView" dialogTitle="[ ${oldScreenData.securityId}] ${instrumentDetailTitle}"
style="outline:0;">${oldScreenData.securityId}</span><span class="hyperlink-old-ico"></span>
<span class="detailTxtVal detail-view-hyperlink new" name="securityCode"
href="/secure/ref/instrument/details/${newScreenData.instrumentPk}?popup=true"
view="InstrumentDetailView" dialogTitle="[ ${newScreenData.securityId}] ${instrumentDetailTitle}"
style="outline:0;">${newScreenData.securityId}</span>

```

```

    </c:when>

    <c:otherwise>
        <span name="securityCode"><a href="/secure/ref/instrument/details/${newScreenData.instrumentPk}?popup=true" view="instrumentDetailView" dialogTitle="[$(newScreenData.securityId)] ${instrumentDetailTitle}" style="outline:0;" class="detail-view-hyperlink">${newScreenData.securityId}</a></span>
    </c:otherwise>
</c:choose>
<div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.security.name" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="securityName" oldValue="${oldScreenData.officialName}" newValue="${newScreenData.officialName}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.alternative.security.code" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="alternateSecurityCode" oldValue="${oldScreenData.alternateSecurityCode}" newValue="${newScreenData.alternateSecurityCode}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.security.par.val" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="parVal" oldValue="${oldScreenData.parValStr}" newValue="${newScreenData.parValStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.date" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
    <label><spring:message code="trade.label.trade.date" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="tradeDate" oldValue="${oldScreenData.tradeDateStr}" newValue="${newScreenData.tradeDateStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.trade.time" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="tradeTime" oldValue="${oldScreenData.tradeTime}" newValue="${newScreenData.tradeTime}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>

```

```

<div class="detailItem">
    <label><spring:message code="trade.label.value.date" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="valueDate" oldValue="${oldScreenData.valueDateStr}"
    newValue="${newScreenData.valueDateStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1><spring:message code="trade.label.volume" htmlEscape="false"/></h1>
<div class="detailItemBlock">
<div class="detailItem">
    <label><spring:message code="trade.label.quantity" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="quantity" oldValue="${oldScreenData.quantityStr}"
    newValue="${newScreenData.quantityStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.principal.amount" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="principalAmount"
    oldValue="${oldScreenData.principalAmountStr}" newValue="${newScreenData.principalAmountStr}"
    diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.accrued.interest" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="accruedInterestAmount"
    oldValue="${oldScreenData.accruedInterestAmountStr}"
    newValue="${newScreenData.accruedInterestAmountStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.net.amount" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="netAmount"
    oldValue="${oldScreenData.netAmountStr}" newValue="${newScreenData.netAmountStr}"
    diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.principalamt.issueccy" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="principalAmountInIssueCurrency"
    oldValue="${oldScreenData.principalAmountInIssueStr}"
    newValue="${newScreenData.principalAmountInIssueStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/>
    <div class="clear"></div>
</div>
<div class="detailItem">
    <label><spring:message code="trade.label.netamount.trdccy" htmlEscape="false"/></label>
    <valueDifference:valueDifferenceDetailsId fieldName="netAmountInTradingCurrency"

```

```

oldValue="${oldScreenData.netAmountInTradingCurrency}"
newValue="${newScreenData.netAmountInTradingCurrency}"
diffEnableFlag="${commandForm.diffEnableFlag}"/>
<div class="clear"></div>
</div>
<div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>

```

.....

.....

```

<table class="tableStyle sixCol more printTableStyle tableBox" style="display:none;">
<tr>
<td><label name="orderrefno"><spring:message code="trade.label.order.refno"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.orderReferenceNo}"
newValue="${newScreenData.orderReferenceNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="etcrefno"><spring:message code="trade.label/etc.refno"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.etcReferenceNo}"
newValue="${newScreenData.etcReferenceNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="senderReferenceNo"><spring:message code="trade.label.sender.refno"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.senderReferenceNo}"
newValue="${newScreenData.senderReferenceNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="brokerReferenceNo"><spring:message code="trade.label.brokerrefno"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.brokerReferenceNo}"
newValue="${newScreenData.brokerReferenceNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="basketId"><spring:message code="trade.label.basketid"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.basketId}"
newValue="${newScreenData.basketId}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="projectNo"><spring:message code="trade.label.projectno"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.projectNo}"
newValue="${newScreenData.projectNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="executingBroker"><spring:message code="trade.label.broker"
htmlEscape="false"/></label></td>

```

```

<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.executingBroker}"
newValue="${newScreenData.executingBroker}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="principalType"><spring:message code="trade.label.principal.type"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.principalType}"
newValue="${newScreenData.principalType}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="executionMethod"><spring:message code="trade.label.execution.method"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.executionMethod}"
newValue="${newScreenData.executionMethod}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="clearingParty"><spring:message code="trade.label.clearing.party"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.clearingParty}"
newValue="${newScreenData.clearingParty}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="deliveryMethod"><spring:message code="trade.label.delivery.method"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.deliveryMethod}"
newValue="${newScreenData.deliveryMethod}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="asCustomerFlag"><spring:message code="trade.label.as.customerflag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.asCustomerFlag}"
newValue="${newScreenData.asCustomerFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="softDollarFlag"><spring:message code="trade.label.softdollar.flag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.softDollarFlag}"
newValue="${newScreenData.softDollarFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="shortSellExemptFlag"><spring:message code="trade.label.shortsell.exemptflag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.shortSellExemptFlag}"
newValue="${newScreenData.shortSellExemptFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="compressedTradeFlag"><spring:message code="trade.label.compressed.tradeflag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.compressedTradeFlag}"
newValue="${newScreenData.compressedTradeFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="matcheligible"><spring:message code="trd.tradeentryaction.label.matcheligible"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.matchEligible}"
newValue="${newScreenData.matchEligible}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="dvpEligible"><spring:message code="trade.label.dvp.eligible"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.dvpEligible}"
newValue="${newScreenData.dvpEligible}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="currentFaceValueStr"><spring:message code="trade.label.current.facevalue"
htmlEscape="false"/></label></td>

```

```

<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.currentFaceValueStr}"
newValue="${newScreenData.currentFaceValueStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="interestRateStr"><spring:message code="trade.label.interest.rate"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.interestRateStr}"
newValue="${newScreenData.interestRateStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="dirtyPriceFlag"><spring:message code="trade.label.dirtyprice.flag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.dirtyPriceFlag}"
newValue="${newScreenData.dirtyPriceFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="accruedStartDateStr"><spring:message code="trade.label.accrued.startdate"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.accruedStartDateStr}"
newValue="${newScreenData.accruedStartDateStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="accruedDaysStr"><spring:message code="trade.label.accrued.days"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.accruedDaysStr}"
newValue="${newScreenData.accruedDaysStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="startEndType"><spring:message code="trade.label.start.endtype"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.startEndType}"
newValue="${newScreenData.startEndType}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
<td><label name="exCouponFlag"><spring:message code="trade.label.excoupan.flag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.exCouponFlag}"
newValue="${newScreenData.exCouponFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="negativeAccruedInterestFlag"><spring:message code="trade.label.negative.accruedint.flag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.negativeAccruedInterestFlag}"
newValue="${newScreenData.negativeAccruedInterestFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="poolFactorStr"><spring:message code="trade.label.pool.factor"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.poolFactorStr}"
newValue="${newScreenData.poolFactorStr}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="stipulation"><spring:message code="trade.label.stipulation"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.stipulation}"
newValue="${newScreenData.stipulation}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="tipsAdjustPrice"><spring:message code="trade.label.tips.adjustprice"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.tipsAdjustPrice}"
newValue="${newScreenData.tipsAdjustPrice}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
<td><label name="tipsIndexAccruedInitDate"><spring:message code="trade.label.tips.indexissuedate"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.tipsIndexAccruedInitDate}"
```

```

newValue="${newScreenData.tipsIndexAccruedInitDate}"
diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
<td><label name="tipsIndexVd"><spring:message code="trade.label.tips.indexvaluedate"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.tipsIndexVd}"
newValue="${newScreenData.tipsIndexVd}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="stepInOutFlag"><spring:message code="trade.label.step.in/outflag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.stepInOutFlag}"
newValue="${newScreenData.stepInOutFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="tradeMatchType"><spring:message code="trade.label.trade.matchtype"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.tradeMatchType}"
newValue="${newScreenData.tradeMatchType}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="tradeConfirmReqd"><spring:message code="trade.label.trd.conf"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.tradeConfirmReqd}"
newValue="${newScreenData.tradeConfirmReqd}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="excludeFromNetting"><spring:message code="trade.label.excludefrom.netting"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.excludeFromNetting}"
newValue="${newScreenData.excludeFromNetting}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="traceEligibleFlag"><spring:message code="trade.label.trace.eligible"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.traceEligibleFlag}"
newValue="${newScreenData.traceEligibleFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="formTEligibleFlag"><spring:message code="trade.label.formt"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.formTEligibleFlag}"
newValue="${newScreenData.formTEligibleFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="forwardRepoFlag"><spring:message code="trade.label.forward.repo.flag"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.forwardRepoFlag}"
newValue="${newScreenData.forwardRepoFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="verbiagesSelectedString"><spring:message code="trade.label.verbiage.name"
htmlEscape="false"/></label></td>
<td colspan="3"><valueDifference:valueDifferenceld oldValue="${oldScreenData.verbiagesSelectedString}"
newValue="${newScreenData.verbiagesSelectedString}"
diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="remarksForReports"><spring:message code="trade.label.remark.forreports"
htmlEscape="false"/></label></td>
<td colspan="5"><valueDifference:valueDifferenceld oldValue="${oldScreenData.remarksForReports}"
newValue="${newScreenData.remarksForReports}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr>
<td><label name="mbsCCFlag"><spring:message code="trade.label.mbscc.flag"

```

```
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.mbsCCFlag}"
newValue="${newScreenData.mbsCCFlag}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="stlLocationSec"><spring:message code="trade.label.stllocation.security"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.stlLocationSec}"
newValue="${newScreenData.stlLocationSec}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="nsccReportEligibleFlag"><spring:message code="trade.label.nscc.reporteligible"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.nsccReportEligibleFlag}"
newValue="${newScreenData.nsccReportEligibleFlag}"
diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
<tr class="noBdr">
<td><label name="mbsscAccountNo"><spring:message code="trade.label.mbsccfirm.clearingac"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.mbsccAccountNo}"
newValue="${newScreenData.mbsccAccountNo}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>

<td><label name="stlLocationCash"><spring:message code="trade.label.stllocation.for.cash"
htmlEscape="false"/></label></td>
<td><valueDifference:valueDifferenceld oldValue="${oldScreenData.stlLocationCash}"
```

```

newValue="${newScreenData.stlLocationCash}" diffEnableFlag="${commandForm.diffEnableFlag}"/></td>
</tr>
</table>

```

3.4.6.2.23 Add a new Theme

3.4.6.2.24 Handle Constraint List & Values

1. Scope

Constraints are basically limits and restrictions imposed on any variable. With the help of constraints we define a set of limits which one can choose from. This document basically describes how to create a constraint and its values in database.

2. Pre-requisite

Developer should have knowledge about

- Spring Framework

3. Sequence of Steps

3.1 Setup in tables

Data needs to be created in two tables INF_CONSTRAINT_LIST and INF_CONSTRAINT_VALUES.

3.1.1 Table description

3.1.1.1 INF_CONSTRAINT_LIST

This table stores lists all the constraints.

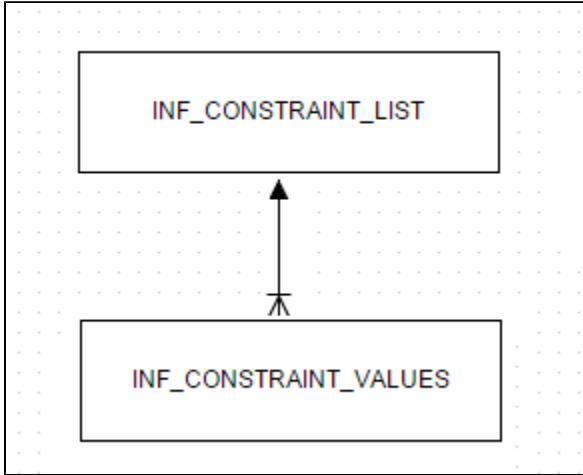
COLUMN_NAME	DATA_TYPE	NULLABLE	Remarks
CONSTRAINT_PK	NUMBER(4,0)	No	Primary key to uniquely identify a row.
CONSTRAINT_NAME	VARCHAR2(100 BYTE)	No	The name of the constraint

3.1.1.2 INF_CONSTRAINT_VALUES

This is a child table of INF_CONSTRAINT_LIST. It stores the list of constraint values of on any constraint present in INF_CONSTRAINT_LIST.

COLUMN_NAME	DATA_TYPE	NULLABLE	Remarks
CONSTRAINT_PK	NUMBER(4,0)	No	Foreign pk of INF_CONSTRAINT_LIST
CONSTRAINT_VALUE	VARCHAR2(100 BYTE)	No	The value of the constraint
LANGUAGE_CODE	VARCHAR2(15 BYTE)	No	The language code like Japanese, English for which this constraint will be displayed
DEFAULT_MSG	CHAR(1 BYTE)	Yes	Flag to denote if this is a default constraint to populate in message. While message processing if value is not present in the source message, then this value is populated.
DEFAULT_UI	CHAR(1 BYTE)	Yes	Flag to denote if this is a default constraint to populate in UI. Whenever a screen is loaded if the flag is set as Y then this constraint will be selected by default
DISPLAY_SEQ	NUMBER(2,0)	Yes	The display sequence of the constraints is mentioned here.
ENTERPRISE_ID	VARCHAR2(6 BYTE)	Yes	The enterprise id for which this constraint belongs.
UI_DISPLAY_VALUE	VARCHAR2(100 BYTE)	Yes	Display value in UI

3.1.2 Data model



One to many relation exists between INF_CONSTRAINT_LIST and INF_CONSTRAINT_VALUES.

3.1.3 Example to set up data in constraint tables

For example we have to populate drop down list in Trade Type field. A new records needs to be added in INF_CONSTRAINT_LIST with suitable constraint name like TRADE_TYPE.

INF_CONSTRAINT_LIST

CONSTRAINT_PK	CONSTRAINT_NAME
1001	TRADE_TYPE

1) After adding row in INF_CONSTRAINT_LIST table, all the constraint values needs to be defined in INF_CONSTRAINT_VALUES. In this table, add the pk of INF_CONSTRAINT_LIST table for TRADE_TYPE which is 1001. Now give constraint values in CONSTRAINT_VALUE column. Similarly create records for multiple constraints. We created FI and EQ as constraint values.

2) If developer wants some constraints to be shown when user preference of language is set a particular language then add language code of that language in LANGUAGE_CODE column. Give valid UI_DISPLAY_VALUE as this value will be shown in UI.

INF_CONSTRAINT_VALUES

CONSTRAINT_PK	CONSTRAINT_VALUE	DEFAULT_MSG	DEFAULT_UI	DISPLAY_SEQ	ENTERPRISE_ID	LANGUAGE_CODE	UI_DISPLAY_VALUE
1001	FI			1		EN	EXTERNAL
1001	EQ			2		EN	INTERNAL
1001	FI			1		JP	定の収入
1001	EQ			2		JP	定の収入

Say Trade Type drop down is present in Trade entry screen. The above example mentions:-

1) Whenever language preference is set as English, then EXTERNAL and INTERNAL will be displayed and when language preference is set to Japanese, then 定の収入, エクイティ be displayed.

2) Whenever language preference is set as English, the display sequence is such that EXTERNAL is shown before INTERNAL in the Trade Type drop down. This display order is maintained by DISPLAY_SEQ column.

3.2 Different Server side API and Utilities

3.3.1 List of already developed API

API	Purpose and usages	Scenario examples

Map<String, String> getConstraintValues(String constraintName, String enterpriseId)	This method returns Map of constraint value and constraint display value ordered by display sequences mentioned in the database. It first tries to get the constraint values user locale. If no constraint values found with given locale it tries to get constraint values with default Locales.	
Map<String, String> getDefaultConstraintValueMap(String constraintName, String enterpriseId)	This method returns the map of default constraint values against the constraint name for the enterprise and user locale.	
String getDefaultDisplayValue(String constraintName, String enterpriseId)	This method returns the constraint display value based on the provided constraintName.	
String getDefaultConstraintValue(String constraintName, String enterpriseId)	This method returns the constraint value based on the provided constraintName.	

3.3.2 Different scenarios of constraints handling

3.3.2.1 Populating Settlement Mode drop down

Developer wants to add a drop down say Trade Type in Trade Query screen and populate the drop down list from the values set in INF_CONSTRAINT_LIST and INF_CONSTRAINT_VALUES. Developer should first create records in INF_CONSTRAINT_LIST and INF_CONSTRAINT_VALUES.

In the command form of trade query, define the settlement mode as list and also define the getter() and setter() methods. For trade query it is TrdQueryCommandForm.java

```
private Map<String, String> tradeTypeList = null;
--
--
public Map<String, String> getTradeTypeList() {
    return tradeTypeList;
}

public void setTradeTypeList(Map<String, String> map) {
    this.tradeTypeList = map;
}
```

In the controller of trade query which is TrdQueryController.java the constraint list TRADE_TYPE is passed as parameter. This method fetches the list of constraint values from INF_CONSTRAINT_VALUES based on the constraint list passed and the user preference of language and then sets into tradeTypeList

```
protected void doInit(TrdQueryCommandForm form, FinchModelMap modelMap)
    throws FinchException {
try {
    Identity principal = null;
    if (Operation.getInstance() != null
        && Operation.getInstance().getContext() != null) {
        principal = ((OperationScope) Operation.getInstance()
            .getContext()).getCallerIdentity();
    }
    // other code block ..
    form.setTradeTypeList(principal.getUserConstraint().get(
        "TRADE_TYPE"));
}
}
```

Developer needs to add trade type in tradeQueryCriteria.jspx.

```

<div class="formItem">
    <form:label path="tradeType">
        <spring:message code="trade.label.trade.type" htmlEscape="false" />
    </form:label>
    <span>
        <form:select path="tradeType" class="dropdowninput">
            <form:option value="" />
            <form:options items="${commandForm.tradeTypeList}" />
        </form:select>
    </span>
    ..
    ..
</div>

```

4. References

- Spring MVC

5. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	04-02-2015	Initial version	Mithu Tokder

3.4.6.2.26 UI Component

3.4.6.2.26.1 Basic Layout Structure

1. Introduction

finch framework consists of forms and detail views. In other words, finch has two types of views :

- Editable views which include the entry and query forms
- Non-editable views which include the result pages, confirmation pages and detail dialogs

finch standard guideline specifies the combination of the HTML elements and the appropriate style that needs to be applied to them.

In order to create the basic layout of the new editable or non-editable pages, developer must follow the finch standard guidelines. Additionally, developer can add the page specific elements according to requirement.

2. Prerequisites

MUST SEE

Refer to [Web Component](#) for an overall idea about the finch web architecture.
Refer to [Web Layer](#) for the implementation details of web components.

3.4.6.2.26.1.1 Building Forms

1. Introduction

The forms used in finch framework can be broadly categorized under three types:

- Entry/Amend forms - Forms used to input all field values required to create a new entity like Employee,Trade, Account etc.
- Popup forms - Forms used to input search parameters to retrieve matching records for data like Account Number, Instrument Code etc. These data is required for the query forms like Employee Query, Trade Query etc.
- Query forms - Forms used to input search parameters to retrieve matching records for an existing entity like Employee,Trade, Account etc.

3.4.6.2.26.1.1.1 Entry/Amend Form

1. Scope

The entry /amend page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)
- Grid (optional)

This document gives the details of the markup required to create the entry/amend form.

To implement a entry/amend page for a particular module, developers only need to define the form fields in the section as shown below :

2. Prerequisites

MUST SEE

[View templates](#) for the details on the templates used in finch.

[UI Component](#) for the details on how to use the UI components in finch

[CSS](#) for the CSS related details.

3. Sequence of Steps

Please note that we will use the Employee Entry use case as a reference for this document.

3.1 Markup for form container and blocks

The code snippet below from the view "userEntry.jspx" shows the employee entry form container containing:

- The script section for page initialization logic
- The two form item blocks represented by two `<div>` elements having a class "entryBlkArea". These blocks form the outer container of the form elements.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx
```

```
<div id="formContainer" class="entryContainer paddingFour" ... >
  <jsp:element name="script">
    <jsp:body>
      /* page initialization logic goes here */
      .
      .
    </jsp:body>
  </jsp:element>

  <!-- Markup for the employee entry page goes here -->
  <!-- First form block/section starts here -->
  <div class="entryBlkArea">
    ...
  </div>
  <!-- Second form block/section starts here -->
  <div class="entryBlkArea">
    ...
  </div>
</div>
```

3.2 Markup for form item blocks

<div> element having a class `formItemBlock` forms the inner container of the form elements. Within the form item block, the individual form items are placed.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx
```

```
<!-- First form block/section starts here -->
<div class="entryBlkArea">
<div class="formItemBlock dottedBg">
<div class="formItem">
...
</div>
</div>
</div>
```

3.3 Markup for form items

Each individual form item consists of the form label and its associated form element wrapped in a `span`.

<form:label> having a class "required" signifies that its associated form element is mandatory.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx
```

```
<div class="formItem">
<form:label path="commandForm.employeeDTO.defaultBranch" class="required">
<spring:message code="inf.employeeentryaction.label.defaultbranch" htmlEscape="false" />
</form:label>
<form:select id="defaultBranch" path="commandForm.employeeDTO.defaultBranchPk"
class="dropdowninput" defaultValue="defaultBranch">
<form:option value="" ></form:option>
<form:options items="${commandForm.branches}" />
</form:select>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
```

Reference

See [Form Elements](#) document for markup details of individual form components like label, text-input, text-area, date picker, popup, tree-component etc.

3.4 Markup for grid

<div> element having a class "finch-grid" serves as a container for finch grid component.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx
```

```
<div class="entryGrid">
<div id="userApplRoleGrid" class="finch-grid" style="width: 100%; height: 120px;">
<spring:message text="" htmlEscape="false" />
</div>
</div>
```

3.5 Markup for action buttons

<div> element having a id "formActionArea" serves as a wrapper for form action buttons.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx

<div id="formActionArea" class="gridBtnMargin">
  <div class="right">
    <div class="btnWrapStyle submitBtn gridAddBtn">
      <span class="addBranchRole" style="display: block"><input id="addBranchRoleBtn" type="button" value="${formadd_label}" class="inputBtnStyle" /></span>
    </div>
    <span class="gridEditBtn" style="display: none">
      <div class="btnWrapStyle submitBtn">
        <input id="updateBranchRoleBtn" type="button" value="${formsave_label}" class="inputBtnStyle" />
      </div>
      <div class="btnWrapStyle resetBtn">
        <input id="cancelBranchRoleBtn" type="button" value="${formcancel_label}" class="inputBtnStyle" />
      </div>
    </span>
  </div>
</div>
```

3.6 Markup for popups

In Trade Entry page, popups like currency popup, account popup etc. are required.

<div> element having a class "popupBtn" serves as a wrapper for the button used to launch the popup. <input> button having a class "popupBtnlco" is used for launching the popup.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userEntry.jspx

<!-- Trade Currency -->
<div class="formItem">
  <form:label path="commandForm.tradeDTO.tradeCurrency" class="required"><spring:message code="trd.tradeentryaction.label.tradecurrency" htmlEscape="false"/></form:label>
  <span><form:input id="tradeCcy" path="commandForm.tradeDTO.tradeCurrency" class="textBox" /></span>
  <div class="popupBtn">
    <input id="trdCcyPopupButton" type="button" class="popupBtnlco amendReadOnlyPopup" popType="result" popFor="currency" value="" />
  </div>
  <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
```

Reference

See [Use a pop-up document for details on how to use all popup components in finch.](#)

CSS Modification

If any css modification is required, then include it inline in the element itself. Do not change the core css classes.

4. See Also

For more details refer to the following :

<http://www.w3.org/TR/CSS21/> on CSS2 specifications

<http://api.jquery.com/> on jQuery

<http://www.willcam.com/cmat/html/crossref.html> on HTML tags

<http://www.dzone.com/tutorials/java/spring/spring-form-tags-1.html> on Spring form tags

<http://static.springsource.org/spring/docs/current/> on all Spring references

<http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example> on handling spring mvc forms

3.4.6.2.26.1.1.2 Popup Form

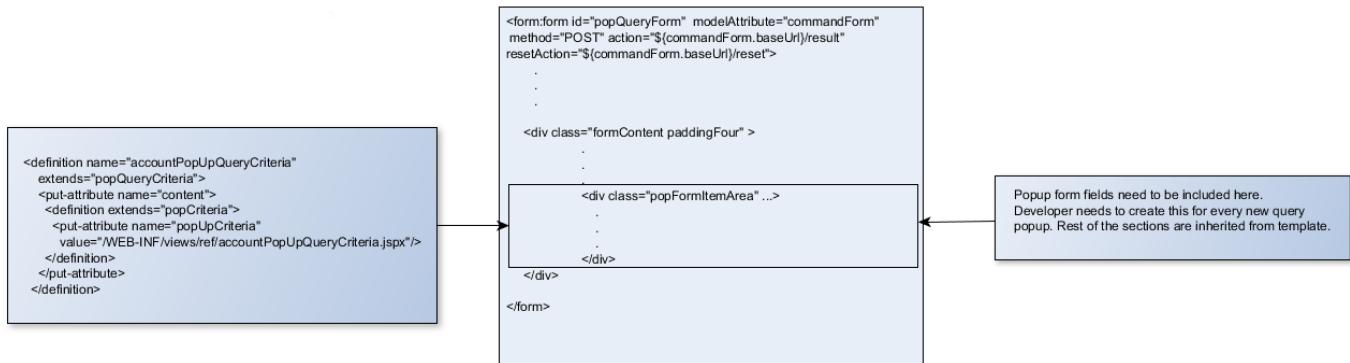
1. Scope

The query popup consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Popup (optional)

This document gives the details of the markup required to create the query popup form.

To implement a query popup form, developers only need to define the form fields in the section as shown below :



2. Prerequisites

MUST SEE

[View templates](#) for the details on the templates used in finch.

[UI Component](#) for the details on how to use the UI components in finch

[CSS](#) for the CSS related details.

3. Sequence of Steps

Please note that we will be using the "accountPopUpQueryCriteria.jspx" as a reference for this document.

3.1 Markup for form container

<div> element having class "popFormItemArea" forms the parent container for the form items.

```
sample-ref/src/main/web/WEB-INF/views/ref/accountPopUpQueryCriteria.jspx
```

```
<div class="popFormItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core"
.
.
.
</div>
```

3.2 Markup for form items

<div> element having class "popFormItem" forms the container for the individual form elements.

```
sample-ref/src/main/web/WEB-INF/views/ref/accountPopUpQueryCriteria.jspx
```

```
<div class="popFormItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core"
<div class="popFormItem">
    <form:label path="defaultShortName"><spring:message code="ref.label.account.nameshort"
htmlEscape="false"/></form:label>
    <span><form:input value="" path="defaultShortName" class="textBox" /></span>
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
</div>
```

3.3 Action buttons

The popup buttons and their classes are defined in finch-popquery-form.js as shown below :

```
finch-web-infra/src/main/web/scripts/inf/finch-popquery-form.js
```

```
var popupbtms;
if(targetFieldId.length == 1) {
    popupbtms= [
        {
            text: FINCH.title.resetBtnLabel,
            'class': "popResetBtn",
            click: resetHandler
        },
        {
            text: FINCH.title.submitBtnLabel,
            'class': "popSubmitBtn",
            click: submitHandler
        },
        {
            text: FINCH.title.requeryBtnLabel,
            'class': "popRequery",
            click: requeryHandler
        },
        {
            text: FINCH.title.selectBtnLabel,
            'class': "popSelect",
            click: selectHandler
        }];
} else {
    popupbtms = [
        {
            text: FINCH.title.cancelBtnLabel,
            'class': "popResetBtn",
            click: function(){
                tag.dialog('destroy');
                tag.remove();
                btn.removeAttr("disabled");
            }
        }];
}
```

The popup buttons are used to initialize the ui-dialog plugin as shown below :

finch-web-infra/src/main/web/scripts/inf/finch-popquery-form.js

```
tag.dialog({title : title,
  modal: true,
  width:popupSettings.width,
  height:popupSettings.height,
  zIndex:10000,
  resizable : true,
  position:'center',
  closeOnEscape:false,
  close: function(event, ui) {
    if (typeof FINCH$Popup$Destroy$Hook !== 'undefined') {
      FINCH$Popup$Destroy$Hook.call(tag);
      FINCH$Popup$Init$Hook = undefined;
      FINCH$Query$Popup$Pre$Reset$Hook = undefined;
      FINCH$Query$Popup$Post$Reset$Hook = undefined;
      FINCH$Query$Popup$Pre$Submit$Hook = undefined;
      FINCH$Query$Popup$Post$Submit$Hook = undefined;
      FINCH$Query$Popup$Pre$Requery$Hook = undefined;
      FINCH$Query$Popup$Post$Requery$Hook = undefined;
      FINCH$Query$Popup$Pre$Select$Hook = undefined;
      FINCH$Query$Popup$Post$Select$Hook = undefined;
      FINCH$Popup$Destroy$Hook = undefined;
    }
    if(typeof FINCH.generalPopupGrids !== 'undefined') {
      $.each(FINCH.generalPopupGrids, function(index,currentGrid){
        currentGrid.destroyGrid();
      });
      delete FINCH.generalPopupGrids;
    }
    if(typeof popGrid !== 'undefined'){
      popGrid.destroyGrid();
    }
    if($('.div.ui-dialog-titlebar').children('.popupFormTabErrorIcon').length > 0) {
      FINCH.clearNotice();
    }
    $('.ui-dialog.topMost').off('focus');
    tag.dialog('destroy');
    tag.remove();
    btn.removeAttr("disabled");
    targetFieldIdStack.pop();
    popupClassArray.pop();
    btnArray.pop().focus();
  },
  buttons: popupbtms
}).dialog('open').keypress(function(e){
  if ((e.which && e.which == 13) || (e.keyCode && e.keyCode == 13)) {
    e.preventDefault();
    if( $('.popSubmitBtn',container).is(':visible') ) {
      $('.ui-dialog.topMost .ui-dialog-buttonpane .popSubmitBtn').click();
    }
  }
});
```

Popup buttons

Developer need not include any markup for the popup buttons. These buttons are configured in finch-popquery-form.js.

Query result page

The query popup result "accountPopUpQueryResult.jspx" contains the grid initialization parameters. Developer need not include any markup for this page. See the [Grid Component](#) document for further details.

4. See Also

For more details refer to the following :

<http://www.w3.org/TR/CSS21/> on CSS2 specifications

<http://api.jquery.com/> on jQuery

<http://www.willcam.com/cmat/html/crossref.html> on HTML tags

<http://www.dzone.com/tutorials/java/spring/spring-form-tags-1.html> on Spring form tags

<http://static.springsource.org/spring/docs/current/> on all Spring references

<http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example> on handling spring mvc forms

3.4.6.2.26.1.1.3 Query Form

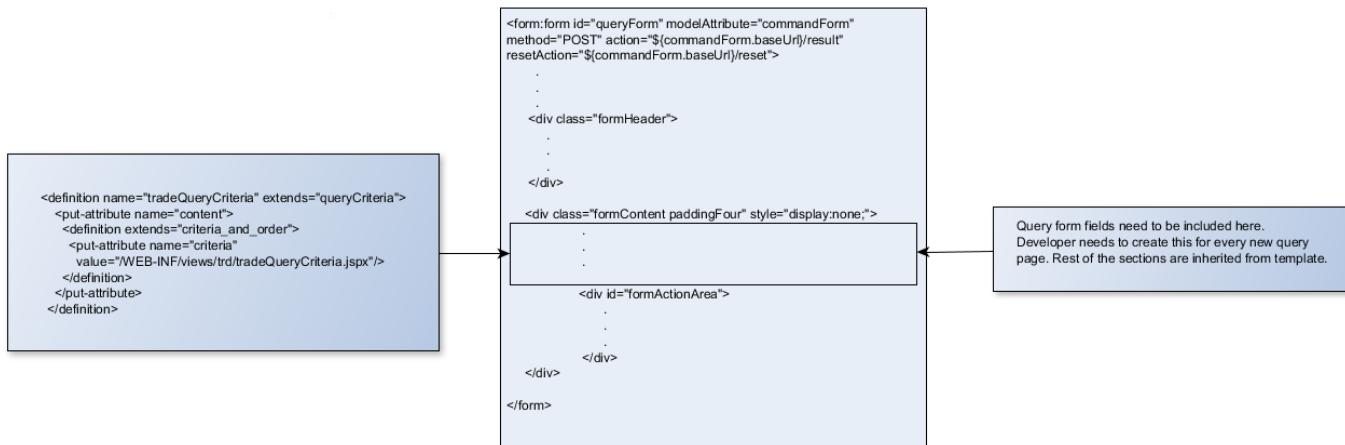
1. Scope

The query page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Tree View (optional)
- Popup (optional)

This document gives the details of the markup required to create the query form.

To implement a query page for a particular module, developers only need to define the form fields in the section as shown below :



2. Prerequisites

MUST SEE

[View templates](#) for the details on the templates used in finch.

[UI Component](#) for the details on how to use the UI components in finch

[CSS](#) for the CSS related details.

3. Sequence of Steps

Please note that we will be using the "tradeQueryCriteria.jspx" as a reference for this document.

Every query criteria page needs to have a basic DOM structure which is a must before inserting any page specific form item.

A query page can have the below sections/blocks:

- Top block (mandatory)
- Collapsible More block (optional)

Purpose of each DOM is mentioned as comments in the below code snippets.

3.1 Markup for form container of top block

<div> element having classes "formItemBlock" and "topitems" form the container for the top block of the query form.

```
sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core" ...>
    .
    .
    .

    <!-- Container for all the form items in the top Block -->
    <div class="formItemBlock topitems">
        <!-- All Form items for top block goes here -->
        <div class="clear clear-block"><spring:message text="" htmlEscape="false"/></div>
    </div>
    <!-- Used to Prevent Overlapping of two divs -->
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>

    <!-- Added for styling purpose -->
    <div class="formBrdHide"><spring:message text="" htmlEscape="false"/></div>
</div>
```

3.2 Markup for form in the personalization mode

In personalization mode each of the non-mandatory items in the query form is followed by a cross icon to remove the field. In case the field is removed, then a section is displayed with the option to restore the deleted fields. In order to achieve this, a <div> element having classes "delFormItems" and "deltopitems" are used as the container for the restore section.

```
sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core" ...>
    .
    .
    .

    <!--This div is only displayed, when the form is in personalization mode, for framework use only, do not
Add/Edit any item-->
    <div class="formItemBlock delFormItems deltopitems" style="display:none;" >
        <!-- Restore All Button. -->
        <input type="button" value="Restore All" />
        <div class="clear clear-block"><spring:message text="" htmlEscape="false"/></div>
    </div>
</div>
```

3.3 Section to give an unique id to every instance of a query form

Each instance of the query form needs an unique id. This unique id is required to initialize the finchform plugin for the query page.

```
sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core" ...>
    .
    .
    .
    <!-- This section gives a unique id to every form, which in turn is used to identify screens, when doing DOM operations. -->
    <spring:eval expression="T(java.lang.System).currentTimeMillis()" var="myId"/>
    <input type="hidden" id="${myId}" />
    <jsp:element name="script">
        <jsp:attribute name="type">text/javascript</jsp:attribute>
        <jsp:body>
            <!-- Call to finchform plugin is a must, else the query form won't be displayed. It is this plugin which contains most of the event handlers. -->
            jQuery('#' + ${myId}).parent().finchform({datepickerOptions:{changeMonth: true,changeYear: true}});
        </jsp:body>
    </jsp:element>
</div>
```

3.4 Markup for form container of collapsible more block and the clickable +/- icon

More block should only be added, in case the form needs the collapsible more block in addition to the top block. `<div>` element having classes "formItemBlock" and "more" form the container for the more block of the query form.

```
sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

<div class="formItemArea" xmlns:c="http://java.sun.com/jsp/jstl/core" ...>
    .
    .
    .
    <!--More Block: Begin-->
    <!-- This DOM contains the divider, and clickable more handle, clicking on which expands/collpase the more block. -->
    <div class="moreHandle">
        <div class="handleBlock">
            <span title="Expand" class="handler"><spring:message text="" htmlEscape="false"/></span>
        </div>
    </div>
    <div class="formItemBlock more">
        <div class="moreitems">
            <!-- All Form items for more block goes here -->
            <div class="clear clear-block"><spring:message text="" htmlEscape="false"/></div>
        </div>
        .
        .
    </div>
```

Sorting Block

The query form also optionally includes the sorting block. This block is inherited from the "queryCriteria" template. Developer need not include any markup for this block. See [Form Elements](#) Sort field control section for further details.

3.5 Markup for adding page specific form items

In the Trade Query page, the top and the more blocks are the containers of the form items. So after implementing the basic query screen, developer can add the page specific form items.

The steps involved in adding the new form items include:

1. Identification of the location of the form item:

The new form item can be placed either in the top block within the div which has a class named "topitems" or in the more block within the div which has a class named "moreitems".

2. Placing the form item inside the chosen block:

For eg. Lets say developer wants to add Trade Reference Number input field in the top block, then the markup required to be inserted in the top block is shown below:

```
sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx

<!-- Form Item Block for Trade Reference Number-->
<div class="formItem">
    <!-- Label -->
    <form:label path="referenceNo"><spring:message code="trade.label.trade.referenceNo"
htmlEscape="false"/></form:label>
    <!-- Input field -->
    <span><form:input value="" path="referenceNo" class="textBox" /></span>

    <!-- Used during personalization -->
    <!-- Remove this block in case the new field is a mandatory field/developer does not want to allow
the user to remove this form item during personalization-->
    <span class="remove" title="Remove this field" style="display: none;"><spring:message text=""
htmlEscape="false"/></span>

    <!-- Used to prevent overlapping of two div blocks -->
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>
</div>
```

Query result page

The query result page "tradeQueryResult.jspx" consists of the grid component. Developer need not include any markup for this page. See the Grid Component document for further details.

4. See Also

For more details refer to the following :

<http://www.w3.org/TR/CSS21/> on CSS2 specifications

<http://api.jquery.com/> on jQuery

<http://www.willcam.com/cmat/html/crossref.html> on HTML tags

<http://www.dzone.com/tutorials/java/spring/spring-form-tags-1.html> on Spring form tags

<http://static.springsource.org/spring/docs/current/> on all Spring references

<http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example> on handling spring mvc forms

3.4.6.2.26.1.2 Detail Page Layout

1. Scope

The detail block displays the form elements and their associated labels in a non-editable tabular format. In finch it is suggested to reuse the same page style for all of the items as follows :

- User Confirmation
- System Confirmation
- Cancel Operation

This document gives the details of the markup required to create the detail page.

2. Prerequisites

MUST SEE

[View templates](#) for the details on the templates used in finch.

[UI Component](#) for the details on how to use the UI components in finch.

[CSS](#) for the CSS related details.

3. Sequence of Steps

Please note that we will be using the Employee Entry use case as a reference for this document.

3.1 Markup for form container and blocks

The code snippet from the existing view "userDetail.jspx" shows the values input in the employee entry form in the tabular format.

The entire confirmation page markup needs to be wrapped in a parent div element having classes "formItemArea", "entryContainerConfirm", "paddingFour". The individual blocks of information is wrapped in a div element having classes "detailBlock ", "entrySingleCol".

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userDetail.jspx

<div class="formItemArea entryContainerConfirm paddingFour" ...>
  .
  .
  .
  <div class="detailBlock entrySingleCol">
    .
    .
    .
  </div>
</div>
```

"entryContainerConfirm" class denotes the container div for the confirmation page.

Each entry page may consist of more than one block of information. For example, in employee entry page the top block contains employee information and the lower block contains application role information.

In the details page, "detailBlock" class denotes the container div for each such block of information.

3.2 Markup for details table

Within the individual block of information, the tabular data is present in columns containing the label and the value of the form element from the employee entry form . The table has a class "tableStyle".

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userDetail.jspx
```

```
<table class="tableStyle sixCol">
<tr>
<td><label><spring:message code="inf.employee.label.userid"
    htmlEscape="false" /></label></td>
<td><span class="detailTxtVal">${commandForm.employeeDTO.userId}</span></td>
<td><label><spring:message
    code="inf.employee.label.firstname" htmlEscape="false" /></label></td>
<td><span class="detailTxtVal">${commandForm.employeeDTO.firstName}</span></td>
<td><label><spring:message
    code="inf.employee.label.middleinitial" htmlEscape="false" /></label></td>
<td><span class="detailTxtVal">${commandForm.employeeDTO.middleInitial}</span></td>
</tr>
.
.
.
.
</table>
```

3.3 Markup for details table columns

The detail values are displayed as label and values in successive table columns. The values of form elements are wrapped in a span having a class "detailTxtVal".

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userDetail.jspx
```

```
<td><label><spring:message code="inf.employee.label.lastname" htmlEscape="false" /></label></td>
<td><span class="detailTxtVal">${commandForm.employeeDTO.lastName}</span></td>
```

"tableStyle" class denotes the table for displaying the entry form information.

Each row of this table consists of six columns. First column contains the label, second column contains the form element value. This pattern is followed for all the form elements.

"detailTxtVal" class denotes the class for form element values.

3.4 Markup for grid with data in details page

If a grid is present in the entry form, then its information is displayed in a table having a class "reportTbl".

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userDetail.jspx
```

```
<table class="reportTbl">
<tr>
<th><spring:message code="inf.employee.label.branchid"
    htmlEscape="false" /></th>
<th><spring:message code="inf.employee.label.applicationsrole"
    htmlEscape="false" /></th>
</tr>
<c:forEach items="${commandForm.userApplRoleWithBranchList}" var="dl">
<tr>
<td>${dl.branch}</td>
<td>${dl.role}</td>
</tr>
</c:forEach>
</table>
```

"reportTbl" class denotes the table for displaying the grid data in a details page.

3.5 Markup for grid with no data in details page

If a grid is present in the entry form with no records in it, then an empty table is shown.

```
finch-web-infra/src/main/web/WEB-INF/views/inf/user/userDetail.jspx
```

```
<table class="reportTbl">
<c:if test="${(empty commandForm.emppage.defaultOfficeStr)}">
<tr>
<td class="noRecord" colspan="10">
    <spring:message code="igv.no.record" htmlEscape="false"/>
</td>
</tr>
</c:if>
.
.
.
</table>
```

"noRecord" class denotes the table for displaying the no record message in a details page.

See [Grid Component](#) for further details on how to use the grid component.

4. See Also

For more details refer to the following :

<http://www.w3.org/TR/CSS21/> on CSS2 specifications

<http://api.jquery.com/> on jQuery

<http://www.willcam.com/cmat/html/crossref.html> on HTML tags

<http://www.dzone.com/tutorials/java/spring/spring-form-tags-1.html> on Spring form tags

<http://static.springsource.org/spring/docs/current/> on all Spring references

<http://www.mkyong.com/spring-mvc/spring-mvc-form-handling-example> on handling spring mvc forms

3.4.6.2.26.2 Dashboard Component

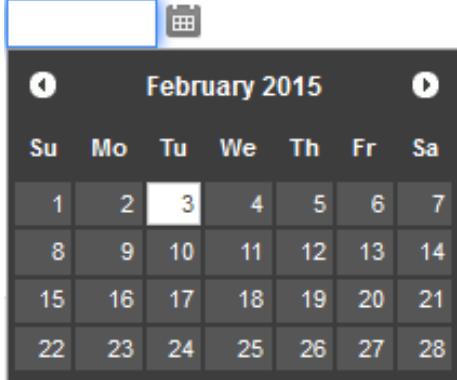
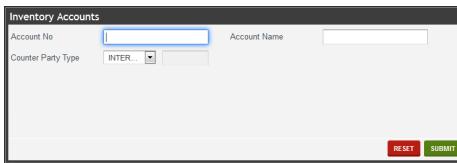
3.4.6.2.26.2.1 Add New Business Feed

3.4.6.2.26.3 Form Elements

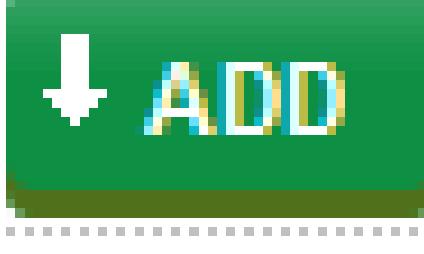
1. Scope

This document describes how to use the form elements in application that use finch.

Currently, in finch, following kind of form input items are available:

Sl. No.	Input Type	
1	Text input control	<input type="text"/>
2	Drop-down input control	<input type="text"/> 
3	Date Picker control	<input type="text"/>  
4	Popup control	<input type="text"/>  
5	Sort field control	 <input type="text"/>  <input type="checkbox"/> 

6	Label	
7	Label indicating mandatory fields	
8	Color Picker	
17	Submit Button	
18	Reset button	
19	Previous button	

20	Next button	
21	Back button	
22	Confirm button	
23	Ok button	
24	Add button	

2. Prerequisites

3. Controls

3.1 Text Input control

Text input controls are used for simple text input from the user. First Name input box is an example of text input control.

```
<form:input id="firstName" path="commandForm.employeeDTO.firstName" class="textBox"/>
```

class="textBox" signifies that it is a text input control.

3.2 Drop-down input control

The usage of drop-down input control is to select an item from list. Status, Trade type are the example of drop-down input control.

```
<form:select id="tradeType" path="commandForm.tradeDTO.tradeType" class="dropdowninput">
    <form:option value="" />
    <form:options items="${commandForm.tradeDTO.tradeTypeValues}" />
</form:select>
```

class="dropdowninput" signifies that it is a drop down input control.

3.3 Date Picker control

Purpose of Date Picker control is to input date from user. User can input a single date or composite date (from-to).

For Single date

```
<form:input id="tradeDate" path="commandForm.tradeDateStr" class="dateinput" />
```

For composite date (from-to)

```
<span><form:input id="tradeDateFrom" value="" path="tradeDateFrom" class="dateinput noOptions" /></span>
<span><form:input id="tradeDateTo" value="" path="tradeDateTo" class="dateinput noOptions" /></span>
```

class="dateinput" signifies that it is a date picker combo.

3.4 Popup control

All popups are composite components. By clicking icon beside the text input, user can select value from popup.

The code snippet below shows the markup needed to include a account popup in a form :

```
<span><form:input id="accountNo" path="commandForm.tradeDTO.accountNo" class="textBox" /></span>
<div class="popupBtn">
    <input id="accountPopupBtn" type="button" class="popupBtnlco" popType="query" popFor="cpAccount"
    actTypeContext="T|B" actCPTYPEContext="CLIENT|BROKER|INTERNAL" actStatusContext="OPEN" value="" />
</div>
```

Reference

The popup launch button has the following custom attributes:

- popType
- actTypeContext
- actCPTYPEContext
- actStatusContext

For server side call application has to configure the following :

Popup button handler

```
SaccountNoBtnObj = $('#accountPopupBtn');
SaccountNoBtnObj.die();
SaccountNoBtnObj.live('click', function(e){
    var setting = {
        'id': 'account',
        'title' : SAMPLE$REF$i18n.title.accPopup,
        'url' : '/ref/account/popup/query',
        'extUrlParams' : prepareAccDependentParams($accountNoBtnObj),
        'tgt' : '#accountNo',
        'multiSelect' : true
    };
    openPopUpForm(e, setting);
});
```

In the above code openPopUpForm() function is called with two parameter event(e) and setting. openPopUpForm() function is provided by finch framework, Make sure "finch-popquery-form.js" file is loaded before call to this function.

setting has the following attribute

- 1) id - unique id
- 2) title - title to be displayed in the popup form
- 3) url - url for controller mapping
- 4) extUrlParams - data to be appended in the url
- 5) tgt - id of the form input where the value will be saved after selecting value(s) in the popup form
- 6) multiSelect - flag whether multiple values can be selected

setting has the following attribute as well

- width - Integer value for the popup width
- height - Integer value for the popup height
- isTreeView - boolean
- isDateValidation - boolean

class="popupBtlnCo" signifies that it is a button to launch the popup.

3.5 Sort field control

Sort field control is a composite field, consisting of one or more sort order components. Sort field controls are present in the query pages like Trade Query, Employee Query etc.

For example,

In trade query page, tradeQueryCriteria.jspx inherits from criteriaOrder.jspx template. This template has a placeholder to insert the sort field control based on the \${commandForm.sortDetail} value as shown below :

criteriaOrder.jspx

```
<c:if test="${not empty commandForm.sortDetail}">
<tiles:insertAttribute name="order" />
</c:if>
```

Each sort order component has two parts, drop-down list containing sorting field options and check box determining order (desc or asc).

Finch supports dynamic sort field in query screen.

Sort field controls are defined in order.jspx as shown below

```
order.jspx

<c:forEach items="${commandForm.sortDetail}" var="dl" varStatus="index">
    <div class="formItem">
        <span class="orderCount">${index.count}</span>
        <span>
            <form:select path="${dl}" class="dropdowninput" property="${dl}" defaultValue="sortCriteria">
                <form:option value="" label="" />
                <form:options items="${commandForm.sortFieldList}" itemLabel="label" itemValue="value" />
            </form:select>
        </span>
        <spring:message code="global.pref.label.desc" htmlEscape="false" var="desc_label" />
        <span class="desc">
            <form:checkbox path="sortFieldOrder${index.count}" label="${desc_label}"
value="DESC"></form:checkbox>
        </span>
        <div class="clear">
            <spring:message text="" htmlEscape="false" />
        </div>
    </div>
</c:forEach>
```

Configuration for sort fields are provided from server side. So for the individual pages in the client-side, developer does not need to do anything for sort field control if there are 6 sort fields.

If there are more than 6 sort fields then following is to be done from application side

- Set total number of sort fields required in application QueryCommandForm's constructor by using setNumOfSortFields(n) method. Example:

```
public TrdQueryCommandForm() {
    setNumOfSortFields(7);
}
```
- Use getSortDetail() in application QueryCommandForm as usual.
- Add getter/setter methods for additional sort fields and sort field orders in application QueryCommandForm as below

```
// Getter/Setter for additional sort fields
public String getSortField7() {
    return getSortField(6);
}
public void setSortField7(String sortField) {
    setSortField(6, sortField);
}

// Getter/Setter for additional sort field orders
public String getSortFieldOrder7(){
    return getSortFieldOrder(6);
}
public void setSortFieldOrder7(String sortFieldOrder) {
    setSortFieldOrder(6, sortFieldOrder);
}
```
- Override getNumOfSortFields() in application QueryCriteriaBuilder class as below

```
@Override
public int getNumOfSortFields() {
    return 7; // this value must be same as specified in QueryCommandForm
}
```

3.6 Label

Every control in an application is associated with a label. Trade Type is a label for trade type select box in Trade Query page as shown below :

```
<form:label path="tradeType">
<spring:message code="trade.label.trade.type" htmlEscape="false" />
</form:label>
```

3.7 Label indicating mandatory fields

Certain fields are mandatory. In order to indicate such fields their associated labels are highlighted using the class "required". First Name is a label for the mandatory field in Employee Entry page as shown below :

```
<form:label path="commandForm.employeeDTO.firstName" class="required">
<spring:message code="inf.employee.label.firstname" htmlEscape="false" />
</form:label>
```

3.8 Submit button

Submit button allows user to submit a form for user confirmation. For example, in Trade Query flow, this button is used to submit the query form.

```
<div class="btnWrapStyle submitBtn fchFormBtn">
<input type="submit" value="${formsubmit_label}" class="inputBtnStyle" />
</div>
```

3.9 Reset button

Reset button allows user to reset a form. For example, in Trade Query flow, this button is used to reset the query form.

```
<div class="btnWrapStyle resetBtn fchFormBtn">
<input id="qry-form-reset" type="reset" value="${formreset_label}" class="inputBtnStyle" />
</div>
```

3.10 Previous button

Previous button allows user to navigate to the previous tab of a wizard based entry/amend. For example, in Trade Entry flow, this button is used to navigate to the "Trade General" tab.

```
<div class="btnWrapStyle wizPrevious fchFormBtn">
<input type="button" class="inputBtnStyle" value="${formprevious_label}" style="display: none;" disabled="disabled" />
</div>
```

3.11 Next button

Next button allows user to navigate to the next tab of a wizard based entry/amend. For example, in Trade Entry flow, this button is used to navigate to the "Trade Details" tab.

```
<div class="btnWrapStyle wizNext fchFormBtn">
<input id="nextButtonID" type="button" class="inputBtnStyle" value="${formnext_label}" style="display: none;" disabled="disabled" />
</div>
```

3.12 Back button

Back button allows user to navigate back to the previous mode the form was submitted. For example, in Trade Entry flow, this button is used to navigate from the user confirmation page to the tab from where the form was submitted.

```
<div class="btnWrapStyle wizBack fchFormBtn">
<input type="button" class="inputBtnStyle" value="${formback_label}" style="display: none;" disabled="disabled" />
</div>
```

3.13 Confirm button

Confirm button allows user to navigate to the system confirmation page. For example, in Trade Entry flow, this button is used to navigate

from the trade entry user confirmation page to the trade entry system confirmation page.

```
<div class="btnWrapStyle wizConfirm submitBtn fchFormBtn">
<input type="submit" class="inputBtnStyle" value="${formconfirm_label}" style="display: none;" disabled="disabled" />
</div>
```

3.14 Ok button

Ok button allows user to navigate back to the fresh entry page or to the query result page. For example, in Trade Entry flow, this button is used to navigate to the fresh trade entry page.

```
<div class="btnWrapStyle wizOk submitBtn fchFormBtn">
<input type="submit" class="inputBtnStyle" value="${formok_label}" style="display: none;" disabled="disabled" />
</div>
```

3.15 Add button

Add button allows user to add records to the grid component. For example, in Employee Entry flow, this button is used to add employee application roles to the grid component.

```
<div class="btnWrapStyle submitBtn gridAddBtn">
<span class="addBranchRole" style="display: block"><input id="addBranchRoleBtn" type="button" value="${formadd_label}" class="inputBtnStyle" /></span>
</div>
```

4. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	03-02-2015	Initial version	Mithu Tokder

3.4.6.2.26.4 Grid Component

1. Scope

In finch application for grid component, SlickGrid plugin has been used. It has the following features :

- Column resize/reorder/show/hide
- Column autosizing & force-fit
- Customize using pluggable cell formatters
- Sorting

This document outlines the steps required to work with finch grid component.

Grid Component Column ResizeColumn ReorderColumn Show/HideColumn Force FitCell Formatters Cell Sorting

TRDRS:Trade Query Result [03-May-2014]

Trade Date	Reference No	Trade ...	Trade Type	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...
04-Dec-2012	T20140103004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-2012	T20140103004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-2012	T20140103004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001555	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001557	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001578	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST

1. Customer query result displayed in grid component with the auto sized column widths.

TRDRS:Trade Query Result [03-May-2014]

Trade Date	Reference No	Trade Currency	Trade Type	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...
01-Dec-2012	T20140303004734	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-2012	T20140303004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-2012	T20140303004733	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001555	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001557	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001578	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-2012	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST

1. Grid component with columns resized.

TRDRS:Trade Query Result [03-May-2014]

Reference No	Trade Date	Trade Currency	Trade Type	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...
T20140303004734	01-Dec-2012	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
T20140303004732	01-Dec-2012	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
T20140303004733	01-Dec-2012	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
T20140103001555	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001556	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001557	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001558	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001578	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001577	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
T20140103001581	11-Dec-2012	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST

1. Grid component with columns reordered.

TRDRS:Trade Query Result [03-May-2014]

Trade Date	Trade Currency	Value Date	Account B...	Inventory ...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...	
01-Dec-2012	1	USD	03-Dec-2012	-1	121212007	S	TEST007	CANCEL	ST
01-Dec-2012	USD	03-Dec-2012	-1	121212007	S	TEST007	CANCEL	ST	
01-Dec-2012	USD	03-Dec-2012	-1	121212007	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	
11-Dec-2012	USD	03-Dec-2012	0	121212005	S	TEST007	CANCEL	ST	

1. Grid component with the "Customer Code Type" column hidden by de-selecting from the column picker.
2. Column picker showing the unchecked "Customer Code Type" checkbox.

TRDRS:Trade Query Result [03-May-2014]											Query	Result	
Page	1	of 268	Records:	2679									
Trade Date	Reference No	Trade Currency	Trade Type	Value Date	Account Balance T...	Inventory Account...	Settlement Curre...	Buy/Sell Flag	External Referenc...	Trade Status	Instrument Type		
01-Dec-2012	T20140303004734	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST		
01-Dec-2012	T20140303004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST		
01-Dec-2012	T20140303004733	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001555	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001557	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001559	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001576	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		
11-Dec-2012	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST		

1. Grid component showing force fit columns. This option can be chosen from the column picker.

TRDRS:Trade Query Result [03-May-2014]											
Page	1	of 1268	Records: 2675								
Trade Date	Reference No	Trade Type	Value Date	Account B	Inventory	Settlement	Buy/Sell Flg	External Ref	Trade Status	Instrumen...	
01-Dec-...	T201403004734	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-...	T201403004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec-...	T201403004733	USD	EQ	03-Dec-2012	1	121212007	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001555	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001557	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001576	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec-...	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST

1. Grid component with the "Customer Code" column formatted to display the data as a link using custom formatters.

TRDRS:Trade Query Result [03-May-2014]												Query	Result	
Page	1	of 268	Records:	2879										
Trade Date	Reference No +	Trade ...	Trade Typ...	Value Date	Account B...	Inventory ...	Settlemen...	Buy/Sell Fl...	External R...	Trade Statu...	Instrumen...			
11-Dec-2012	T20140103001555	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001557	USD	EQ	03-Dec-2012	0	121212006	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001558	USD	EQ	03-Dec-2012	0	121212006	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001576	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST			
11-Dec-2012	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST			
01-Dec-2012	T20140303004732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST			
01-Dec-2012	T20140303004733	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST			
01-Dec-2012	T20140303004734	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST			

1. Grid component with sorting enabled for "Customer Code" column.

2. Prerequisites

MUST SEE

[UI Component](#) for the details on how to use the UI components in finch.

CSS for the CSS related details.

[UI Core - Grid](#) and [grid related section](#) for complete implementation details of all the features of grid component.

3. Sequence of Steps

There are three definitions that we need for rendering the grid

- Column Definition
 - Data
 - Settings

3.1 Column Definition

This definition helps the grid to find out:

- How many columns are there.
 - Name/Header of the Columns.
 - If the columns are sortable or not.

- Width of each column.
- Special formatting in case any.
- If a particular column represents numbers, which helps in the alignment and sorting.
- Column - DataSet mapping

In finch application column set is defined as an array of objects, where each object represents a single column.

For reference, the column definition of Trade Query Result is :

```
var grid_result_columns = [
  {name:<spring:message code="trade.label.trade.date" htmlEscape="false" />, field:"tradeDateStr", id:"tradeDateStr", width:64, sortable:true, cssClass:'finch-grid-date'},
  {name:<spring:message code="trade.label.reference.no" htmlEscape="false" />, field:"tradeReferenceNo", id:"tradeReferenceNo", width:147, sortable:true, formatter:Slick.Formatters.TradeDetailViewFormater, isForAuth:false},
  {name:<spring:message code="trade.label.trdae.ccy" htmlEscape="false" />, field:"tradeCcy", id:"tradeCcy", width:57, sortable:true},
  {name:<spring:message code="trade.label.trade.type" htmlEscape="false" />, field:"tradeType", id:"tradeType", width:78, sortable:true},
  {name:<spring:message code="trade.label.value.date" htmlEscape="false" />, field:"valueDateStr", id:"valueDateStr", width:78, sortable:true, cssClass:'finch-grid-date'},
  {name:<spring:message code="trade.label.accountbal.type" htmlEscape="false" />, field:"accountBalanceType", id:"accountBalanceType", width:78, sortable:true},
  {name:<spring:message code="trade.label.inventory.accountno" htmlEscape="false" />, field:"inventoryAccountNo", id:"inventoryAccountNo", width:78, sortable:true},
  {name:<spring:message code="trade.label.settlement.ccy" htmlEscape="false" />, field:"settlementCcy", id:"settlementCcy", width:78, sortable:true},
  {name:<spring:message code="trade.label.buysell.flag" htmlEscape="false" />, field:"buySellFlag", id:"buySellflag", width:78, sortable:true},
  {name:<spring:message code="trade.label.external.referenceno" htmlEscape="false" />, field:"externalReferenceNo", id:"externalReferenceNo", width:78, sortable:true},
  {name:<spring:message code="trade.label.trade.status" htmlEscape="false" />, field:"status", id:"status", width:78, sortable:true},
  {name:<spring:message code="trade.label.instrument.type" htmlEscape="false" />, field:"instrumentType", id:"instrumentType", width:78, sortable:true}
];
```

3.1.1 Defining a Simple Column

```
{
  //Text displayed as column Header
  name:<spring:message code="trade.label.trade.date" htmlEscape="false"/>,
  //Used for dataset-column mapping
  field:"tradeDateStr",
  //Unique id for every column
  id:"tradeDateStr",
  //Width of the column
  width:64,
  //If the column is sortable or not when clicking on column header.
  sortable:true
}
```

3.1.2 Defining a Column which needs some formatting

This is a valid requirement when we want some of our columns to display the data in a different way, requirement may be to display the data in different color or the data to be a hyperlink and not a normal text.

For eg. In the Trade Query Result Screen we need our Reference Number Column to display a hyperlink, for introducing some

functionality, to achieve this we need to follow the following steps :

Step1: Define the formatter in "sample-trd/src/main/web/scripts/sample-trd-formatters.js"

First add an entry in the Formatters object, which is nothing but a unique name to function mapping.

```
"Slick": {  
    "Formatters": {  
        //Unique Name : Function Name  
        "TreeViewFormatter": TreeViewFormatter  
    }  
}
```

and then define the function itself. This function is called for every piece of data for the column which has mentioned this as its formatter by using the unique name above (more details below), and then displays whatever this function returns.

```
function TradeDetailViewFormater(row, cell, value, columnDef, dataContext) {  
    if (value == null || value === "") {  
        return "";  
    }  
    var urlPath = '/trd/query/';  
    if($('form').attr('action').indexOf('/trd/amendcancel') > -1) {  
        var urlPath = '/trd/amendcancel/';  
    }  
    var markup = ""  
        + "<span class='detail-view-hyperlink' "  
        + "view='tradeDetailView' "  
        + "referenceno=" + value + " "  
        + "tradepk=" + dataContext.tradePk + """  
        + "dialogIdentifier='tradeDetail'" + dataContext.tradePk + """  
        + "duplicate = " + FINCH.detailDialogTypeBehaviour+ """  
        + "dialogTitle='[" + value + "] " + SAMPLE$TRD$i18n.formatter.trade_Details + """  
        + "href=" + urlPath + "details/" + dataContext.tradePk + "/" + value + "?diffEnableFlag=" +  
columnDef.isForAuth +"">"  
        + value  
        + "</span>";  
    return markup;  
}
```

Step2: After the formatter is ready, mention it in the column to be formatted.

```
{  
    name:<spring:message code="trade.label.reference.no" htmlEscape="false"/>,  
    field:"referenceNo",  
    id:"referenceNo",  
    width: 147,  
    sortable:true,  
    //Below TradeDetailViewFormater is the unique name that we mentioned while defining our formatter  
    formatter:Slick.Formatters.TradeDetailViewFormater  
}
```

3.1.3 Defining a Column which represents a date column

Date column has few special requirements, two of them being making them left aligned, and also to sort them not as text or numbers. To achieve this, only one extra param needs to be added to the column definition i.e.

```
cssClass:'finch-grid-date'
```

So the column definition should be like:

```
{
  name:<spring:message code="trade.label.trade.date" htmlEscape="false" />,
  field:"tradeDateStr",
  id:"tradeDateStr",
  width:64,
  sortable:true,
  //Specifying that this column is a date column.
  cssClass:'finch-grid-date'
}
```

3.2. Data Preparation

finch being an hybrid application, data has to be prepared in two ways.

- First way is when any result page is opened, the first thing it returns is a jspx, with data being a part of it.
- Second way is for all further communication with the server for that page, which is done via Ajax.

Case1: Initially when we want the data to be returned within the jspx, we build an array of objects in javascript in the following manner

```
//Initial Data Set.
var grid_result_data = [];
var rec = {};
<c:forEach items="${value}" var="dl">
  rec = {};
  rec.id = //Should refer to some unique id i.e data that is different for every row.
  rec.tradePk= "<c:out value='${dl.tradePk}' />";
  rec.tradeDateStr = "<c:out value='${dl.tradeDateStr}' />";
  ...
  grid_result_data.push(rec);
</c:forEach>
```

Case2: Mentioning the url for making Ajax call to fetch data sets in the settings definition (more details in Section 3.3)

3.3. Settings

This is the section where the grid is modified, based on the requirement. Most of the grid attributes have default values.

```

var grid_result_settings = {
    enableToolbar:true,
    consolidateActionFlag:false,
    buttons:{
        print:true, //Show/Hide print button in the toolbar.
        xls:true, //Show/Hide Excel Report button in the toolbar.
        pdf:true, //Show/Hide Pdf Report button in the toolbar.
        csv:true, //Show/Hide Csv Report button in the toolbar.
        columnPicker:true, //Show/Hide ColumnPicker button in the toolbar.
        save:true, //Show/Hide Save Button in the toolbar.
        reset:true //Show/Hide Reset Button in the toolbar.
    },
    //Information related to Paging
    pagingInfo:{
        isNext : isNext, //Specifies if there is any next page or not
        url: urlPath + 'count.json',
        pageNo: <c:out value="${pageNo}" />, //Specifies the page number
        isPrevious: <c:out value="${isPrevious}" /> //Specifies if there is any previous page or not
    },
    urls:{
        nextPage : '/trd/query/result.json?fetch=next',
        prevPage : '/trd/query/result.json?fetch=previous',
        pdfReport: '/trd/query/report.json?outputType=pdf',
        xlsReport: '/trd/query/report.json?outputType=xlsx',
        csvReport: urlPath + 'report/trade.csv?filetype=csv'
    }
};

```

3.3.1 Available Properties

1. enableToolbar (boolean) : If true the Grid toolbar is displayed, is hidden when the value is set to false.
2. consolidateActionFlag (boolean): Consolidation is enabled when set to true.
3. buttons (object): This contains a list of all buttons that needs to be displayed/hidden. See above code for reference, buttons which are set to true are displayed in the toolbar, in case set to false the corresponding button is hidden.
4. pagingInfo (object): This object gives information related to paging, more details on this object is [here](#).
5. urls (object): Contains the list of all the urls, which is required by the grid to communicate with the server for generating reports or retrieving data sets.

3.4. Rendering

The use of the definitions and constructing the grid object can be seen in "finch-web-infra/src/main/web/scripts/inf/finch-form.js".

The code snippet below shows how to initialize the non-editable grid component with the required parameters:

```
$('#queryResultForm .finch-grid').finchgrid(grid_result_data, grid_result_columns, grid_result_settings)
```

For editable grid component editMode option is available which can have 'none', 'both', 'edit' or 'delete' values. Edit and delete actions are implemented using the editCallback and deleteCallback options respectively.

```

var userAppIRoleGridConf = {
    editMode      : 'both',
    editCallback   : editBranchRoleHandler,
    deleteCallback : removeUserAppIRoleMappingHandler
};

```

The code snippet below shows how to initialize the editable grid component with the required parameters:

```

$userAppIRoleGrid = $('#userAppIRoleGrid', $employeeEntry$context);
var $userAppIRoleGridObject = $userAppIRoleGrid.fincheditablegrid(userAppIRoleData, userAppIRoleColumns,
userAppIRoleGridConf);

```

For editable grid2 component provides all the editable grid option as well as some extra options.

```

var taxFeeGridConf = {
    editMode      : 'both', // For editing. Possible values 'none', 'both', 'edit', 'reset'
    customResetAction : resetTaxFeeHandler, // For resetting the grid
    customDeleteAction : deleteTaxFeeHandler, // For deleting
    preAddHandler   : preAddHandler, // Handler before adding a row
    postAddHandler  : postAddHandler, // Handler after adding a row
    newRowPosition  : 'current', // Position of the new record. Possible values are 'start','end','current'
    validateBeforeAdd : true, // Validate before adding a new row
    checkEmptyColumns : true, // Check for empty column
    customGridResetAction : gridResetAction // Customizable grid reset action
};

```

The code snippet below shows how to initialize the editable grid2 component with the required parameters:

```

$taxFeeGrid = $('#taxFeeGrid', $tradeEntry$context);
var $taxFeeGridObject = $taxFeeGrid.fincheditablegrid2(taxFeeData, taxFeeGridColumns, taxFeeGridConf);

```

4. See Also

For more details on slickgrid plugin and related examples refer to <https://github.com/mleibman/SlickGrid/wiki>

5. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	26-02-2015	Initial version	Sourav Mahato

3.4.6.2.26.4.1 Page and Record Count component

1. Scope

This document outlines the steps required to show the page and record count in query result pages using grid component.

Within the grid component, the page and record count is displayed as shown below :

Total no. of pages TRDRS:Trade Query Result [03-May-2014]											
Page	1	2	3	4	5	6	7	8	9	Total no. of records	
Trade D...	Reference No	Trade ...	Trade Typ...	Value Date	Account B...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...
01-Dec...	T20140303094734	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec...	T20140303094732	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
01-Dec...	T20140303094733	USD	EQ	03-Dec-2012	-1	121212007	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001556	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001557	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001558	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001576	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001577	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST
11-Dec...	T20140103001581	USD	EQ	03-Dec-2012	0	121212005	USD	S	TEST007	CANCEL	ST

Fig 1 : Trade query result showing the page and record count

2. Prerequisites

MUST SEE

[Grid Component](#) document for complete details on how to use the grid component.

3. Sequence of Steps

Please note that this explanation uses Trade Query Result screen as an example.

3.1 Grid Result Settings

To display the page and record count, we need to specify an extra url param in the pagingInfo object of the grid result settings:

For eg : For the Trade Query Page, we specify the url as 'trd/query/count.json'.

```
var grid_result_settings = {
    enableToolbar:true,
    consolidateActionFlag:false,
    buttons:{
        print:true,
        xls:true,
        pdf:true,
        columnPicker:true,
        save:true
    },
    pagingInfo:{
        isNext : isNext,
        //Specifying the url: Begin
        url: 'trd/query/count.json'
        //Specifying the url: End
    },
    urls:{
        nextPage : '/trd/query/result.json?fetch=next',
        prevPage : '/trd/query/result.json?fetch=previous',
        pdfReport: '/trd/query/report.json?outputType=pdf',
        xlsReport: '/trd/query/report.json?outputType=xls'
    }
};
```

This url when used in an ajax request returns data in the following format:

```

POST http://localhost:8080/sample-web//trd/query/count.json?commandFormId=1424940032579 200 OK 20ms
Params Headers Post Response JSON Cookies
Sort by key
finchModelMap
  modelMap
    success
    value
    errorFields
    count
    totalPageNo
    successFlag
    totalCnt
Object { modelMap:{...}, successFlag=true, totalCnt=100 }
Object { success=true, value=[0], errorFields="errorFields", more... }
true
[ ]
"errorFields"
2679
268
true
100

```

Rest will be handled by the finch grid itself.

4. See Also

For more details on slickgrid plugin and related examples refer to <https://github.com/mleibman/SlickGrid/wiki>

5. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	26-02-2015	Initial version	Sourav Mahato

3.4.6.2.26.5 Popup Component

finch popups are composite components consisting of text inputs, select boxes, action buttons and grid components.

These popups are used in the entry and query pages to retrieve data required for creating or querying for an entity like Trade, Account etc.

Such types of data include

- [Account No](#)
- [Inventory Account No](#)
- [Security Code](#)

finch popups are used for querying and retrieving these above mentioned types of data .

Developer needs to follow the finch standard guidelines to create a new popup and use an existing popup.

3.4.6.2.26.5.1 Add a new pop-up

1. Scope

This document gives a brief description of the working principal of popup framework used in finch application. It outlines the steps needed to create a new popup.

2. Prerequisites

MUST SEE

Refer to [Popup Component](#) for an introduction to popup.

Refer to [View templates - Screen Composition](#) section for details on popup templates.

Refer to [Rich UI Framework - Popup framework](#) section for further details.

Refer to [Query/Query Result](#) - For detail about query framework.

Refer to [Access Control](#) document for implementation details of authorization for accessing a resource.

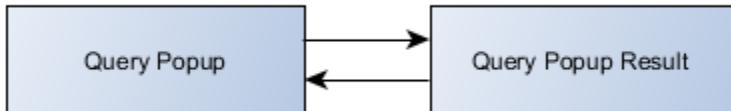
2.1 Templating

In finch, a template is used for all screens so that the individual screen developer will provide only screen specific implementation. Rest of the functionality common to all screens is handled by the template.

See the [View templates](#) document for more details on tile definitions and inheritance pattern.

2.2 Screen Flow

The flow for query page is shown below :



2.3 UI components

The popup query and result page consists of the following UI components:

- Form & Form elements (mandatory)
- Action buttons (mandatory)
- Grid (mandatory)
- Popup (optional)

The classes and ids for the various UI components, used in the existing popups, should be retained to get the basic layout, look & feel and functionality of finch popup query forms and result grids.

See the [UI Component](#) document for more details about the individual UI components.

2.4 Request-Response

In finch, a common FINCH\$Handler\$function is used for handling all request-response type operations.

See the [Request Response](#) document for more details on this method.

3. Sequence of Steps

Please note that we have used Trade Account Popup Query as a reference throughout this document, for all the explanations.

3.1 Create the Model

For query operation, no database changes are required. Though it has to be decided what are the columns we are going to show in QueryResult screen.

Create Command Form

See the [Query/Query Result - Create Command Form](#) section for complete details.

Following command form is required for account query pop up

AccountPopUpQueryCommandForm.java

[Expand](#)

```
public class AccountPopUpQueryCommandForm extends QueryCommandForm{
```

[source](#)

```
    private String accountNo= StringUtils.EMPTY;  
    .  
    public String getAccountNo() {  
        return accountNo;  
    }  
    public void setAccountNo(String accountNo) {  
        this.accountNo = accountNo  
    }  
    .  
    @Override  
    public void reset() {  
        accountNo= null;  
    }  
}
```

3.2 Create the view

3.2.1 Usage of Tiles

In order to create the layout for any query popup, developer needs to provide a tile definition of its corresponding view in views.xml.

For account query popup, user needs two views :

- Account query popup
- Account query popup result

The view corresponding to query popup say "accountPopUpQueryCriteria.jspx" needs to be defined in views.xml as shown below :

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/views.xml

[Expand](#)

```
<definition name="accountPopUpQueryCriteria" extends="popQueryCriteria">  
    <put-attribute name="content">  
        <definition extends="popCriteria">  
            <put-attribute name="popUpCriteria" value="/WEB-INF/views/ref/accountPopUpQueryCriteria.jspx"/>  
        </definition>  
    </put-attribute>  
</definition>
```

The new view "accountPopUpQueryCriteria.jspx" needs to extend "popQueryCriteria". By doing so, the new view inherits all the common functionality and markup from "popQueryLayout.jspx".

Trade query page inherits the following features/elements from its popQueryCriteria template :

- Form tag and all its attributes

The view corresponding to trade query result say "accountPopUpQueryResult.jspx" needs to be defined in views.xml as shown below :

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/views.xml

› Expand

```
<definition name="accountPopUpQueryResult" extends="popResult">
    <put-attribute name="content">
        <definition extends="popQueryResult">
            <put-attribute name="popResultGrid" value="/WEB-INF/views/ref/accountPopUpQueryResult.jspx"/>
        </definition>
    </put-attribute>
</definition>
```

source

See the [View templates - Screen Composition](#) section to get further details.

3.2.2 Create new query popup

After providing the tile definition, the developer needs to create the new query page "accountPopUpQueryCriteria.jspx".

The "accountPopUpQueryCriteria.jspx" needs to be placed in a specific folder. Its location will depend on the type of data it queries for.

In general, each entity in finch application is maintained in a particular finch core module depending on its type. Whatever be the type of data, all jsp files are kept within views folder. Within each views folder, a folder with the same name as the <module-name> is created.

So the general path for the new jsp should be :

finch/sample-app/sample-<module-name>/src/main/web/WEB-INF/views/<module-name>/new.jspx

For example :

"accountPopUpQueryCriteria.jspx" will query for account related information. Account related information should always be placed in the "sample-ref" module.

Under the views folder within "sample-ref" create a "ref" folder. So the path for "accountPopUpQueryCriteria.jspx" will be :

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/views.xml

In the "accountPopUpQueryCriteria.jspx" developer needs to create the form elements using specific classes and ids.

See the [Popup Form](#) section to get the details about the markup needed to create a query popup.

Now that the "accountPopUpQueryCriteria.jspx" markup is ready, developer needs to map the form attributes and element values to a data model. In finch application commandForms have been used for sending the form values from client to the server.

The variable names used in the commandForms should be identical to the names used in the form elements and attributes.

For example :

For counter party detail type map, the variable defined in AccountPopUpQueryCommandForm class is "counterPartyDetailTypeMap".

In "accountPopUpQueryCriteria.jspx", the conuter party detail type select element options should be mapped to this variable using "\${commandForm.counterPartyDetailTypeMap }" as shown below :

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/accountPopUpQueryCriteria.jsp

[Expand](#)

```
<div class="popFormItem">
<form:label path="counterPartyDetailType">
<spring:message code="ref.label.counterparty.type" htmlEscape="false"/>
</form:label>
<span>
<form:select path="counterPartyDetailType" class="dropdowninput counterPartyCodeType" >
<form:options items="${commandForm.counterPartyDetailTypeMap}"/>
</form:select>
</span>
<span>
<form:input value="" path="counterPartyCode" class="textBox smallInput marginLeft" disabled="true"/>
</span>
<div class="clear">
<spring:message text="" htmlEscape="false"/>
</div>
</div>
```

[source](#)

See the section on Create Command Form and Create the Controller for further details.

3.2.3 Initialize new query popup

The popup query form allows users to :

- Provide inputs
For example, for Account Query Popup, user can input the below query parameters:
 - Account No
 - Account Name(Short)
 - Counter Party Code
- Submit the form
- Reset the form

The hooks for all the above mentioned actions are defined in finch-popquery-form.js .The list of hooks available are :

- submitHandler - It submits the query form, initializes the result grid with popup specific initialization parameters and displays the query result.
- resetHandler - It resets the form element values to the original state.

For example,

In the Account Query Popup, when user clicks on submit then the result grid is populated with account codes.

In the individual query popup, developer needs to include the required js files necessary for the form elements included within the popup.

For example, in Account Query Popup, counter party code control is required. So the corresponding finch-counterparty-code.js file needs to be included as shown below :

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/accountPopUpQueryCriteria.jsp

[Expand](#)

```
<jsp:attribute name="src">
    <c:url value="/scripts/ref/finch-counterparty-code.js"/>
</jsp:attribute>
```

[source](#)

3.2.4 Initialize new query popup result

The query popup result allows user to :

- View the retrieved results and select one record from it
- Requery

The hook for selecting a record is defined in finch-query-result.js :

- select-pop-query handler - It selects a record from the result grid, populates the value in the target text input in the main page and

then closes the popup.

The hook for requery is defined in finch-popquery-form.js :

- requeryHandler - It cleans up the result grid and navigates user back to the query popup with the previous query parameters retained.

In the individual query popup result, developer needs to define the result grid initialization parameters.

For example, in accountPopUpQueryResult.jspx the result grid initialization parameters are defined as follows :

[Expand](#)

finch/sample-app/sample-ref/src/main/web/WEB-INF/views/ref/accountPopUpQueryResult.jspx

```
var grid_result_data = [];
var grid_result_columns = [
  {name:<spring:message code="ref.label.account.no"
htmlEscape="false"/>,targetColumn:true,field:"accountNo",id:"accountNo", width:100,sortable:true},
  {name:<spring:message code="ref.label.account.nameshort"
htmlEscape="false"/>,field:"accountName",id:"accountName", width:160,sortable:true},
  {name:<spring:message code="ref.label.account.customer.name"
htmlEscape="false"/>,field:"customerName",id:"customerName", width:100,sortable:true},
  {name:<spring:message code="ref.account.label.acstatus" htmlEscape="false"/>, field:"status",id:"status",
width:100,sortable:true,formatter:Slick.Formatters.CancelRecordFormatter}
];
var grid_result_settings = {
  enableToolbar:true,
  consolidateActionFlag:true,
  consolidateAttribute:{type:'check'},
  isPopUpQuery:true,
  forceFitColumns:true,
  pagingInfo:{
    isNext : isNext
  },
  urls:{
    nextPage : '/ref/account/popup/query/result.json?fetch=next',
    prevPage : '/ref/account/popup/query/result.json?fetch=previous'
  }
};
var row_id = 0;
var rec = {};
<c:forEach items="${value}" var="dl">
  rec = {};
  row_id+=1;
  rec.id = "finch_" + row_id;
  rec.accountNo = "<c:out value='${dl.accountNo}'/>";
  rec.accountName = "<c:out value='${dl.accountName}'/>";
  rec.customerName = "<c:out value='${dl.customerName}'/>";
  rec.status = "<c:out value='${dl.status}'/>";
  grid_result_data.push(rec);
</c:forEach>
</jsp:body>
```

[source](#)

targetColumn:true, for an item in grid_result_columns array, signifies that value of this column would be selected when user clicks on select button.

Using the grid initialization parameters mentioned above, the popup result grid is initialized in the submitHandler method defined in finch-popquery-form.js.

Reference

See [Grid Component](#) for details on how to use finch grid component.

3.2.5 Integrate new query popup

On the trade query page where the account query popup needs to be integrated the following code is required:

```
finch/sample-app/sample-trd/src/main/web/WEB-INF/views/trd/tradeQueryCriteria.jspx  
Expand  
<div class="formItem">  
    <form:label path="accountNo"><spring:message code="trade.label.account.no"  
    htmlEscape="false"/></form:label>  
    <span><form:input value="" path="accountNo" class="textBox accountNo"/></span>  
    <span class="remove" title="Remove this field" style="display: none;"><spring:message text=""  
    htmlEscape="false"/></span>  
    <div class="popupBtn">  
        <input type="button" class="popupBtnlco" tgt="accountNo" popType="cpAccount"  
        actTypeContext="T|B" actCPTYPEContext="CLIENT|INTERNAL|BROKER" actStatusContext="OPEN" value="" />  
    </div>  
    <div class="clear"><spring:message text="" htmlEscape="false"/></div>  
</div>
```

source

Reference

See [Use a pop-up](#) document for further details on how to integrate different type of popups in finch.

The "openPopUpForm" function in finch-popquery-form.js opens the popup on button click. It performs the following tasks:

1. Determines popup object from popType.
2. Determines target field using tgt and the selected data from popup page to be set in this target field.
3. If required, populates the dependent parameters from the given custom properties of popup button.
4. Sends an ajax request to load the requested popup in a dialog.
5. "setTopMost" function sets dialog on top of all.

For example, for account query popup the below parameters are required:

- tgt: "accountNo"
- popType: "cpAccount"
- actTypeContext="T|B"
- actCPTYPEContext="CLIENT|INTERNAL|BROKER"
- actStatusContext="OPEN"

3.3 Create the Controller

3.3.1 Follow Inheritance rule

See the [Query/Query Result - 4.3.1 Follow Inheritance rule](#) section for complete details

AccountPopUpQueryController.java

› [Expand](#)

[source](#)

```
@Controller

public class AccountPopUpQueryController extends
AbstractQueryController<AccountPopUpQueryCommandForm>{
    /**
     * The <code>Log</code> instance for this class.
     */
    private static final Log log = LogFactory.getLog(AccountPopUpQueryController.class);

    .
    .
}
```

3.3.2 Provide request URL mapping

See the [Query/Query Result - 4.3.2 Provide request URL mapping](#) section for complete details

AccountPopUpQueryController.java

› [Expand](#)

[source](#)

```
@Controller
@RequestMapping(value = "/ref/account/popup/query")
@SessionAttributes({"commandForm"})
public class AccountPopUpQueryController extends
AbstractQueryController<AccountPopUpQueryCommandForm>{
    .
    .
}
```

3.3.3 Define screen specific initialization logic

See the [Query/Query Result - 4.3.3 Define screen specific initialization logic](#) section for complete details

When user clicks a pop up icon (Account query pop up), the method init() of the base class is being called using the URL Pattern. Implicitly, the dolnit() method is being called here.

AbstractQueryController.java

› [Expand](#)

[source](#)

```
public abstract class AbstractQueryController<C extends QueryCommandForm> {
    .
    .
    @RequestMapping(value="/init",method = {RequestMethod.GET,RequestMethod.POST})
    public String init(ModelMap modelMap,C form) {
        .
        .
        dolnit(form,finchModelMap);
        .
        .
    }
    private void dolnit(C form, FinchModelMap finchModelMap) throws FinchException{
        .
        .
    }
    .
    .
}
```

The following is the screen initialization of Account pop up query:

```
AccountPopUpQueryController.java
```

Expand

source

```
@Controller  
@RequestMapping(value = "/ref/account/popup/query")  
@SessionAttributes({"commandForm"})  
public class AccountPopUpQueryController extends  
AbstractQueryController<AccountPopUpQueryCommandForm>{  
  
    . . .  
  
    @Override  
    protected void doInit(AccountPopUpQueryCommandForm form,  
        FinchModelMap FinchModelMap) throws FinchException {  
        HttpServletRequest request = ((ServletRequestAttributes) RequestContextHolder  
            .getRequestAttributes()).getRequest();  
        try {  
            initQuery(FinchModelMap, form);  
            if (request != null) {  
                form.setBaseUrl(StringUtils.removeEnd(request.getServletPath(),  
                    "/init"));  
            }  
        } catch (Exception e) {  
            log.error("Error in account pop-up init", e);  
            FinchModelMap.addError(e);  
        }  
    }  
  
    private void initQuery(FinchModelMap modelMap, AccountPopUpQueryCommandForm cmdForm){  
  
        commandForm.setCounterPartyDetailTypeMap(context.getCallerIdentity().getUserConstraint().get("COUNTER_  
PARTY_TYPE"));  
        prepareContextsFromDependents(commandForm);  
        prepareUIForContext(commandForm);  
  
        . . .  
    }  
}
```

4. See Also

For more details refer to the following :

<https://tiles.apache.org/> on using Apache Tiles

<http://api.jquery.com/> on jQuery

<https://github.com/mleibman/SlickGrid/wiki> on slickgrid plugin

<http://growl.info/> on notification system

5. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	27-02-2015	Initial version	Sourav Mahato

3.4.6.2.26.5.2 Use a pop-up

1. Scope

This document outlines the different attributes/properties required to use different types of popups in finch application.

2. Prerequisites

Refer to [Popup Component](#) for an introduction to popup.

Refer to [Rich UI Framework - Popup framework](#) section for further details.

Refer to [Add a new pop-up document](#) for further details on how to create a new popup.

2.1 Popup Component

All finch popups are composite components. There are different kinds of popups available in finch application :

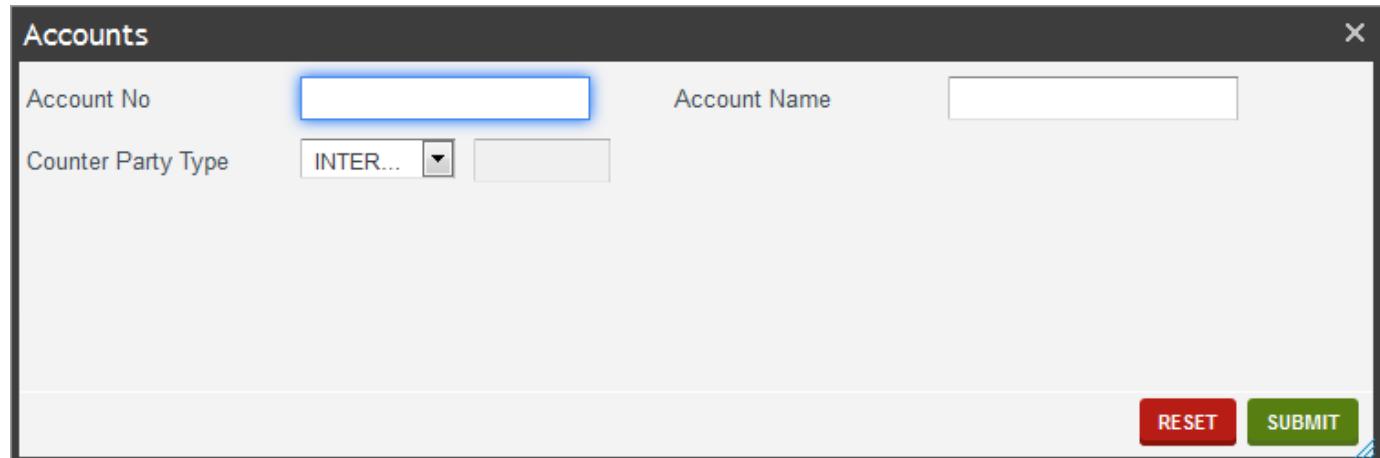
- Account popup
- Inventory Account popup
- Security code popup

For all the popups following properties are mandatory :

- type : button
- class : popupBtnlco
- tgt : Id of the form:input element where the selected data from popup page to be set.
- popType : Popup type (for Account/Inventory Account/Security code etc)
- Any other dependent configuration for specific popup pages

3. Popups

3.1 Account popup



The screenshot shows a modal dialog titled "Accounts". It contains three input fields: "Account No" (with a blue selection bar), "Counter Party Type" (with a dropdown menu showing "INTER..."), and "Account Name". At the bottom right are two buttons: a red "RESET" button and a green "SUBMIT" button.

```
<span><form:input value="" path="accountNo" class="textBox accountNo" /></span>
<div class="popupBtn">
  <input id="accountPopupBtn" type="button" class="popupBtnlco amendReadOnlyPopup" tgt="accountNo"
  popType="query" popFor="cpAccount" actTypeContext="T|B"
  actCPTYPEContext="CLIENT|BROKER|INTERNAL" actStatusContext="OPEN" value="" />
</div>
```

- popType="query" signifies that it is account query popup
- tgt="accountNo", here 'accountNo' is an id of form:input element where the selected data from popup page to be set
- actTypeContext, actCPTYPEContext, actStatusContext are available dependent parameters
- Account popup contains the counter party code control. Based on the actCPTYPEContext parameter passed, the counter party control dropdown options is populated. See [Form Elements Counter party code control](#) section for further details.

3.2 Inventory Account popup

Inventory Accounts

Account No	<input type="text"/>	Account Name	<input type="text"/>
Counter Party Type	<input type="button" value="INTER..."/>	<input type="text"/>	
<input type="button" value="RESET"/> <input type="button" value="SUBMIT"/>			

```
<span><form:input value="" path="inventoryAccountNo" class="textBox invAccountNo" /></span>
<div class="popupBtn">
<input id="invAccountPopupBtn" type="button" class="popupBtnlco" tgt="inventoryAccountNo" popType="query"
popFor="invAccount" invActTypeContext="T|B" invCPTypeContext="INTERNAL" actStatusContext="OPEN" value="" />
</div>
```

- popType="query" signifies that it is a inventory account query popup
- tgt="invetoryAccountNo", here 'invetoryAccountNo' is an id of form:input element where the selected data from popup page to be set.

3.3 Security Code popup

Instruments

Instrument Code	<input type="text"/>	Instrument Type	<input type="text"/>
Instrument Name	<input type="text"/>	Issue Ccy	<input type="text"/> 
<input type="button" value="RESET"/> <input type="button" value="SUBMIT"/>			

```
<span><form:input value="" path="securityCode" class="textBox" /></span>
<div class="popupBtn">
<input id="securityPopupBtn" type="button" class="popupBtnlco" tgt="securityCode" popType="query"
popFor="security" value="" />
</div>
```

- popType="query" signifies that it is a instrument query popup
- tgt="securityCode", here 'securityCode' is an id of form:input element where the selected data from popup page to be set.

4. Document History

Version	Date	Description of Changes	Author/Reviewer
0.1	27-02-2015	Initial version	Sourav Mahato

3.4.6.2.26.6 Rich UI Feature

1. Scope

This document describes different rich UI features available in finch framework.

Currently, in finch framework, following kind of Rich UI controls and features are available:

- Form controls
 - Auto Complete
 - Tree control
 - Multi Select
 - Date Picker control
 - Color Picker
- Personalisation
 - Form personalisation
 - Saved Query
- Popup
 - Popup Component
 - Window Docking
- Grid Component
 - Non-editable grid
 - Editable grid
- Dashboard

2. Prerequisites

MUST SEE

Refer to [Web Component](#) for an overall idea about the finch web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Rich UI Framework](#) for complete details of Rich UI features and components.

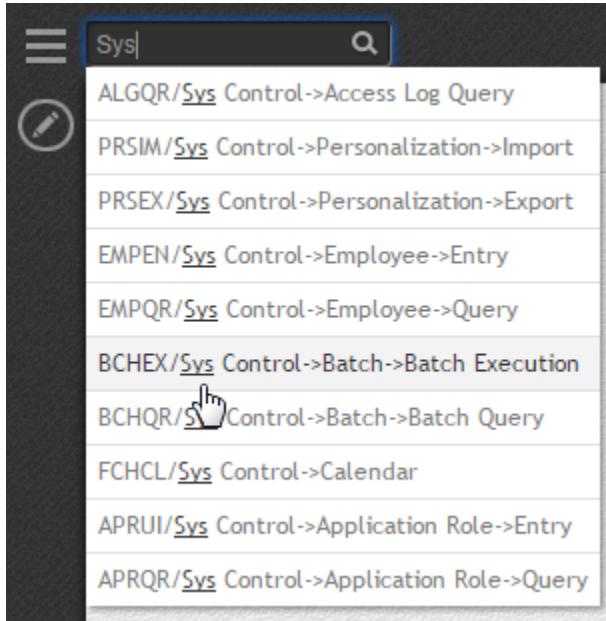
3. Rich UI Controls & Features

3.1 Form Controls

3.1.1 Auto Complete

Autocomplete control suggests the options as we type into the text input. In finch, the [Autocomplete widget of jQuery UI](#) has been used to create an autocomplete control such as the menu search component.

[Screenshot Sequence of Steps](#)



Menu search component operation can be logically divided into three parts:

1. Load the json data and cached it when menu search icon is clicked first time
2. Each time filter the internal cached data based on user input
3. Based on user selection fire the menu url

Step 1

Autocomplete widget uses JSON data as its data source. The data format for menu search component is shown below :

```
{"menuTypeAheadJson": {
  "success":true,
  "data": "[{"CompositeMenuNameDisplay": "Display Name",
    "label": "Search Key",
    "menuUrl": "URL",
    "version": "Version Id",
    "componentId": "Component Id"
  }]
}
```

Here data part contains the menu data. Each menu record has three major component -

1. CompositeMenuNameDisplay - Used for display purpose. It consists of full menu path.
2. Label- Used as search key.
3. Menu Url - When user selects a menu then menu url will be fired.

Example of Menu JSON data:

```
{  
  "menuTypeAheadJson": {  
    "success": true,  
    "data": "[  
      {  
        \"$CompositeMenuNameDisplay$\": \"$TRDEN/Trade->Entry$\",  
        \"$label$\": \"$TRDEN/Trade Entry$\",  
        \"$menuUrl$\": \"$sample-web//trd/entry/init$\",  
        \"$version$\": \"$1$\",  
        \"$componentId$\": \"$TRD$\"  
      },  
      .  
      .  
      .  
    ]"  
  }  
}
```

Step 2

To enable auto-completion in menu search component, the following markup defined in menu.jspx should be inserted as shown below :

```
menu.jspx  
  
<div id="menuSearchArea" class="left">  
  <spring:message code="dbd.label.tsm.menu" htmlEscape="false"  
    var="placeholder" />  
  <div class="left menuSearchInputDiv">  
    <input type="text" id="menuSearchInput" value="${placeholder}"  
      title="${placeholder}" />  
    <span id="menuSearchInputImg"><spring:message text="" htmlEscape="false" /></span>  
  </div>  
</div>
```

Step 3

Then the Autocomplete widget, defined in jquery-ui-1.8.16.custom.js, is initialized in menu.jspx as shown below :

```

menu.jspx

function initAutoComplete(){
    $('#menuSearchInput').autocomplete({
        ...
        ...
    }).data( "autocomplete" )._renderItem = function( ul, item ) {
        var displayLabel =
            doHighlightTreeData($('#menuSearchInput').attr('value'),item.CompositeMenuNameDisplay);
        return $( "<li></li>" ).data( "item.autocomplete", item ).append( "<a>" + displayLabel + "</a>" ).appendTo(
        ul );
    };
}

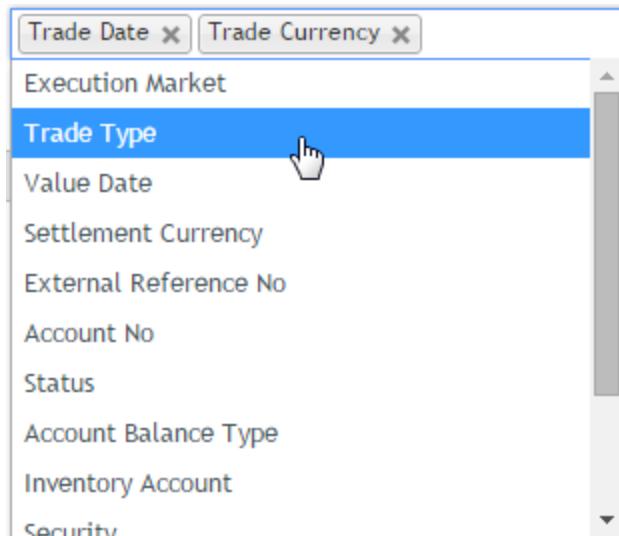
```

Here the widget is bound to the search input box having an id "menuSearchInput". initAutoComplete method, defined in menu.jspx, contains all the auto completion related configuration.

3.1.2 Multi Select

Multi Select control is used to select multiple items from a dropdown. In finch sample application, an example of such a control is the Select Tags control in the second tab of Trade Upload. It can be used to select more than one tag from the dropdown.

[Screenshot Sequence of Steps](#)



finch uses the [jQuery Chosen plugin](#) to implement a multi-select dropdown.

The following code snippet from uploadMappingScreen.jspx shows how the multi-select dropdown widget is attached to a form field:

```

var $fieldNames = $("#field_names", $uploadMapping$context);
...
...
$fieldName.chosen({no_results_text: FINCH.i18n.upload.no_results_found});

```

3.1.3 Date Picker control

Date Picker control is used to input date from user.

Screenshot Sequence of Steps



Date picker control is used in most of the entry and query screens such as Trade Entry and Trade Query screens.

User can input a single date or date-range (from-to).

For Single date the below markup needs to be inserted as present in tradeGeneralEntry.jspx:

tradeGeneralEntry.jspx

```
<form:input id="tradeDate" path="commandForm.tradeDateStr" class="dateinput" />
```

For composite date (from-to) the below markup needs to be inserted as present in tradeQueryCriteria.jspx:

tradeQueryCriteria.jspx

```
<form:input id="tradeDateFrom" value="" path="tradeDateFrom" class="dateinput noOptions"/>
<form:input id="tradeDateTo" value="" path="tradeDateTo" class="dateinput noOptions"/>
```

class="dateinput" signifies that it is a date picker combo.

See Rich UI Framework - Date picker section for details on finch-datepicker plugin.

3.1.4 Color Picker

Color picker is used to choose color from color swatch and provide hexadecimal color codes (Red = #ff0000), if required.

Screenshot Sequence of Steps



Step 1

In finch, the color picker control is implemented in Application Preference screen . To enable color picker the following markup has been defined in infPrefs.jspx :

```
<div class="formItem">
    <form:label path="zebraColorEven">
        <spring:message code="global.pref.label.colour.zebra" htmlEscape="false"/>
    </form:label>
    <span>
        <form:input id="zebraColorEven" path="zebraColorEven"/>
        <form:input id="zebraColorOdd" path="zebraColorOdd"/>
    </span>
    <div class="clear">
        <spring:message text="" htmlEscape="false"/>
    </div>
</div>
```

Step 2

For color picker control the colourpicker plugin, defined in finch-colourpicker.js, is initialized in finch-pref-screen.js as shown below :

```
StabPrefs.find(".zebracolor").colorPicker();
```

See [Rich UI Framework - Color picker](#) section for details on colorPicker plugin.

3.2 Personalisation

3.2.1 Form Personalisation

Form personalisation feature is one of the best Rich UI feature available in finch. Here user can re-order the form items as per his/her desired position. User can also delete the form item and restore the deleted form items.

Screenshots Sequence of Steps

This screenshot shows the 'TRDQR:Trade Query [03/05/2014]' form interface. It features a search bar at the top with 'Search Menu' and a date '03/03/2014'. On the right are buttons for 'FIN', 'Logout', and user 'rounaka'. Below the search bar is a header row with fields for 'Trade Type' and 'Trade Reference No'. The main search area contains various filters: 'Account No', 'Trade Currency', 'Inventory Account No', 'Trade Date From-To', 'Value Date From - To', 'Execution Market', 'Instrument Type', 'Buy/Sell Flag', 'Entry Date Form - To', 'External Reference No', 'Account Balance Type', 'Data Source', 'Trade Status', and 'Sort Criteria'. A 'PERSONALIZE' button is located on the left. A 'Deleted items drop zone' is highlighted with a red arrow pointing from the 'Instrument Type' field. A green box highlights the 'RESTORE ALL' button. A yellow box highlights the 'Action buttons replaced by Save, Cancel and Reset buttons' area, which includes 'SAVE', 'CANCEL', and 'RESET' buttons.

Figure 01 Form items can be dragged and dropped into desired place

This screenshot shows the same 'TRDQR:Trade Query [03/05/2014]' form as Figure 01. The 'Instrument Type' field has been removed, indicated by a red dashed arrow pointing to its position and a red 'X' button. The 'Remove this field' button is visible. The rest of the form structure and buttons are identical to Figure 01.

Figure 02 User can remove the form Items by clicking the cross button

Step 1

Personalization feature is present in the query screens such as Trade Query. To enable the form personalization feature, the code snippet below needs to be included in the query form as present in tradeQueryCriteria.jspx :

```

<div class="formItem">
<form:label path="externalReferenceNo">
<spring:message code="trade.label.external.referenceno" htmlEscape="false" />
</form:label>
<span><form:input value="" path="externalReferenceNo" class="textBox txtNormal" /></span>
<span class="remove" title="Remove this field" style="display: none;">
<spring:message text="" htmlEscape="false" />
</span>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>

```

Step 2

The above code snippet will show a cross button beside every form-item when Personalize button is clicked. When the form item is removed by clicking the cross button a placeholder should be added in the query form as present in tradeQueryCriteria.jspx :

```

<div class="formItemBlock delFormItems deltopitems" style="display: none;">
<!-- Deleted Item goes here -->
<input type="button" class="restoreAllBtn" value="Restore All" />
<div class="clear clear-block">
<spring:message text="" htmlEscape="false" />
</div>
</div>

```

- In personalization mode each form item is followed by a cross item to remove the element. All form items are also draggable in this mode.
- If any form item is deleted then the restore section is displayed.
- In this mode Personalize button is disabled and Save and Cancel buttons are enabled.

For all the features mentioned above see Personalization in Query/Query Result document and Query/Query Result - Initialize new query page section for further details.

3.2.2 Saved Query

TRDQR:Trade Query [20140503]

Trade Type	EQ	Trade Reference No	<input type="text"/>
Account No	<input type="text"/>	Inventory Account No	<input type="text"/>
Security Code	<input type="text"/>	Trade Currency	<input type="button"/>
Settlement Currency	<input type="button"/>	Trade Date From-To	<input type="text"/> <input type="button"/> <input type="button"/>
Value Date From - To	<input type="text"/> <input type="button"/> <input type="button"/>	Instrument Type	<input type="button"/>
Execution Market	<input type="text"/>		
Buy/Sell Flag	<input type="button"/>	Principal/Ag	<input type="text"/>
Entry Date Form - To	<input type="text"/> <input type="button"/> <input type="button"/>	Last Entri	<input type="button"/>
External Reference No	<input type="text"/>		
Account Balance Type	<input type="button"/>		
Trade Status	Data Source <input type="button"/>		
Sort Criteria	① Trade Da <input type="button"/> DESC	② <input type="button"/> DESC	
	③ <input type="button"/> DESC	④ <input type="button"/> DESC	
	⑤ <input type="button"/> DESC	⑥ <input type="button"/> DESC	
	⑦ <input type="button"/> DESC	⑧ <input type="button"/> DESC	

Save Query

Criteria Name?

OK CANCEL

PERSONALIZE **SAVE QUERY** **RESET** **SUBMIT**

Save Query feature is present in the query screens such as Trade Query. See [Rich UI Framework - Save/Template Query \(Expression support\)](#) section and [Query/Query Result - Initialize new query page](#) section for complete details.

3.3 Popup

3.3.1 Popup Component

Accounts

Account No	<input type="text"/>	Account Name	<input type="text"/>
Counter Party Type	<input type="text"/> INTERNAL	<input type="button"/>	<input type="button"/> RESET <input type="button"/> SUBMIT

See Popup Component and Rich UI Framework - Popup framework section for complete details.

3.3.2 Window Docking

In finch, there is an option to open the details in Dialog popup. Every dialog window have Multiple Document Interface feature like minimize, maximize, restore, close and minimize at specific taskbar zone.

Screenshot Sequence of Steps

TRDRS:Trade Query Result [20140503]

Trade D...	Reference N...	Trade ...	Trade Type	Value Date	Account B...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...	
20131114	T20131009002233	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST	
20131114	T20131009002235					121212007	USD	S	1q2w3e	NORMAL	ST	
20131114	T20131009002195					121212007	USD	S	2222	NORMAL	ST	
20131114	T20131009002330					121212007	USD	S	222	NORMAL	ST	
20131114	T20131009002348	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST	
20131114	T20131009002193	USD	EQ	20131115	1	121212007	USD	S	2222e	NORMAL	ST	
20131114	T20131009002185	USD	EQ	20131115	1	121212007	USD	S	555	NORMAL	ST	
20131114	T20131009002306	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST	
20131114	T20131009002294	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST	
20131114	T2013100900406	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST	
20131114	T20131009002298	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002283	USD	EQ	20131115	1	121212007	USD	S	3444	NORMAL	ST	
20131114	T20131009002285	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST	
20131114	T20131009002287	USD	EQ	20131115	1	121212007	USD	S	444	NORMAL	ST	
20131114	T20131009002346	USD	EQ	20131115	1	121212007	USD	S	111	NORMAL	ST	
20131114	T20131009002155	USD	EQ	20131115	1	121212007	USD	S	12346	NORMAL	ST	
20131114	T20131009002281	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002304	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST	
20131114	T20131009002116	USD	EQ	20131115	1	121212007	USD	S	TRDDDD	NORMAL	ST	
20131114	T20131009002217	USD	EQ	20131115	1	121212007	USD	S	33	NORMAL	ST	

Figure 1

TRDRS:Trade Query Result [20140503]

dialog popup

The screenshot shows a table of trade records at the top, with one specific row highlighted. A red arrow points from the text "dialog popup" to this highlighted row. A modal dialog box titled "[T20131009002223] Trade details" is displayed over the table. The dialog has a title bar with "General" and "Details" tabs, and a "PRINT" button at the bottom right. The "General" tab is selected. Inside the dialog, there are several sections: "Counterparty Account", "Inventory Account", "Security Information", "Basic Trade Attributes", "Date", "Volume", and "Price". Each section contains various trade-related fields. Red arrows point to the "minimize", "maximize", and "close" buttons in the top right corner of the dialog box. Another red arrow points from the text "dialog popup docked here" to the bottom left corner of the dialog box.

Trade D...	Reference No	Trade ...	Type	Value Date	Account E...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Instrument...
20131114	T20131009002223	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST

[T20131009002223] Trade details

General Details

Counterparty Account

Counter Party Code INTERNAL
Account No 1212120091
Short Name USA-JPMC-CUR

Trade Reference

Reference No T20131009002223 - 3
Trade Status NORMAL
External Reference No 9988
Data Source SCREEN

Currency

Trade Currency USD
Settlement Currency USD
Execution Market HSBC

Inventory Account

Inventory Account No 121212007
Inventory A/C Short USA-JPMC-CUR
Name

Security Information

Instrument Type ST
Security Code IBM
Security Name IBM_ST

Basic Trade Attributes

Trade Type EQ
Buy/Sell Orientation AB
Principal/Agent Flag A
Account Balance Type 01

Date

Trade Date 20131114
Value Date 20131115

Volume

Quantity 12
Net Amount 43.2
Net Amount In Trading Currency

Price

Price 1

PRINT

[T20131009002223] Trad...

© 2012 NRI Financial Technology, INDIA. Powered By finch (v 2.3.1.0-SNAPSHOT)

Figure 2

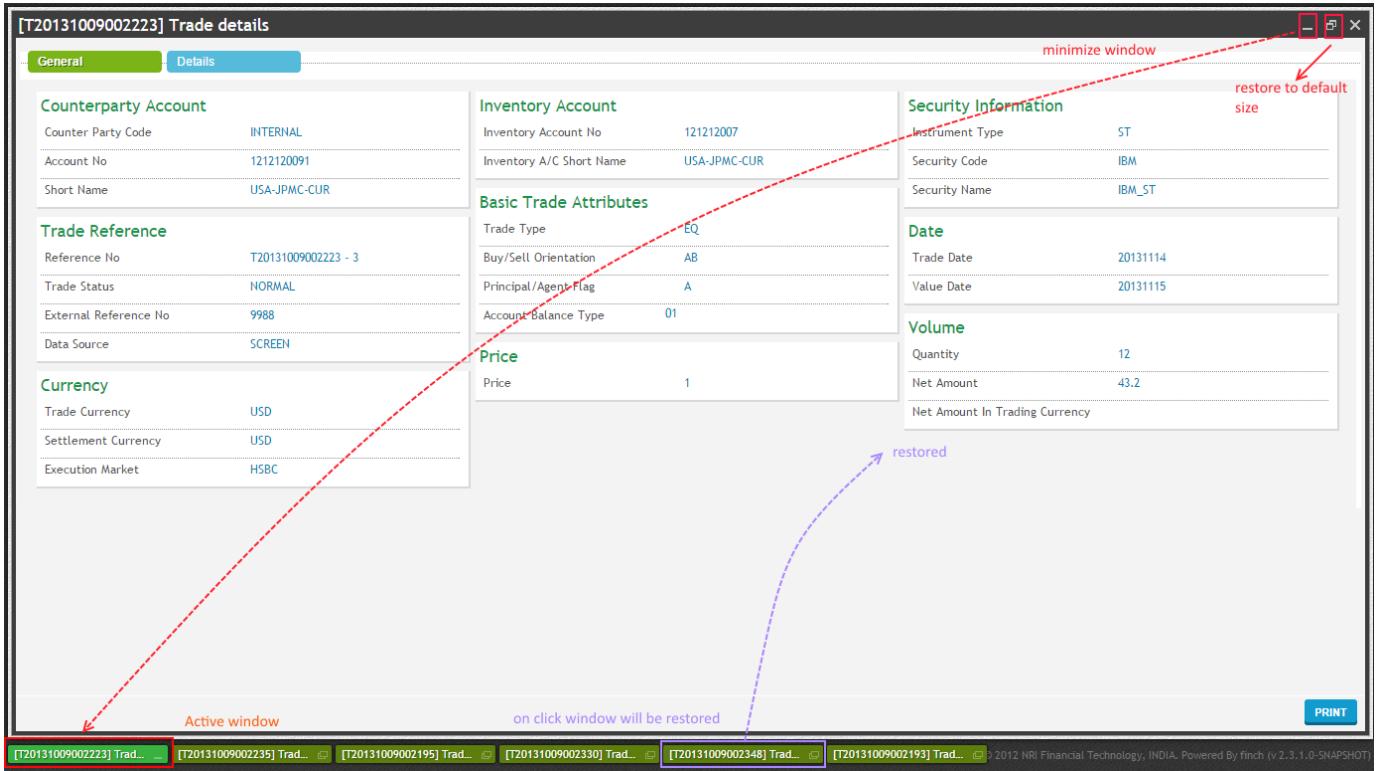


Figure 3

TRDRS:Trade Query Result [20140503]												
Page	1	of 100	Records:	1989								
Trade D...	Reference No	Trade ...	Trade Type	Value Date	Account B...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Instrument	
20131114	T20131009002223	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST	
20131114	T20131009002235	USD	EQ	20131115	1	121212007	USD	S	1q2v3e	NORMAL	ST	
20131114	T20131009002195	USD	EQ	20131115	1	121212007	USD	S	2222	NORMAL	ST	
20131114	T20131009002330	USD	EQ	20131115	1	121212007	USD	S	222	NORMAL	ST	
20131114	T20131009002348	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST	
20131114	T20131009002193	USD	EQ	20131115	1	121212007	USD	S	2222e	NORMAL	ST	
20131114	T20131009002185	USD	EQ	20131115	1	121212007	USD	S	555	NORMAL	ST	
20131114	T20131009002306	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST	
20131114	T20131009002294	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST	
20131114	T20131009000406	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST	
20131114	T20131009002298	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002283	USD	EQ	20131115	1	121212007	USD	S	3444	NORMAL	ST	
20131114	T20131009002285	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST	
20131114	T20131009002282	USD	EQ	20131115	1	121212007	USD	S	444	NORMAL	ST	
20131114	T20131009002346	USD	EQ	20131115	1	121212007	USD	S	111	NORMAL	ST	
20131114	T20131009002155	USD	EQ	20131115	1	121212007	USD	S	12346	NORMAL	ST	
20131114	T20131009002281	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST	
20131114	T20131009002304	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST	
20131114	T20131009002116	USD	EQ	20131115	1	121212007	USD	S	TRDDDD	NORMAL	ST	
20131114	T20131009002217	USD	EQ	20131115	1	121212007	USD	S	33	NORMAL	ST	

After opening maximum number of popups alert will be shown

Figure 4

Step 1

We can take an example for implementing the feature of MDI or Window docking from Trade Query Result. For reference number, the column definition in Trade Query Result is:

```

var grid_result_columns = [
    {name:"<spring:message code="trade.label.reference.no" htmlEscape="false" />",
     field:"tradeReferenceNo",id:"tradeReferenceNo", width:147,sortable:true,formatter:
     Slick.Formatters.TradeDetailViewFormater,isForAuth:false },
    .
    .
];

```

Step 2

In the Trade Query Result Screen we need our Reference Number column to display a hyperlink, for opening the detail in a popup window with MDI feature, to achieve this we need to follow the following steps:

Define the formatter in "sample-trd/src/main/web/scripts/trd/sample-trd-formatters.js".

First add an entry in the Formatters object, which is nothing but a unique name to function mapping.

```

$.extend(true, window, {
    "Slick": {
        "Formatters": {
            "TradeDetailViewFormater" : TradeDetailViewFormater
        }
    }
});

```

and then define the function itself. This function is called its formatter by using the unique name above (more details below), and then displays whatever this function returns.

```

function TradeDetailViewFormater(row, cell, value, columnDef, dataContext) {
    if (value == null || value === "") {
        return "";
    }
    var urlPath = '/trd/query/';
    if ($('form').attr('action').indexOf('/trd/amendcancel') > -1) {
        var urlPath = '/trd/amendcancel/';
    }
    var markup = "" + "<span class='detail-view-hyperlink' " + "view='tradeDetailView' "
        + "referenceno=" + value + " " + "tradepk=" + dataContext.tradePk
        + " " + "dialogidentifier='tradeDetail" + dataContext.tradePk + "'"
        + "duplicate = " + FINCH.detailDialogTypeBehaviour + ""
        + "dialogTitle='[" + value + "] " + SAMPLE$TRD$i18n.formatter.trade_Details
        + " " + "href=" + urlPath + "details/" + dataContext.tradePk
        + " /" + value + "?diffEnableFlag=" + columnDef.isForAuth + ">" + value + "</span>";
    return markup;
}

```

3.4 Grid

3.4.1 Non-Editable Grid

In finch , non-editable grid is used to display query result data.

[Screenshots Sequence of Steps](#)

In finch for grid component, SlickGrid plugin has been used. We take an example screenshot from Trade Query Result screen.

TRDRS:Trade Query Result [20140503]													Query	Result
Page	1	of 100	Records:	1989										
Trade D...	Reference No	Trade ...	Trade Type	Value Date	Account B...	Inventory ...	Settlement...	Buy/Sell Fl...	External R...	Trade Stat...	Instrumen...			
20131114	T20131009002223	USD	EQ	20131115	1	121212007	USD	S	9988	NORMAL	ST			
20131114	T20131009002235	USD	EQ	20131115	1	121212007	USD	S	1q2w3e	NORMAL	ST			
20131114	T20131009002195	USD	EQ	20131115	1	121212007	USD	S	2222	NORMAL	ST			
20131114	T20131009002330	USD	EQ	20131115	1	121212007	USD	S	222	NORMAL	ST			
20131114	T20131009002348	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST			
20131114	T20131009002193	USD	EQ	20131115	1	121212007	USD	S	2222e	NORMAL	ST			
20131114	T20131009002185	USD	EQ	20131115	1	121212007	USD	S	555	NORMAL	ST			
20131114	T20131009002306	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST			
20131114	T20131009002284	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST			
20131114	T20131009000406	USD	EQ	20131115	1	121212007	USD	S	1111	NORMAL	ST			
20131114	T20131009002298	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST			
20131114	T20131009002283	USD	EQ	20131115	1	121212007	USD	S	3444	NORMAL	ST			
20131114	T20131009002285	USD	EQ	20131115	1	121212007	USD	S	22	NORMAL	ST			
20131114	T20131009002287	USD	EQ	20131115	1	121212007	USD	S	444	NORMAL	ST			
20131114	T20131009002346	USD	EQ	20131115	1	121212007	USD	S	111	NORMAL	ST			
20131114	T20131009002155	USD	EQ	20131115	1	121212007	USD	S	12346	NORMAL	ST			
20131114	T20131009002281	USD	EQ	20131115	1	121212007	USD	S	333	NORMAL	ST			
20131114	T20131009002304	USD	EQ	20131115	1	121212007	USD	S	11	NORMAL	ST			
20131114	T20131009002116	USD	EQ	20131115	1	121212007	USD	S	TRDDDD	NORMAL	ST			
20131114	T20131009002217	USD	EQ	20131115	1	121212007	USD	S	33	NORMAL	ST			

Figure 1 Trade Query Results

There are three definitions that we need for rendering the grid

- Column Definition
- Data
- Settings

3.4.2 Editable Grid

In finch , editable grid is used to add records to the grid dynamically. There are two types of editable grids in finch.

Screenshots Sequence of Steps

Editable grid 1 :

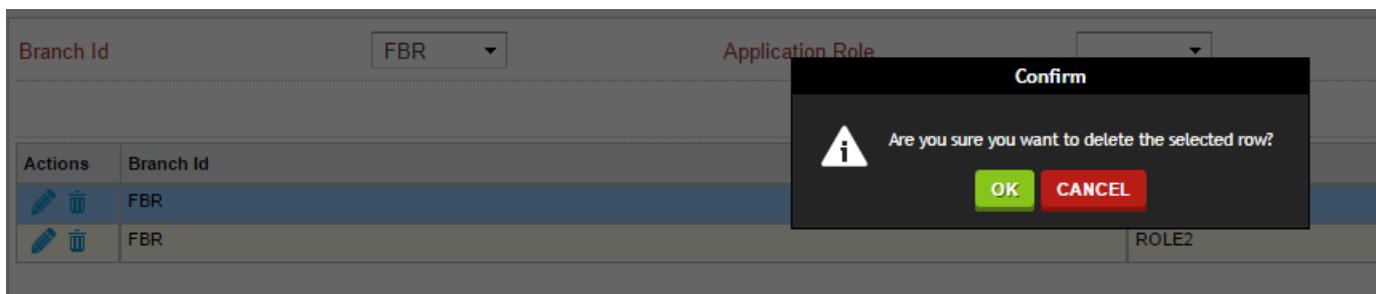
An example of this type of grid can be found in the Employee Entry screen. In this type of grid, the rows cannot be edited in place, instead they are added by using input controls outside the body of the grid.

Actions	Branch Id	Application Role
	FBR	ROLE1
	FBR	ROLE2

On clicking edit icon

Branch Id	Application Role	Actions
FBR	ROLE1	
FBR	ROLE2	

On clicking trash icon



Editable grid 2 :

In this type of grid the rows can be edited in place. An example of this type of grid can be seen in Account Entry screen in the sample application

Account Code Information		
	Account No Type	Account No
	BRKNO	1234567
	BRKNO	

Rows can be reordered by dragging

	Account No Type	Account No
	BRKNO	1234567
	BRKNO	9683524

On clicking the reset button

	Account No Type	Account No
	BRKNO	1234567
	BRKNO	

The code snippet below shows how to initialise the Employee Entry grid:

```

var userAppIRoleGridConf = {
    editMode      : 'both',
    editCallback   : editBranchRoleHandler,
    deleteCallback : removeUserAppIRoleMappingHandler
};

var userAppIRoleColumns = [
    {name:<spring:message code="inf.employee.label.branchid" htmlEscape="false" />, field:"branch",
    id:"branch"},
    {name:<spring:message code="inf.employee.label.applicationsrole" htmlEscape="false" />, field:"role",
    id:"role"}
];

$UserAppIRoleGrid = $('#userAppIRoleGrid', $employeeEntry$context);
var $UserAppIRoleGridObject = $UserAppIRoleGrid.fincheditablegrid(userAppIRoleData, userAppIRoleColumns,
userAppIRoleGridConf);
if( $UserAppIRoleGrid.data != 'undefined')
    $UserAppIRoleGrid.data('theUserAppIRoleGrid', $UserAppIRoleGridObject);

```

The code snippet below shows how editable grid 2 has been initialised in Account Entry

```

var actCodeInfoGridConf = {
    editMode      : 'both',
    newRowPosition : 'end',
    validateBeforeAdd : true
};

var actCodeInfoGridColumns = [
    {name:<spring:message code="ref.account.entry.label.accountntotype" htmlEscape="false"/>,
    field:"accountNoType", id:"accountNoType", populate: "<c:out value='${commandForm.accountNoTypeList}'"
    escapeXml="false"/>" , autoPopulate:true, editor: Slick.Editors.SelectCellEditor,
    validator:Slick.Validators.RequiredFieldValidator},
    {name:<spring:message code="ref.account.entry.label.accountno" htmlEscape="false"/>,
    field:"accountNo", id:"accountNo", defaultValue:"", editor: Slick.Editors.FINCHText,
    validator:Slick.Validators.RequiredFieldValidator}
];

<c:forEach items="${commandForm.accountNoList}" var="dl">
item = {};
rowId+=1;
item.id      = "finch_" + rowId;
item.accountNoType = "<c:out value='${dl.accountNoType}' />";
item.accountNo  = "<c:out value='${dl.accountNo}' />";

actCodeInfoData.push(item);
</c:forEach>

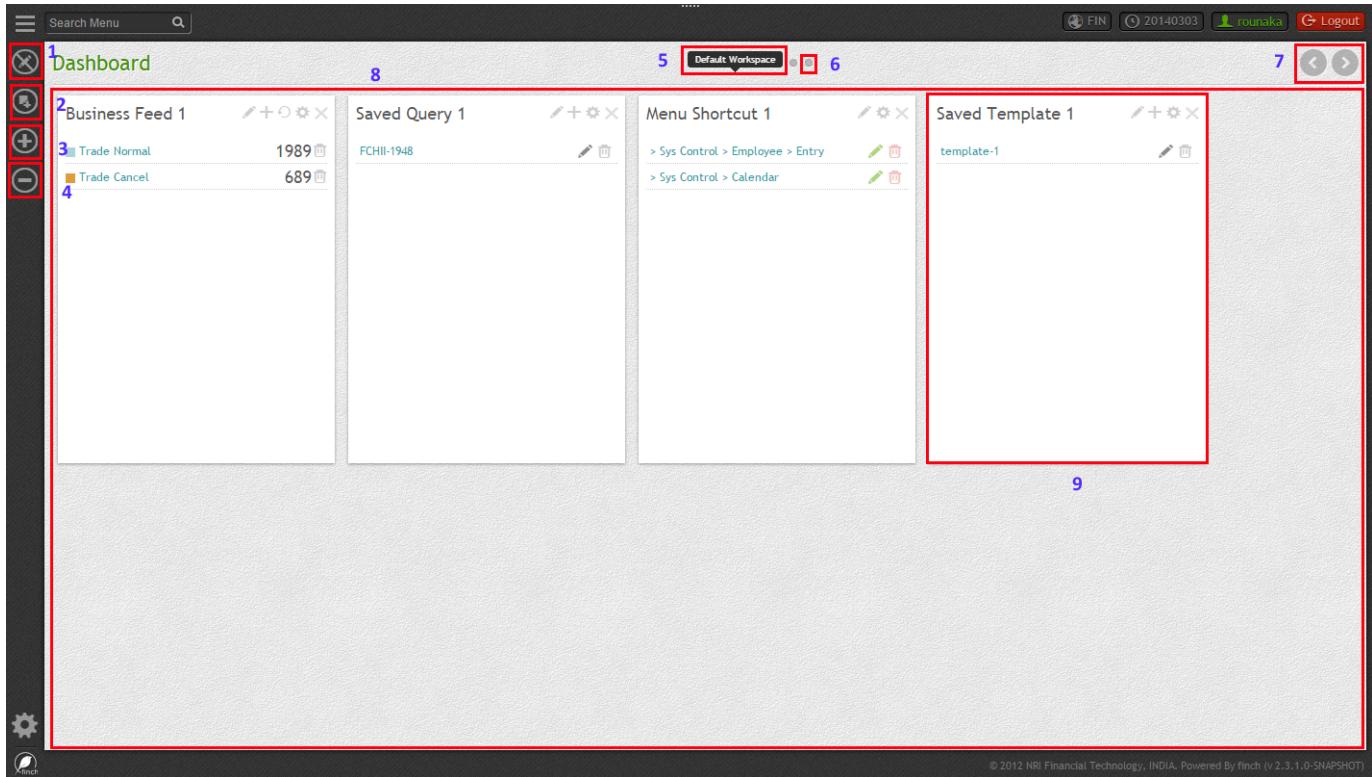
FINCH.gridResetData = actCodeInfoData;

$actCodeInfoGrid = $('#actCodeInfoGrid', $accountEntry$context);
var $actCodeInfoGridObject = $actCodeInfoGrid.fincheditablegrid2(actCodeInfoData,
actCodeInfoGridColumns, actCodeInfoGridConf);
$actCodeInfoGrid.data('theActCodeInfoGrid', $actCodeInfoGridObject);

```

See [Grid Component](#) and [Finch Grid](#) documents for more details.

3.5 Dashboard



Legends

1. Dashboard Editable/Non-editable toggle
2. Add widget
3. Add workspace
4. Delete workspace
5. Workspace name
6. Workspace indicator
7. Workspace navigation buttons
8. Workspace area
9. Widget

See [Dashboard Infrastructure](#) document for complete details.

4. See Also

<http://api.jquery.com/> on jQuery

<http://www.willcam.com/cmat/html/crossref.html> on HTML tags

http://en.wikipedia.org/wiki/Multiple_document_interface on MDI

3.4.6.2.27 Entry Screen Customization in finch

1. Scope

Using finch one can create some entry based screens like “Trade Entry” where certain non-mandatory fields may be unnecessary in the context of different enterprises. Those fields can be removed by configuring it at the enterprise level by specifying it in the preference store of the server where the application is deployed. In addition to it the order in which the fields must be displayed can also be configured through it. The customization is done solely on the client side, and relation between any dependent fields must be handled by the application. This document gives a detailed account of the implementation of the customization of an entry based screen.

2. Prerequisites

Refer to [Web Component](#) for an overall idea about the iGV web architecture.

Refer to [Web Layer](#) for the implementation details of web components.

Refer to [Wizard Based Entry](#) for understanding the entry mechanism and screen layouts of the iGV wizard.

Refer to [Single Page Entry](#) for details on how to create an entry page.

3. Assumptions & Limitations

1. The customization feature is only applicable for screens based on “finch-wizard.js”.
2. The implementation will work with screens based only on the traditional form based layout and not on any table layout. Hence anywhere where the customization needs to be implemented the layout block needs to be in a form items based layout except for a particular scenario in case of a details page which is discussed later in the document.
3. Every page must have a designated unique screenId, versionNo and pageIndex(for entry based wizard screens, not for single page details view) for uniquely identifying it while applying the customization.
4. The screens for amend, cancel, copy all needs to be in the similar expected layout as that of an entry screen for the customization to work uniformly across all these screens.
5. Only non-mandatory fields should be considered for customization.
6. Each of the fields in the form should have an unique name attribute assigned to them.
7. Dynamic List components like “RR Number” are not to be considered for customization.
8. Composite component are not considered for customization. For example if a form item html element is removed from a block, then the corresponding field in a grid representing the element would not be removed simultaneously.

4. Steps for customization

4.1 Change the entry screens page layout to make it a form based layout

The customization is based on the principle that each form item block(div) having a class 'formItem' and containing within it an html element will either be removed or persisted on the page, depending upon the declaration of the value of the name attribute of the html element in the configuration entry file. A set of form item blocks will be inside a block(div) having a class 'formItemBlock'. The input elements of the 'formItem' block is uniquely identified by the value of its 'name' attribute. Hence it is an absolute must that every input element within a 'formItemBlock' must have an unique name attribute assigned to it. Only the html elements with the value of the name attribute declared in the configuration file for the current page will be rendered finally. The image below shows the changes required to be done for changing a table layout to a form based layout.

```

<div class="detailBlock entrySingleCol">
    <table class="tableStyle sixCol">
        <tr>
            <td><label><spring:message code="trd.tradeentryaction.label.tradetype" href="#" /></label></td>
            <td><span class="detailTxtVal">${commandForm.tradeScreenData.tradeType}</span></td>
            .....
            .....
        </tr>
        <tr>
            <td><label><spring:message code="trd.tradeentryaction.label.valuedate" href="#" /></label></td>
            <td><span class="detailTxtVal">${commandForm.tradeScreenData.valueDateStr}</span></td>
            .....
            .....
        </tr>
    </table>
</div>

```

The above image shows the difference in layout between a page designed with a table layout and the same content displayed with a form based layout. The block on the left is the initial table based layout and that on the right is the form based layout. Following are the points of difference that needs to be made as can be seen in the image above :

1. The main wrapper block for the items will not be a table, instead it will be a div. The div should have two classes "formItemBlock" and "dottedBg".
2. Here a row of the table is transformed to a corresponding form based layout. Each column or 'td' element of a row in the table is transformed individually to a formItem block.
3. There are two 'td' blocks for each field, one for the label of the field and the another for displaying the value of the field. The 'td' blocks are to be replaced by a div having the class 'formItem'. Inside the formItem block the label element is to be placed, as it was there in the first 'td' element of the table layout.
4. It should be followed by a span having the class 'detailDisplay'. The name attribute assigned to the span must have an unique value in the context of the current page. The actual text to be displayed is to be placed in this span element.
5. The last statement inside a 'formItem' block should be a div element with the class 'clear' applied to it.
6. In the similar manner as stated in the above points all 'td' elements are to be replaced by a 'formItem' one after another belonging to different rows ('tr') of the initial table layout.
7. Before closing the 'formItemBlock' block, another div having the class 'clear' applied to it must be inserted there.
8. Before closing the div which is wrapping the 'formItemBlock', another div having the class 'linehide' applied to it must be inserted there.

Beside the above mentioned changes, it is advised to keep all the 'formItemBlock' hidden initially. This would prevent the initial flickering of elements when the page loads due to the customization operations. Then manually show the hidden block(s) at the end of the page loading in the FINCH\$Wizard\$onPageComplete block.

4.2 Change the detail screen page layout to make it a form based layout

This customization is to be done for those details screen which doesn't use the entry confirmation pages as its detail view pages. The detail view pages are displayed when an user clicks on any hyperlink in a row of the query result grid which would result in the opening of a detail dialog. One example of it is the file tradeDetailView.jspx. The layout of the pages in this case is a set of nested tables. Each row of the concerned table block is to be replaced by divs having class 'detailItem'. Initially each table element signifying a block or a logical section has a class 'detailSmlBlk'. This is to be replaced by 'divs' having class 'detailItemBlock'. The current layout of the detail view is such that the sections of the pages are all either rows or columns of the parent table which in itself is a row or column of its parent table. The table layout needs to be replaced by a form based layout only in the sections where the actual displayable data lies. The image below shows the area of the page which needs to be replaced by a form based layout:

[T20131009002223] Trade details

General

Details

Table with class
by div have

Counterparty Account

Counter Party Code	INTERNAL
Account No	1212120091
Short Name	USA-JPMC-CUR

Invento

Inventory

Inventory

Basic T

Trade Ty

Buy/Sell

Principa

Price

Price

Trade Reference

Reference No	T20131009002223 - 3
Trade Status	NORMAL
External Reference No	9988
Data Source	SCREEN

Currency

Trade Currency	USD
Settlement Currency	USD
Execution Market	HSBC

So the concerned area for layout change is every nested table having the class 'detailSmlBlk' and the contents within it. Here also like in the case of other entry based screens, the element which actually holds the displayable value for a field should have a 'name' attribute with an unique value. This 'name' value will be used to uniquely identify a field and depending on the declaration of the name in the configuration the corresponding field would either be removed or persisted in the page. The image below shows the changes required to be done for changing a table layout to a form based layout.

```
<table class="detailSmlBlk">
    <tr>
        <td><form:label path="">${data.counterPartyLabel}</form:label></td>
        <td>${data.counterPartyCode}</td>
    </tr>
    <tr>
        <td><form:label path=""><spring:message code="trade.label.account.no" htmlEscape="false"/>
        <td><span class="detailTxtVal">${data.accountNo}</span></td>
    </tr>
    <tr class="noBdr">
        <td><form:label path=""><spring:message code="trade.label.short.name" htmlEscape="false"/>
        <td>${data.counterPartyName}</td>
    </tr>
</table>
```

The above image shows the difference in layout between a detail view page designed with a table layout and the same content displayed with a form based layout. The block on the left is the initial table based layout and that on the right is the form based layout. Following are the points of difference that needs to be made as can be seen in the image above :

1. The changes needs to be done in the table block which is closest to the actual row of data, for example here in the table with the class 'detailSmlBk'. The table element needs to be replaced with a div having class 'detailItemBlock'. There are other table blocks wrapping this table block, but since it is closest to the actual row of data, hence the customization is commenced from here.
2. Each row(tr) of the table needs to be replaced with a div having class 'detailItem'.
3. The two enclosing <td> elements inside the tr element needs to replaced directly by the the html block wrapped inside the td elements.
4. Instead of using 'form:label' one may also choose to use the native html 'label' tag.
5. The last row(tr) of the current table block has a class 'noBdr' applied to it, no such classes needs to be applied to the last detailItem div block.
6. As discussed previously every element should have an unique 'name' attribute, so here also the span should have a name attribute with an unique value

Beside the above mentioned block, one also needs to modify those table blocks which has six column style appearance, i.e the total block is a single table with each row having three columns of data. The image below shows the changes that needs to be done :

```


|                                                                                                                                                                                     |                                                                                                                                                                                   |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <form:label path=""><spring:message code="trade.label.order.refno" htmlEscaped="true" />                                                                                            | <valueDifference:valueDifferenceId oldValue="\${oldScreenData.orderReferenceNo}" newValue="\${newScreenData.orderReferenceNo}" diffEnableFlag="\${commandForm.diffEnableFlag}" /> |  |
| <form:label path=""><spring:message code="trade.label.etc.refno" htmlEscaped="true" />                                                                                              | <valueDifference:valueDifferenceId oldValue="\${oldScreenData.etcReferenceNo}" newValue="\${newScreenData.etcReferenceNo}" diffEnableFlag="\${commandForm.diffEnableFlag}" />     |  |
| <form:label path=""><spring:message code="trade.label.sender.refno" htmlEscaped="true" />                                                                                           |                                                                                                                                                                                   |  |
| <valueDifference:valueDifferenceId oldValue="\${oldScreenData.senderReferenceNo}" newValue="\${newScreenData.senderReferenceNo}" diffEnableFlag="\${commandForm.diffEnableFlag}" /> |                                                                                                                                                                                   |  |
|                                                                                                                                                                                     |                                                                                                                                                                                   |  |
|                                                                                                                                                                                     |                                                                                                                                                                                   |  |
|                                                                                                                                                                                     |                                                                                                                                                                                   |  |


```

1. A class 'tableBox' needs to applied to the table to signify that the table must be considered for customization.
2. Instead of using 'form:label', the simple html <label> element must be used. In the 'td' of the label element a name attribute must be specified with an unique value which will be used to identify the column. If the given name attribute is not present in the configuration for customization, then the corresponding label and the following td holding the value against it will not be displayed in the table.
3. The elements are placed in the table in the order in which the name of the elements are stated in the configuration. If the colspan of a particular element needs to be more than 1 such as 3 or 5, then the element is either placed in the current row or the next row as per the available space in the row and the required colspan of the element.

One thing that needs to be considered in the case of detail dialogs like this(detail dialogs in which a different view to that of the entry detail/confirmation pages are used) is that in these screens the 'screenId' and the 'versionNo' fields are not populated by the same process as is done in the case of entry, amend, cancel or copy screens. The 'screenId' and 'versionNo' of a screen are read from the hidden fields within the displayed page with the name 'screenId' and 'versionNo' respectively. The value assigned to those fields are interpreted as the corresponding values of screenId and versionNo. Hence if in the detail page that you are modifying has no such fields or either any one of them are missing, then they must be added to the page, and the value should be populated from the server side.

Markup for adding a hidden 'versionNo' field

```
<input type="hidden" name="versionNo" value="${detailsVersionNo}" />
```

The value can be populated by any means as the application may want, but the easiest way is to put the value for screenId or versionNo in the modelMap itself and to retrieve it directly in the jspx page by using EL.

Adding the value for the field versionNo in the model Map

```
finchModelMap.getModelMap().addAttribute("detailsVersionNo", "1.2");
```

4.3 Specify the entry screen configuration in a json file :

A sample json configuration for Trade related screens is depicted below :

Sample-Trade-Entry-Screen-Config.json

```

/* The number specified within the comments relates to the point described below */

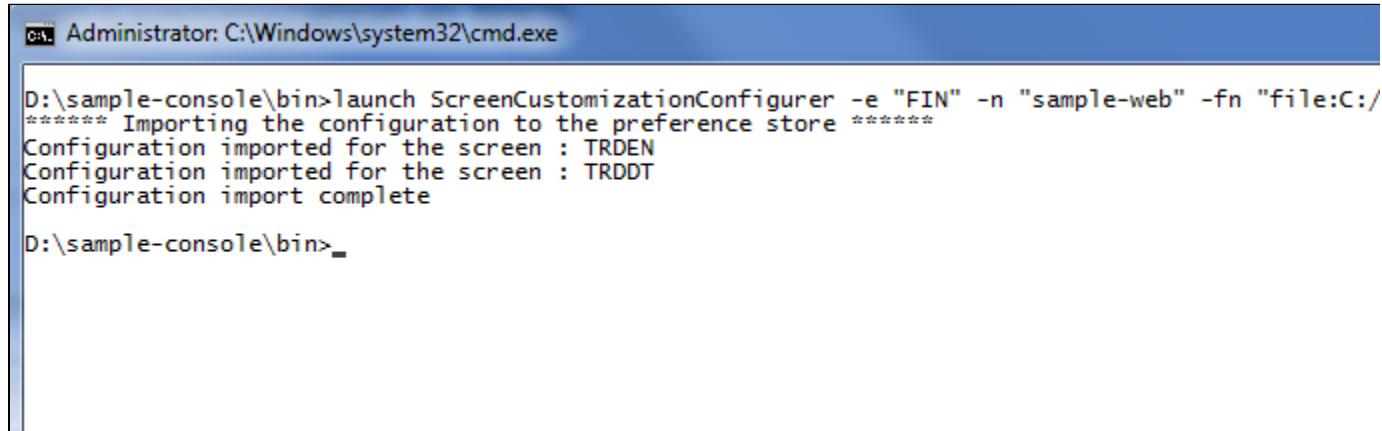
/* 1 */
{
  "TRDEN": { /* 2,3 */
    "1.2": { /* 4 */
      "0": { /* 4 */
        "ScreenConfig": ["tradeType", "subTradeType"] /* 4 */
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    },
    "2": {
      "0": {
        "ScreenConfig": ["accountType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"] /* 6,7,8 */
      }
    }
  },
  "TRDDETAIL": {
    "1.2": {
      "NA": { /* 5 */
        "ScreenConfig": ["tradeType", "subTradeType"]
      }
    }
  },
  "TRDAM": {
    "1.2": {
      "0": {
        "ScreenConfig": ["tradeType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    },
    "2": {
      "0": {
        "ScreenConfig": ["accountType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    }
  },
  "TRDCX": {
    "1.2": {
      "0": {
        "ScreenConfig": ["tradeType", "subTradeType"]
      },
      "1": {
        "ScreenConfig": ["testData", "subTestData", "moreData"]
      }
    }
  }
}

```

The following are the set of observations that must be noted in the above mentioned JSON config file :

1. The whole configuration should be a valid JSON object.
2. Inside the JSON object the configuration related to each screens should be described. It should be noted here that for each screens belonging to a particular module distinct set of configurations needs to be described for each of it's concerned actions, i.e for amend, cancel, copy, screen configuration needs to be described separately even though they may be same as that to the entry screen and uses the same jspx pages. The purpose of this design is to be as verbose as possible about configurations of screens with different screenIds and to allow custom screen configuration of the same jspx pages in the context of different actions if any such requirement arises.
3. The top level entry in the JSON configuration for a screen is specifying it's screenId. For example the configuration for a screenId 'TRDEN' should be an object with the key being the screenId, and the value being another object containing the configuration details.
4. The nesting of the JSON configuration for a screen should start with the screenId followed by the versionNo, pageIndex and a key "ScreenConfig". The versionNo is the version no of the page, pageIndex is basically the tab order of the page in the wizard for which the configuration is being described.
5. If "NA" is given in the pageIndex section, then it signifies that pageIndex is not applicable for the current screen, i.e, the screen doesn't have multiple pages. Hence the preference would be saved directly under the versionNo key in the preference store. This is applicable in cases of single page details screen.
6. The actual configuration for the screen is described inside an array which serves as the value for the key 'ScreenConfig'. The array should be populated with the value of the 'name' attribute of the fields which you want displayed in the screen. The fields from the jspx whose names are absent in this configuration will not be displayed in that page for that combination of screenId, versionNo and pageIndex.
7. It should be noted here that keeping in sync with the behavior of query screen configuration, here also the order in which the names are stated inside the array is significant. The fields would be displayed in the page in the same order as the names are stated in the configuration array.
8. Another thing that must be noted here is that though the order of fields displayed in the page is according to the order in which the names are stated in the configuration file, the order is only preserved within the same 'formItemBlock' or 'detailItemBlock'. For example, lets say there are five 'formItems' with names "A", "B", "C", "D" and "E". The 'formItems' "A", "B" and "C" belongs to the 'formItemBlock' "Block1" and 'formItem' "D" and "E" belongs to the 'formItemBlock' "Block2". Now let us suppose that there is a configuration array like this ["B", "C", "E", "A", "D"], then though "E" is stated before "A", "E" will not be displayed before "A" in the page. This is because "E" belongs to a different 'formItemBlock' than "A", "B" and "C". In the 'formItemBlock' "Block 1" the order of items would be "B", "C" followed by "A", and in the 'formItemBlock' "Block2" the order of items would be "E" followed by "D". The same behavior is true for 'detailItems' and 'detailItemBlock'.
9. The configuration does not take into consideration the 'mode' (such as 'EDIT', 'VIEW', 'CONFIRMATION' etc.) in which the page is. So in cases where there are additional fields in a mode for the same combination of screenId, versionNo and pageIndex in respect to that of in some other mode, the configuration should take care of that extra fields and must state them in the configuration array if they want them to be displayed. The fields are searched with the names in the given array and are then persisted in the page accordingly, so if there is any name stated in that configuration for which no such field exists in the jspx for the current mode it would cause no unanticipated behavior and the name would simply be ignored. But in a different mode for the same combination of screenId, versionNo and pageIndex if a field exists with that name then that field will be acknowledged and displayed accordingly.

4.3 Run a tool to import the required configuration to the preference store of the server :



```
Administrator: C:\Windows\system32\cmd.exe
D:\sample-console\bin>launch ScreenCustomizationConfigurer -e "FIN" -n "sample-web" -fn "file:C:/*****
***** Importing the configuration to the preference store *****
Configuration imported for the screen : TRDEN
Configuration imported for the screen : TRDDT
Configuration import complete

D:\sample-console\bin>
```

A CLI tool is provided to import the entry screen configuration to the preference store of the server in which the application is to be deployed. The tool should be run in the following way :

Tool for importing entry based screen preferences

```
launch ScreenConfigurer -e "NCS" -n "igv-tomcat" -fn "trden-config.json"
```

The tool is run with the command 'launch ScreenConfigurer', the enterprise Id for which the preference is to be imported has to be provided with the "-e" parameter. The name of the app(context-root) is to be supplied with the argument "-n" and the filepath is to be supplied with the argument "-fn". The file path is resolved according to the specifications as described in the [PathMatchingResourcePatternResolver](#) of the spring core. Please refer to the link for more details. In the above mentioned case as only the name of the file is specified, the scope of the search for the file is restricted to the classpath of the application.

5. Screen configuration JSON creation helper:

A Chrome extension is also being provided to help in the creation of the configuration JSON for any entry based screen and details page. The following steps must be followed for using the extension:

1. Install [this Chrome extension](#) locally to your machine. The Chrome version should be above 23.
2. Go to the first page of the entry screen or details screen. Click on the chrome extension to start the creation of the JSON configuration file.
3. Navigate to every next screen one after another until you reach to the final page of the wizard.
4. After reaching to the end, click the extension icon again to indicate the end of the recording. A new tab will open with the JSON configuration as recorded from each of the screens which were navigated.
5. In the case of a detail screen being present on the screen at the time of the recording, the configuration relating to it will also be recorded simultaneously.
6. For a single page detail screen, as there is no need of navigation hence the chrome extension button can be clicked twice simultaneously to indicate the start and the stop of the recording.
7. Whenever the recording will be stopped, the configuration recorded till that moment will be displayed in the new tab.
8. Only a single screen(for example only TRDEN/TRDCP/TRDAM, but not all) must be considered for recording at a time. The configuration related to a screen must be recorded in a single flow without escaping from it to any other screen while the configuration is being recorded.
9. The JSON configuration created consists of the names of all the relevant fields in a screen which can be considered for customization. The user may remove fields from the list in the JSON as per his requirement and feed the final JSON to the command line tool for saving the configuration.

The images below shows the flow while recording the configuration of trade copy screen(TRDEN) :

localhost:8080/sample-web/

Search Menu

TRDEN:Trade Entry [20140503]

1 Trade General 2 3

Click on the screen configuration creator chrome extension

Trade Type	EQ	Trade Date	20150101	Value Date	20150101	Security Code	CISCO
Quantity	23	Price	12233	Account Balance Type			
Buy/Sell Orientation	Client Bu	Account No	124456892/F	Inventory Account No	1501020/F	Execution Market	HSBC
Trade Currency	JPY	Settlement Currency	JPY				

SAVE TEMPLATE RESET NEXT

The user is notified that the configuration creation has started

Screen Configurator Recording has started

© 2013

Search Menu FIN 20140503 Logout

TRDEN: Trade Entry [20140503]

1 **2** **3 Trade Details**

Trade Type	EQ	Trade Date	20150101	Value Date	20150101	Buy/Sell Orientation	AB
Account No	124456892/F	Inventory Account No	1501020/F	Security Code	CISCO	Quantity	23
Price	12233	Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY
Account Balance Type							

Tax Fee

+ <input type="button"/>	ID	Rate	Rate Type	Amount

RR

RR Role	RR Number	Commission Amount
Executing RR	<input type="text"/>	<input type="text"/>
RR	<input type="text"/>	<input type="text"/>
Split RR1	<input type="text"/>	<input type="text"/>
Split RR2	<input type="text"/>	<input type="text"/>
Trader	<input type="text"/>	<input type="text"/>

Net Amount External Reference No

© 2012 NB Financial Technology, INDIA. Powered By Finch (v 2.3.1.0) (upnp)

localhost:8080/sample-web/

Search Menu FIN 20140303

TRDEN:Trade Entry [20140503]

1 Trade General 2 Trade View 3 Trade Details

Click again on the extension icon to stop recording

Trade Type	EQ	Inventory Account No	1501020	Inventory A/C Short Name		Account Balance Type
Account No	124456892	Short Name		Security Code	CISCO	Quantity
Price	12233	Trade Date	20150101	Value Date	20150101	Buy/Sell Orientation
Execution Market	B001	Trade Currency	JPY	Settlement Currency	JPY	

BACK CONFIRM

localhost:8080/sample-web/# © 2012 NRI Financial Technology, INDIA. Powered By Finch (v 2.3.1.0-SNAPSHOT)

The screenshot shows a browser window with the URL `chrome-extension://olgdpkjecejjlnaoapmbaabppcnhlpo/config.html`. The page displays a JSON configuration object with two main sections, '0' and '2'. Each section contains a 'ScreenConfig' object with various trade-related properties like 'tradeDTO.tradeType', 'tradeDTO.tradeDate', etc. A red arrow points from a callout box labeled 'The recorded JSON configuration displayed in a new tab' to the JSON code. Another red arrow points from a callout box labeled 'The user is notified that the configuration has been recorded from the screen' to a notification message in a separate window titled 'Screen Configurator Recording Completed'. The notification window also features a red icon.

```

{
  "TRDEN": {
    "0": {
      "ScreenConfig": {
        "tradeDTO.tradeType",
        "tradeDTO.tradeDateStr",
        "valueDateStr",
        "trade.inventoryAccount.accountNo",
        "trade.inventoryAccount.shortName",
        "tradeDTO.accountBalanceTypeDisplay",
        "trade.account.accountNo",
        "trade.account.shortName",
        "tradeDTO.securityId",
        "tradeDTO.quantity",
        "tradeDTO.price",
        "tradeDTO.accountBalanceTypeStr",
        "tradeDTO.tradeDate",
        "tradeDTO.valueDate",
        "tradeDTO.buySellOrientation",
        "tradeDTO.accountNo",
        "tradeDTO.account.shortName",
        "tradeDTO.securityId",
        "tradeDTO.quantity",
        "tradeDTO.price",
        "tradeDTO.executionMarket",
        "tradeDTO.tradeCurrency",
        "tradeDTO.settlementCurrency"
      }
    },
    "2": {
      "ScreenConfig": {
        "tradeDTO.tradeType",
        "tradeDTO.tradeDate",
        "tradeDTO.valueDate",
        "tradeDTO.buySellOrientation",
        "tradeDTO.accountNo",
        "tradeDTO.account.shortName",
        "tradeDTO.securityId",
        "tradeDTO.quantity",
        "tradeDTO.price",
        "tradeDTO.executionMarket",
        "tradeDTO.tradeCurrency",
        "tradeDTO.settlementCurrency",
        "tradeDTO.accountBalanceType",
        "rMap1": "R1\r\nlNumber",
        "rMap2": "Split R1\r\nlNumber",
        "rMap3": "Split R2\r\nlNumber",
        "rMap4": "Trade\r\nlNumber",
        "tradeDTO.netAmount",
        "tradeDTO.externalReferenceNo"
      }
    }
  }
}

```

6. Sample implementation:

The new code blocks for supporting customization in trade module screens are shown below.

6.1 Entry based wizard pages:

tradeDetailsEntry.jspx
› Expand

```

<div class="detailBlock entrySingleCol">

  <div class="formItemBlock dottedBg">
    <div class="formItem">

```

```

<label><spring:message
    code="trd.tradeentryaction.label.tradetype" htmlEscape="false" /></label>
<span class="detailDisplay"
name="tradeDTO.tradeType">${commandForm.tradeDTO.tradeType}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.tradedate"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.tradeDate">${commandForm.tradeDTO.tradeDate}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.valuedate" htmlEscape="false" /></label> <span
    class="detailDisplay" name="tradeDTO.valueDate">${commandForm.tradeDTO.valueDate}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.buysellorientation"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.buySellOrientation">${commandForm.tradeDTO.buySellOrientation}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.accountno" htmlEscape="false" /></label>
<span class="detailDisplay"
name="tradeDTO.accountNo">${commandForm.tradeDTO.accountNo}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message code="trd.tradeentryaction.label.inventoryaccountno"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="trade.account.shortName">${commandForm.tradeDTO.inventoryAccountNo}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message code="trade.label.security.code"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.securityId">${commandForm.tradeDTO.securityId}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.quantity" htmlEscape="false" /></label> <span
    class="detailDisplay" name="tradeDTO.quantity">${commandForm.tradeDTO.quantity}</span>
<div class="clear"></div>
</div>

```

```

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.price" htmlEscape="false" /></label> <span
    class="detailDisplay" name="tradeDTO.price">${commandForm.tradeDTO.price}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.executionmarket" htmlEscape="false" /></label>
<span class="detailDisplay"
name="tradeDTO.executionMarket">${commandForm.tradeDTO.executionMarket}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.tradecurrency" htmlEscape="false" /></label>
<span class="detailDisplay"
name="tradeDTO.tradeCurrency">${commandForm.tradeDTO.tradeCurrency}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.settlementcurrency"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.settlementCurrency">${commandForm.tradeDTO.settlementCurrency}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
    code="trd.tradeentryaction.label.accbalancetype"
    htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.accountBalanceType">${commandForm.tradeDTO.accountBalanceTypeStr}</span>
<div class="clear"></div>
</div>

<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
<div class="lineHide">

```

```
<spring:message text="" htmlEscape="false" />
</div>
</div>
```

tradeGeneralDetail.jspx

› Expand

```
<div class="entryBlkArea">
<div class="formItemBlock dottedBg">

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.tradetype" htmlEscape="false" /></label>
<span class="detailDisplay" name="tradeDTO.tradeType">${commandForm.tradeDTO.tradeType}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.inventoryaccountno"
htmlEscape="false" /></label> <span class="detailDisplay"
name="trade.inventoryAccount.accountNo">${commandForm.trade.inventoryAccount.accountNo}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trade.label.inventory.shortname" htmlEscape="false" /></label> <span
class="detailDisplay"
name="trade.inventoryAccount.shortName">${commandForm.trade.inventoryAccount.shortName}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.accbalancetype"
htmlEscape="false" /></label> <span class="detailDisplay"

name="tradeDTO.accountBalanceTypeDisplay">${commandForm.tradeDTO.accountBalanceTypeDisplay}</sp
an>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.accountno" htmlEscape="false" /></label>
<span class="detailDisplay"
name="trade.account.accountNo">${commandForm.trade.account.accountNo}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message code="trade.label.short.name"
htmlEscape="false" /></label> <span class="detailDisplay"
name="trade.account.shortName">${commandForm.trade.account.shortName}</span>
<div class="clear"></div>
</div>
```

source

```
<div class="formItem">
<label><spring:message code="trade.label.security.code"
htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.securityId">${commandForm.tradeDTO.securityId}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.quantity" htmlEscape="false" /></label> <span
class="detailDisplay" name="tradeDTO.quantity">${commandForm.tradeDTO.quantity}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.price" htmlEscape="false" /></label> <span
class="detailDisplay" name="tradeDTO.price">${commandForm.tradeDTO.price}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.tradedate" htmlEscape="false" /></label>
<span class="detailDisplay" name="tradeDTO.tradeDate">${commandForm.tradeDTO.tradeDate}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.valuedate" htmlEscape="false" /></label>
<span class="detailDisplay" name="tradeDTO.valueDate">${commandForm.tradeDTO.valueDate}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.buySellorientation"
htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.buySellOrientation">${commandForm.tradeDTO.buySellOrientation}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.executionmarket"
htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.executionMarket">${commandForm.tradeDTO.executionMarket}</span>
<div class="clear"></div>
</div>

<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.tradecurrency" htmlEscape="false" /></label>
<span class="detailDisplay"
name="tradeDTO.tradeCurrency">${commandForm.tradeDTO.tradeCurrency}</span>
<div class="clear"></div>
</div>
```

```
<div class="formItem">
<label><spring:message
code="trd.tradeentryaction.label.settlementcurrency"
htmlEscape="false" /></label> <span class="detailDisplay"
name="tradeDTO.settlementCurrency">${commandForm.tradeDTO.settlementCurrency}</span>
<div class="clear"></div>
</div>

<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
<div class="lineHide">
```

```
<spring:message text="" htmlEscape="false" />
</div>
</div>
```

tradeDetailsParticular.jspx

› [Expand](#)

```
<div class="entryBlkArea">
  <div class="formItemBlock dottedBg">
    <div class="formItem">
      <label><spring:message
        code="trd.tradeentryaction.label.netamount" htmlEscape="false" /></label>
      <span class="detailDisplay" name="tradeDTO.netAmount">${commandForm.tradeDTO.netAmount}</span>
      <div class="clear"></div>
    </div>

    <div class="formItem">
      <label><spring:message
        code="trd.tradeentryaction.label.externalreferenceno"
        htmlEscape="false" /></label> <span class="detailDisplay"
        name="tradeDTO.externalReferenceNo">${commandForm.tradeDTO.externalReferenceNo}</span>
      <div class="clear"></div>
    </div>
    <div class="clear">
      <spring:message text="" htmlEscape="false" />
    </div>
    </div>
    <div class="lineHide">
      <spring:message text="" htmlEscape="false" />
    </div>
  </div>
</div>
```

[source](#)

6.2 Details page:

tradeDetailView.jspx

› [Expand](#)

```
<table class="detailBlock3Col">
  <tbody>
    <tr>
      <td>
        <table class="tableStyle">
          <tbody>
            <tr>
              <td>
                <table class="detailBlock detailSmlBlkWidth">
                  <tbody>
                    <tr>
                      <td class="detailSmlBrdPad">
                        <h1>
                          <spring:message code="trade.label.counterpartyaccount"
                            htmlEscape="false" />
                        </h1>

```

[source](#)

```

<div class="detallItemBlock">
<div class="detallItem">
<label>${data.counterPartyLabel}</label> <span
name="counterPartyCode">${data.counterPartyCode}</span>
<div class="clear"></div>
</div>
<div class="detallItem">
<label><spring:message
code="trade.label.account.no" htmlEscape="false" /></label>
<span name="accountNo">${data.accountNo}</span>
<div class="clear"></div>
</div>
<div class="detallItem">
<label><spring:message
code="trade.label.short.name" htmlEscape="false" /></label>
<span name="counterPartyName">${data.counterPartyName}</span>
<div class="clear"></div>
</div>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.tradereference"
htmlEscape="false" />
</h1>
<div class="detallItemBlock">
<div class="detallItem">
<label><spring:message
code="trade.label.reference.no" htmlEscape="false" /></label>
<span name="tradeReferenceNo">${data.tradeReferenceNo} - ${data.versionNo}</span>
<div class="clear"></div>
</div>
<div class="detallItem">
<label><spring:message
code="trade.label.trade.status" htmlEscape="false" /></label>
<span name="status">${data.status}</span>
<div class="clear"></div>
</div>
<div class="detallItem">
<label><spring:message
code="trade.label.external.Refno" htmlEscape="false" /></label>
<span name="externalReferenceNo">${data.externalReferenceNo}</span>
<div class="clear"></div>
</div>
<div class="detallItem">
<label><spring:message

```

```

        code="trade.label.data.source" htmlEscape="false" /></label>
        <span name="dataSource">${data.dataSource}</span>
        <div class="clear"></div>
        </div>
        <div class="clear">
            <spring:message text="" htmlEscape="false" />
        </div>
        </div>
        </td>
    </tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
    <tbody>
        <tr>
            <td class="detailSmlBrdPad">
                <h1>
                    <spring:message code="trade.label.currency"
                        htmlEscape="false" />
                </h1>
                <div class="detaillItemBlock">
                    <div class="detaillItem">
                        <label><spring:message
                            code="trade.label.trade.ccy" htmlEscape="false" /></label>
                        <span name="tradeCcy">${data.tradeCcy}</span>
                        <div class="clear"></div>
                    </div>
                    <div class="detaillItem">
                        <label><spring:message
                            code="trade.label.settlement.ccy" htmlEscape="false" /></label>
                        <span name="settlementCcy">${data.settlementCcy}</span>
                        <div class="clear"></div>
                    </div>
                    <div class="detaillItem">
                        <label><spring:message
                            code="trade.label.execution.market"
                            htmlEscape="false" /></label> <span name="executionMarket">${data.executionMarket}</span>
                        <div class="clear"></div>
                    </div>
                    <div class="clear">
                        <spring:message text="" htmlEscape="false" />
                    </div>
                </div>
            </td>
        </tr>
    </tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
<td class="detailBlockMiddlePad">
<table class="tableStyle">
    <tbody>

```

```

<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.inventoryaccount"
htmlEscape="false" />
</h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message
code="trade.label.inventory.accno"
htmlEscape="false" /></label> <span
name="inventoryAccountNo">${data.inventoryAccountNo}</span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message
code="trade.label.inventory.shortname"
htmlEscape="false" /></label> <span
name="inventoryAccountName">${data.inventoryAccountName}</span>
<div class="clear"></div>
</div>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.basictradeattributes"
htmlEscape="false" />
</h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message
code="trade.label.trade.type" htmlEscape="false" /></label>
<span name="tradeType">${data.tradeType}</span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message
code="trade.label.buySell.orientation"
htmlEscape="false" /></label> <span
name="buySellOrientation">${data.buySellOrientation}</span>
<div class="clear"></div>
</div>

```

```
<div class="detailItem">
<label><spring:message
    code="trade.label.principleagent.flag"
    htmlEscape="false" /></label> <span
    name="principalAgentFlag">${data.principalAgentFlag}</span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message
    code="trade.label.account.balancetype"
    htmlEscape="false" /></label>
<c:choose>
<c:when test="${data.accountBalanceType > 0}">
<fmt:formatNumber minIntegerDigits="2"
    value="${data.accountBalanceType}" />
</c:when>
<c:otherwise>${data.accountBalanceType}</c:otherwise>
</c:choose>
<div class="clear"></div>
</div>
<div class="clear">
    <spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
    <spring:message code="trade.label.price"
        htmlEscape="false" />
</h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message
    code="trade.label.price" htmlEscape="false" /></label> <span
    name="price">${data.price}</span>
<div class="clear"></div>
</div>
<div class="clear">
    <spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
```

```
</td>
<td>
<table class="tableStyle">
<tbody>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.securityinformation"
htmlEscape="false" />
</h1>
<div class="detaillItemBlock">
<div class="detaillItem">
<label><spring:message
code="trade.label.instrument.type"
htmlEscape="false" /></label> <span name="instrumentType">${data.instrumentType}</span>
<div class="clear"></div>
</div>
<div class="detaillItem">
<label><spring:message
code="trade.label.security.code" htmlEscape="false" /></label>
<span name="securityCode">${data.securityCode}</span>
<div class="clear"></div>
</div>
<div class="detaillItem">
<label><spring:message
code="trade.label.security.name" htmlEscape="false" /></label>
<span name="instrumentName">${data.instrumentName}</span>
<div class="clear"></div>
</div>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.date"
htmlEscape="false" />
</h1>
<div class="detaillItemBlock">
<div class="detaillItem">
<label><spring:message
code="trade.label.trade.date" htmlEscape="false" /></label>
<span name="tradeDate">${data.tradeDate}</span>
<div class="clear"></div>
```

```
</div>
<div class="detailItem">
<label><spring:message
    code="trade.label.value.date" htmlEscape="false" /></label>
<span name="valueDate">${data.valueDate}</span>
<div class="clear"></div>
</div>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<table class="detailBlock detailSmlBlkWidth">
<tbody>
<tr>
<td class="detailSmlBrdPad">
<h1>
<spring:message code="trade.label.volume"
    htmlEscape="false" />
</h1>
<div class="detailItemBlock">
<div class="detailItem">
<label><spring:message
    code="trade.label.quantity" htmlEscape="false" /></label> <span
    name="quantity">${data.quantity}</span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message
    code="trade.label.net.amount" htmlEscape="false" /></label>
<span name="netAmount">${data.netAmount}</span>
<div class="clear"></div>
</div>
<div class="detailItem">
<label><spring:message
    code="trade.label.netamount.trdccy"
    htmlEscape="false" /></label> <span
    name="netAmountInTrdCcy">${data.netAmountInTrdCcy}</span>
<div class="clear"></div>
</div>
<div class="clear">
<spring:message text="" htmlEscape="false" />
</div>
</div>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
```

</td>

```
</tr>
</tbody>
</table>
```

3.4.6.3 Console Features

Introduction

This section describes about the following console related features:

The root page Console Features could not be found in space finch 2.

3.4.6.3.1 Service Process

Introduction

This page describes how to develop a console service process using finch service process abstraction.

Background

finch provides an abstraction for developing console based Service or daemon processes. finch abstraction takes care of life cycle management of service. When a service is started, it first register itself with service registry. The service registry is maintained by a console process called ServiceController. When a service process is stopped it deregister itself from the service registry. The registration and deregistration functions are implemented by finch service abstraction.

We are using Apache Camel for the Service Process in finch. Click here to learn [Apache Camel](#)

Role of finch

- 1) finch provides the bootstrapping of the camel context.
- 2) The abstraction provided by finch has proper exception handling for camel services.
- 3) finch provides few pre-configured message processors that helps to process messages with minimal configuration in database.

Overview of Camel Development Process

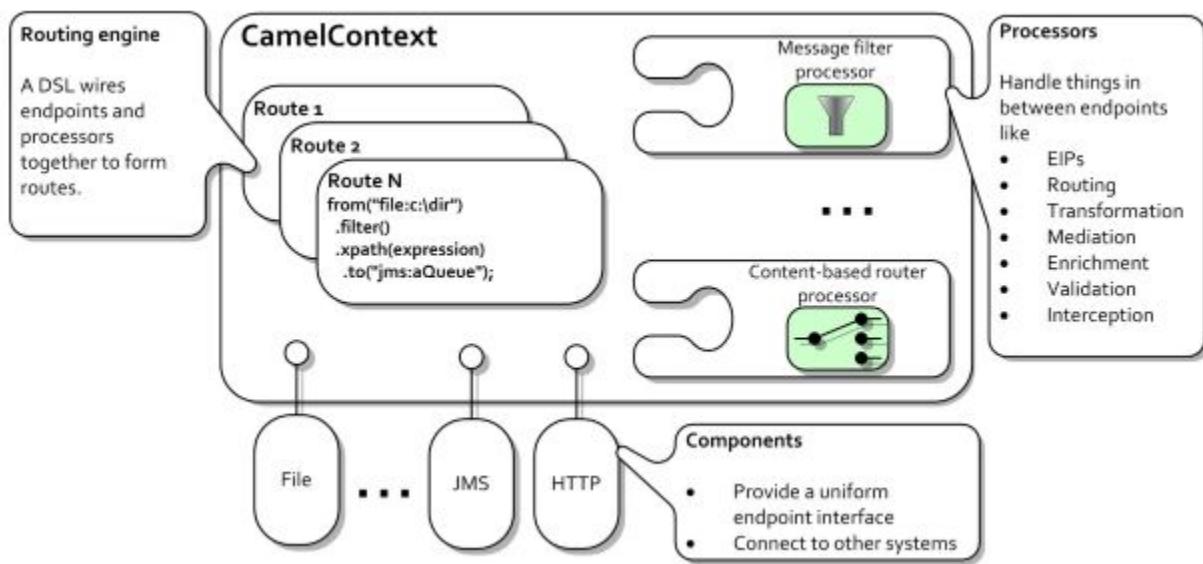
This section gives some overview about camel and development process

Camel Overview

Apache Camel is an open source Java framework that focuses on making integration easier and more accessible to developers. It does this by providing:

- concrete implementations of all the widely used EIPs
- connectivity to a great variety of transports and APIs
- easy to use Domain Specific Languages (DSLs) to wire EIPs and transports together.

Following figure shows how these three items actually map to Camel concepts.



Following are the summary of the key concepts of Apache camel

- Camel Context: Runtime system that keeps all pieces together.
- Message: Fundamental entity – core principle of messaging. It consists of Headers, Body and Attachments.
- Exchange: Container for messages (abstraction of what is actually sent over the system). It contains In and optionally also Out message.
- Route: Chain of processors. If you like more “academic” explanation then route is a graph, where “node” is represented by some Processor and “line” is represented by some Channel.
- Processor: Uses/modifies incoming exchange. Output of one processor is connected to the input of another one.
- Endpoint: Models end of channel. Configured using URI. Example:[jms://INF.inq](#)
- Component: Factory for endpoints. Referred with prefixes (jms:, file:, etc.). There are many more components are there check this [link](#) for list of components.

Camel defines multiple DSLs in regular programming languages such as Java, Scala and Groovy. Example as below

- Java DSL

JAVA DSL

```
from("<SOURCE_URI>").to("jms:QueueName");
```

- Spring DSL

Spring DSL

```
<route>
<from uri="<SOURCE_URI>"/>
<to uri="jms:QueueName"/>
</route>
```

- Scala DSL

Scala DSL

```
from "<SOURCE_URI>" -> "jms:QueueName"
```

Development steps using Camel

Following are the basic development steps using Apache Camel and ActiveMQ

1. Download activemq.
2. Extract activemq.zip file and start ActiveMQ (Click on activemq.bat file).
3. Check activemq console using <http://localhost:8161/>.
4. Create a JMS Message Producer to send Messages.
5. After Creating message producer check the Queue from the link in step 3.
6. Create a Camel Router and start Camel context to Route Messages.
7. Handle all sorts of exception raised during the routing of messages.

The Above are the basic steps to learn configuration for Spring please [click here](#)

Development steps using finch

Following are the basic development steps using finch

1. finch provides a default JMS Message Producer to send messages based on destination point.
2. finch provides an abstraction over route builder to route the messages easily. The application just need to create the camel router. The starting the camel context is taken care from the finch end.
3. Basic Exception raised on routing the message has been handled in finch. Hence only custom exception should be handled by the application.
4. Application need to refer the bean in the jms context xml file.

The Above are the basic steps to learn configuration for Spring please [click here](#)

Configurations for Service Process in finch

Database Setup

Following tables are used for Service Process.

INF_ENDPOINT - Contains ENDPOINT_URI the pattern should be Apache camel complied URI pattern. For More about URI please [click here](#)

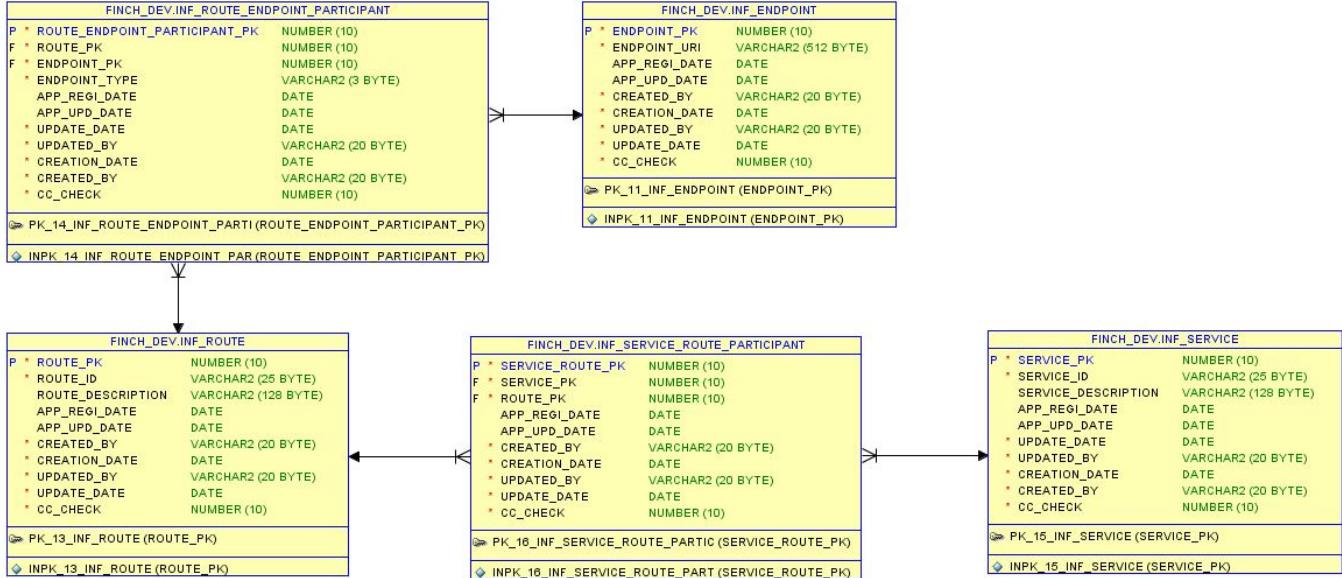
INF_ROUTE - Contains ROUTE_ID for Service Process. Click [here](#) to know about [Camel Route](#)

INF_ROUTE_ENDPOINT_PARTICIPANT - This table is used for relationship between Route and Endpoint.

INF_SERVICE - This Table contains the Service Id for message processor service,message router service etc.

INF_SERVICE_ROUTE_PARTICIPANT - This table describes the relationship between Service and Route.

Following is the ER-Diagram of the above Tables



Common Configurations

- Add eai-context.xml for ActiveMQ. Below is the example

```

eai-context.xml

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
                           http://camel.apache.org/schema/spring
                           http://camel.apache.org/schema/spring/camel-spring.xsd">
    <!-- import the ActiveMQ broker to act as server -->
    <!-- import resource="activemq-broker.xml"/-->
    <import resource="jms-context.xml"/>

    <bean id="activemq"
          class="org.apache.activemq.camel.component.ActiveMQComponent">
        <property name="brokerURL" value="tcp://localhost:61616"/>
    </bean>
    <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
        <package>com.nrft.finch.inf.console.utility</package>
    </camelContext>
</beans>

```

- Add jms-context.xml and include Camel Route Builders.Example as below

```

Camel Route Builders

<!-- Sample Dist console Config -->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context">

```

```

xmlns:p="http://www.springframework.org/schema/p"
xmlns:util="http://www.springframework.org/schema/util"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.2.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util-3.2.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">
<context:component-scan base-package="com.nrft.finch.inf.msg" />

<!--context:property-placeholder location="classpath:META-INF/spring/jms.properties"/ -->
<bean id="jmsProperties"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>classpath:META-INF/config/jms.properties</value>
        </list>
    </property>
    <property name="ignoreUnresolvablePlaceholders" value="true" />
</bean>
<!-- bean id="MessageImportFileSplitter"
class="com.nrft.finch.inf.console.utility.JSONMessageImportFileSplitter" /-->
<bean id="MessageImportFileSplitter"
class="com.nrft.finch.inf.console.utility.XMLMessageImportFileSplitter" />
<bean id="jmsConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">
    <property name="brokerURL" value="${activemq.broker.uri}" />
    <property name="redeliveryPolicy">
        <bean class="org.apache.activemq.RedeliveryPolicy">
            <property name="maximumRedeliveries" value="${activemq.maxRedeliveries}" />
        </bean>
    </property>
</bean>
<bean id="cachingConnectionFactory"
class="org.springframework.jms.connection.CachingConnectionFactory" destroy-method="destroy">
    <property name="targetConnectionFactory" ref="jmsConnectionFactory" />
    <property name="reconnectOnException" value="true" />
</bean>
<bean id="LogicalQueueResolver" class="com.nrft.finch.inf.msg.LogicalQueueResolver" />
<bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">
    <property name="connectionFactory" ref="cachingConnectionFactory" />
    <property name="sessionTransacted" value="true" />
    <property name="destinationResolver" ref="LogicalQueueResolver" />
</bean>
<!--bean id="messageRoutingRuleFinder"
class="com.nrft.finch.inf.rule.TestMessageRoutingRuleFinder"/ -->
<!-- bean id="textMessageFormatHandler" class="com.nrft.finch.inf.msg.JSONMessageFormatHandler"
/-->
<bean id="textMessageFormatHandler" class="com.nrft.finch.inf.msg.XMLMessageFormatHandler" />
<!-- Components -->
<!-- In JMS component, the cache level should be explicitly set to CACHE_AUTO
for pre 2.8.0 versions of Camel. The default setting is CACHE_CONSUMER. -->
<bean id="jmsComponent" class="org.apache.camel.component.jms.JmsComponent">
    <property name="connectionFactory" ref="cachingConnectionFactory" />
    <property name="cacheLevelName" value="CACHE_AUTO" />
    <property name="concurrentConsumers" value="${jms.concurrentConsumers}" />
    <property name="maxConcurrentConsumers" value="${jms.maxConcurrentConsumers}" />

```

```
<property name="transacted" value="true" />
<property name="transactionManager" ref="global-tm" />
<property name="destinationResolver" ref="LogicalQueueResolver" />
</bean>
<!-- Camel Route Builders -->
<bean id="messagerouter" class="com.nrft.finchn.inf.msg.routebuilder.MessageRouterRouteBuilder" />

<!-- Camel Processor Builder -->
<bean id="messageprocessor"
class="com.nrft.sample.trd.msg.routebuilder.TrdMessageProcessorRouteBuilder" />

<!-- Camel exception handlers -->
<bean id="messageRouterDefaultExceptionHanlder"
class="com.nrft.finchn.inf.msg.exceptionhandler.MessageRouterDefaultExceptionHandler" />
<!-- Camel processors -->
<bean id="msgRouterPostReceiveProcessor"
class="com.nrft.finchn.inf.msg.processor.DefaultMessageRouterPostReceiveProcessor" />
<bean id="msgRouterPreSendProcessor"
class="com.nrft.finchn.inf.msg.processor.DefaultMessageRouterPreSendProcessor" />
<bean id="errorPersistingExceptionHandler"
class="com.nrft.finchn.inf.msg.exceptionhandler.ErrorPersistingExceptionHandler" />
<bean id="errorMessageProcessor" class="com.nrft.finchn.inf.msg.processor.ErrorMessageProcessor" />
<!-- bean id="TRD01Processor" class="com.nrft.sample.trd.msg.processor.TRD01JSONProcessor" /-->
<bean id="TRD01Processor" class="com.nrft.sample.trd.msg.processor.TRD01XMLProcessor" />
<bean id="unhandledMessageProcessor"
class="com.nrft.finchn.inf.msg.processor.UnhandledMessageProcessor" />
<!-- bean id="msgObjectToStringConverter"
class="com.nrft.finchn.inf.msg.processor.JSONObjectToStringConverter" /-->
<bean id="msgObjectToStringConverter"
class="com.nrft.finchn.inf.msg.processor.XMLObjectToStringConverter" />
<bean id="enterpriseldValidator" class="com.nrft.finchn.inf.msg.processor.EnterpriseldValidator" />
<bean id="fatalExceptionHandler"
class="com.nrft.finchn.inf.msg.exceptionhandler.FatalExceptionHandler" />
```

```
<!-- Camel dynamic router -->
<bean id="dynamicRouter" class="com.nrft.fin.ch.inf.msg.dynamicrouter.MessageDynamicRouter" />
</beans>
```

3. Add jms.properties and Include queue mappings. Here three queues are defined IN,OUT and ERROR. Following is the example

Queue Mapping in jms.properties

```
# JMS
jms.concurrentConsumers = 1
jms.maxConcurrentConsumers = 1
# ACTIVE MQ
activemq.broker.uri=tcp://localhost:61616?daemon=true
activemq.maxRedeliveries = -1
#queue mapping
queue.mapping.INF.INQ=INFIN
queue.mapping.INF.OUTQ=INFOOUT
queue.mapping.INF.ERRQ=INFERR
# sample TRD queue mapping
queue.mapping.TRD.inq=TRDIN
queue.mapping.TRD.outq=TRDOUT
queue.mapping.TRD.errq=TRDERR
queue.mapping.NTF.inq=NTFIN
```

4. Need to include eai-context.xml and jms-context.xml in application-context.xml as below

Include jms-context and eai-context

```
<import resource="classpath:META-INF/config/eai-context.xml"/>
<import resource="classpath:META-INF/config/jms-context.xml"/>
```

5. Create <MessageProcessorRouteBuilder> java class which extends AbstractRouteBuilder of finch which override doConfig() method. Below is the example

MessageProcessorRouteBuilder Java Class

```
package com.nrft.fin.ch.inf.msg.route;

import java.util.List;
import org.apache.camel.spi.Policy;
import org.apache.camel.spring.spi.SpringTransactionPolicy;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.transaction.PlatformTransactionManager;
import com.google.common.base.Optional;
import com.google.common.collect.ImmutableList;
import com.google.common.collect.ImmutableMap;
import com.nrft.fin.ch.inf.domain.entity.Route;
import com.nrft.fin.ch.inf.exception.FinchException;
import com.nrft.fin.ch.inf.exception.FinchRuntimeException;
import com.nrft.fin.ch.inf.msg.MessageConstants;
import com.nrft.fin.ch.inf.msg.TrdException;
```

```

import com.nrft.finch.inf.msg.routebuilder.AbstractRouteBuilder;

/**
 * RouteBuilder for sample TRD component.
 */
public class TrdMessageProcessorRouteBuilder extends AbstractRouteBuilder {
 /**
 * The <code>LOGGER</code> instance for this class.
 */
private static final Log log = LogFactory.getLog(TrdMessageProcessorRouteBuilder.class);
@Override
public void doConfig() throws FinchException {

    Policy required = new SpringTransactionPolicy(lookup("global-tm",
PlatformTransactionManager.class));
    Optional<String> srcURI = getSourceEndpointURI();
    if (!srcURI.isPresent()) {
        log.error("No Source endpoint defined for the route [" + getRoute().getRouteld() + "]");
        throw new TrdException("error.msg.endpoint.notfound", "Source");
    }

    Optional<String> errURI = getErrorEndpointURI();
    if (!errURI.isPresent()) {
        log.error("No error endpoint defined for the route [" + getRoute().getRouteld() + "]");
        throw new TrdException("error.msg.endpoint.notfound", "Error");
    }

    Optional<String> destURI = getDestinationEndpointURI();
    if (!destURI.isPresent()) {
        log.error("No destination endpoint defined for the route [" + getRoute().getRouteld() + "]");
        throw new TrdException("error.msg.endpoint.notfound", "Destination");
    }

    onException(FinchRuntimeException.class).processRef("fatalExceptionHandler");

    onException(Exception.class).processRef("errorPersistingExceptionHandler");
    from(srcURI.get())
        .routeld(getRoute().getRouteld())
        .policy(required)
        .processRef("enterpriseIdValidator")
        .choice()
            .when().method("errorMessageProcessor", "isErrorMessage")
                .processRef("errorMessageProcessor")
                .to(errURI.get())
            .when(header(MessageConstants.JMS_PROP_MESSAGE_TYPE).isNotEqualTo("TRD01"))
                .processRef("unhandledMessageProcessor")
            .otherwise()
                .processRef("TRD01Processor").split(body()).processRef("msgObjectToStringConverter")
                .to(destURI.get());
    }

    private static final ImmutableMap<String, ImmutableList<String>> endPoints = ImmutableMap.of("IN",
ImmutableList.of("jms://INF.INQ"),
                    "OUT", ImmutableList.of("jms://INF.OUTQ"),
                    "ERR", ImmutableList.of("jms://INF.ERRQ"));

    protected List<String> getEndpointsByType(String type) throws FinchException {
        return endPoints.get(type);
    }
}

```

```
public Route getRoute() {  
    Route r = new Route();  
    r.setRouteId("messageprocessor");
```

```
    return r;
}
}
```

Note:To see how to run message processor in sample app please [click here](#)

3.4.6.3.2 Batch Process

1. Introduction

finch batch framework is used to create a batch Job with some abstraction for Spring Batch framework.

2. Common Setup to develop a Batch Process

The batch developing process has been described in the following section.

Summary

If the requirement is to run only console base job then have to follow steps - 3,4,5, and 6.

If the requirement is to run only UI base job then have to follow steps - 3,4,5, and 7. and for query screen see chapter-8

2.1 Create a Java Class for Batch Metadata

1. The class representing the batch-job must extend com.nrft.fin.ch.inf.console.batch.ConsoleMetaData and must be annotated with com.nrft.fin.ch.inf.batch.metadata.Batch to avail the finch support for batch-operations.

```
com.nrft.fin.ch.inf.batch.metadata.Batch annotations has four properties
1. id - should be same with the batch id
2. authenticate - It accepts true/false.If true then authentication enabled else false means authentication disabled
3. register - It accepts true/false.If true the it register with ServiceController.If false then no need to register with
ServiceController
4. componentId - It takes application component Id.
```

2. finch uses args4j library for command line parameters. Add batch specific command line parameters parsing using @Option annotation of args4j library in the batch class. For example,

```
@Option(name="-q")
private String queueName
```

Please note that following option names are used by finch console classes and Application should NOT use those option names:

-h -- Option to print help or usage message

-e -- Option for enterprise Ids

Application client can configure the showUsage() which is required to display the switches while running the batch if any required parameter is missing. finch generally display all the usage options (switches) when any required parameter is missing while running the batch.

SampleDBtoCSVBatch.java

```
@SuppressWarnings("unchecked")
@Override
public Optional<String> showUsage() {
    ResourceBundle rb = null;
    rb = new ExtendedBundleDecorator(CONSOLE_OPTION_USAGE_RESOURCE, I18NUtils.getLocale());
    StringBuilder showUsages = new StringBuilder();
    showUsages.append("Showing Usages\n\n");
    Set<Field> fields = ReflectionUtils.getAllFields(this.getClass());
    for (Field field : fields) {
        if (field.isAnnotationPresent(Option.class)) {
            Option opt = field.getAnnotation(Option.class);
            boolean required = opt.required();
            if (required) {
                showUsages.append("\$t" + opt.name() + " " + field.getType().getSimpleName() + "\$t: "
                    + rb.getString(opt.usage()) + "\n");
            }
        }
    }
    Optional<String> usages = Optional.of(showUsages.toString());
    return usages;
}
```

3. Following needs to be annotated to use the Dropdown, checkbox etc. on the Batch UI page. For example,

```
@Option(name="-f", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown)
private String file;
```

The dropdown generated in Batch UI can be configured in various ways:

a) From Constraint : The dropdown value can be fetched from constraint table.The client has to configure the following:

DemoBatch.java

```
@Option(name="-f", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,userConstraintName="<YOU
R_CONSTRAINT_NAME>")
private String file;
```

b) From Static Value : The dropdown value can be fetched from a class that returns a map consisting of key value pairs.The client has to configure the following:

DemoBatch.java

```
@Option(name="-fn", usage="console.usage.filename.fn")
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,isDisplayAble=true,generator
=SampleGenerator.class)
private String file;
```

This is a sample code which shows the implementation for SampleGenerator.java

SampleGenerator.java

```
public class SampleGenerator implements DropDownDataGenerator {  
    /**  
     * The <code>Log</code> instance for this class.  
     */  
    private static final Logger log = LoggerFactory.getLogger(SampleGenerator.class);  
    @Override  
    public Map<String, String> getData() throws FinchException{  
        Map<String, String> dummyMap = Maps.newHashMap();  
        dummyMap.put("Key1","value1");  
        return dummyMap;  
    }  
}
```

b) From Dynamic Table : The dropdown value can be fetched from a class that returns a map consisting of key value pairs fetched from a custom table.The client has to configure the following:

DemoBatch.java

```
@Option(name="-fn", usage="console.usage.filename.fn")  
@UiMetadata(displayName="File",displayType=DisplayTypes.Dropdown,isDisplayAble=true,dynamicG  
enerator=SampleDynamicGenerator.class,isDynamicDropdown=true)  
private String file;
```

This is a sample code which shows the implementation for SampleDynamicGenerator.java

SampleDynamicGenerator.java

```
public class SampleDynamicGenerator implements DropDownDataGenerator {  
    /**  
     * The <code>Log</code> instance for this class.  
     */  
    private static final Logger log = LoggerFactory.getLogger(SampleDynamicGenerator.class);  
    @Override  
    public Map<String, String> getData() throws FinchException{  
        TradeService tradeService = Operation.getInstance().getReference(TradeService.class);  
        List<TradeDetailsView> tradeDetails = null;  
        Map<String, String> dummyMap = Maps.newHashMap();  
        try {  
            tradeDetails = tradeService.getAllNormalTrades();  
            for (TradeDetailsView trade : tradeDetails) {  
                dummyMap.put(String.valueOf(trade.getTradePk()), trade.getExternalReferenceNo());  
            }  
        } catch (TrdException e) {  
            log.error("Failed to get the trade details", e);  
            throw new TrdException(e.getKey());  
        }  
        return dummyMap;  
    }  
}
```

Note:
Provide getter and setter for each instance variables defined and Batch class can override these methods.

2.2 Batch Job Exception Handling

Catch ComponentKeyedException or its subtypes or JobExecutionException exceptions or its subtypes for handle the exceptions

3. Common Configurations to Run Batch Job

3.1 Generate Batch Metadata

To generate metadata, configure the meta data processor in the package phase of build process of the application.

Generate Spring Batch metadata

```
....  
<plugin>  
  <groupId>org.codehaus.mojo</groupId>  
  <artifactId>exec-maven-plugin</artifactId>  
  <version>1.2.1</version>  
  <executions>  
    <execution>  
      <phase>package</phase>  
      <goals>  
        <goal>java</goal>  
      </goals>  
      <configuration>  
        <mainClass>com.nrift.finch.inf.batch.scanner.FinchBatchMetaBuilder</mainClass>  
        <arguments>  
          <argument>${project.build.directory}/batch-meta</argument>  
        </arguments>  
      </configuration>  
    </execution>  
  </executions>  
</plugin>  
....
```

3.2 Creating a Batch Job

3.2.1 Batch Job Creation Steps

1. To use finch batch user need to include batch-context.xml in there application.Below is the example

batch-context.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"  
      xmlns:p="http://www.springframework.org/schema/p"  
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
      xsi:schemaLocation="http://www.springframework.org/schema/batch  
                        http://www.springframework.org/schema/batch/spring-batch-2.2.xsd  
                        http://www.springframework.org/schema/beans  
                        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">  
  <bean id="jobRegistry"  
        class="org.springframework.batch.core.configuration.support.MapJobRegistry" />  
  <bean
```

```

class="org.springframework.batch.core.configuration.support.JobRegistryBeanPostProcessor">
<property name="jobRegistry" ref="jobRegistry" />
</bean>

<bean id="executors" class="java.util.concurrent.Executors"
      factory-method="newFixedThreadPool">
<constructor-arg value="10"/>
</bean>

<bean id="taskExecutor" class="org.springframework.scheduling.concurrent.ConcurrentTaskExecutor">
<constructor-arg ref="executors" />
</bean>

<bean id="jobLauncher"
      class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
<property name="jobRepository" ref="jobRepository" />
<property name="taskExecutor" ref="taskExecutor" />
</bean>
<bean id="jobRepository"
      class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean">
<property name="dataSource" ref="GLOBAL" />
<property name="transactionManager" ref="batchTransactionManager" />
<property name="databaseType" value="oracle" />
<property name="isolationLevelForCreate" value="ISOLATION_DEFAULT" />
</bean>
<bean name="processShutdownListener" class="com.nrift.finch.inf.batch.ProcessShutdownListener">
<property name="executor" ref="taskExecutor" />
<property name="eventBus" ref="eventBus" />
<property name="explorer" ref="jobExplorer" />

</bean>
<job id="baseJob" abstract="true"
      xmlns="http://www.springframework.org/schema/batch">
<listeners>
<listener ref="processShutdownListener" />
</listeners>
</job>
<!-- <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
<property name="entityManagerFactory" ref="sample-em" />
<property name="dataSource" ref="SAMPLE" />
</bean> -->
<bean id="batchMetaDataProperties"
      class="org.springframework.beans.factory.config.PropertiesFactoryBean">
<property name="properties">
<props>
<prop key="filePath">classpath: META-INF/batch/*.json</prop>
</props>
</property>
</bean>

<bean id="eventBus" class="com.google.common.eventbus.EventBus">
<constructor-arg value="batchEventBus"/>
</bean>

<bean id="batchTransactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource" ref="GLOBAL"/>
</bean>

```

```
<bean id="jobExplorer"
```

```

class="org.springframework.batch.core.explore.support.JobExplorerFactoryBean"
    p:dataSource-ref="GLOBAL" p:tablePrefix="BATCH_" />
</beans>

```

2. To create a batch job, define in xml, which will be available to Spring Context at run-time. A sample demo has been provided in sample-job.xml in sample-app, with the name sampleDBtoCVSBatch. The name of the job id in xml should be exactly the same as the id attribute specified in the Batch annotation.

Sample Job

```

<job id="sampleDBtoCVSBatch" xmlns="http://www.springframework.org/schema/batch"
restartable="true" parent="baseJob" >
    <step id="step1">
        <tasklet transaction-manager="sample-tm" >
            <chunk reader="itemReader" writer="csvWriter" commit-interval="4" />
        </tasklet>
    </step>
</job>

```

3. As shown in the above code block, each job must inherit from the finch provided baseJob to use the finch infrastructure. If the job does not inherit from baseJob, the finch provided abstraction will not be available.
 4. The configuration for the itemReader, is provided below.

Item Reader

```

<bean id="itemReader" class="org.springframework.batch.item.database.JpaPagingItemReader"
scope="step">
    <property name="entityManagerFactory" ref="sample-em" />
    <property name="queryString" value="select t from Trade t" />
    <property name="pageSize" value="3" />
</bean>

```

The configuration for the csvWriter, is provided below.

CsvWriter Configuration

```

<bean id="csvWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">
    <property name="resource" value="file:target/test-outputs/dbToCSV.csv" />
    <property name="lineAggregator">
        <bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">
            <property name="delimiter" value="," />
            <property name="fieldExtractor">
                <bean class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor">
                    <property name="names"
value="tradePk,accountBalanceType,cancelDate,cancelReferenceNo,dataSource,employeePk,externalReferenc
eNo" />
                </bean>
            </property>
        </bean>
    </property>
</bean>
</bean>

```

3.3 Data Setup for Batch Job

To run a batch job from UI one has to do the following steps :

1. First create the following tables
 - a. INF_BATCH_MASTER
 - b. INF_BATCH_ROLE_PCPT
2. Insert values to the above tables. For Example:

BATCH_NAME	STATUS	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE	UPDATED_BY	UPDATE_DATE
1 test	NORMAL	27-SEP-13 20:28:52	27-SEP-13 20:28:52	FINCH-SAMPLE	27-SEP-13 20:28:52	FINCH-SAMPLE	27-SEP-13 20:28:52
2 sampleDBtoCSVBatch	NORMAL	27-SEP-13 20:28:52	27-SEP-13 20:28:52	FINCH-SAMPLE	27-SEP-13 20:28:52	FINCH-SAMPLE	27-SEP-13 20:28:52
BATCH_ROLE_PCPT_PK		BATCH_NAME	APPL_R...	APP_RE...	APP_UP...	CREATE...	CREATI...
1	1 test		1	27-SEP-...	27-SEP-...	FINCH-S...	27-SEP-...
2	2 sampleDBtoCSVBatch		1	27-SEP-...	27-SEP-...	FINCH-S...	27-SEP-...

3.4 Steps to Use Application Provided Settings for Batch Process

Steps are as follows:

- a. Create app-ext folder in their application (for example, sample-dist/src/main/resources/console/app-ext)
- b. After creating the above folder, create the file with the name console-option-usage-resource.properties.

Important Note

Name of the files must be identical as it is specified above.

- c. After above file is created, insert application specific properties in that file.

4. Steps to Run Batch Runner both for Console and UI

Steps to register batch job is Zookeeper

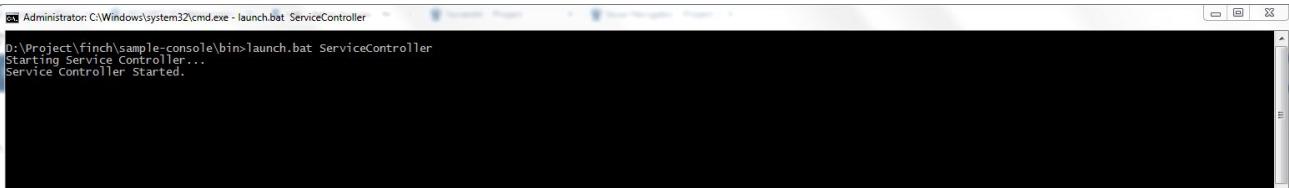
To register the batch in the Zookeeper registry nothing needs to be done. As this is enabled by default. If however an application decides that it does not need any of its batch to register in Zookeeper registry (started with ServiceController) then the developer has to set the property batch.registerInZooKeeper to true in console.properties.

If however the developer needs only some batches to register in Zookeeper, then he needs to specify the attribute register in com.nr.ift.finch.inf.batch.metadata.Batch to false, while leaving the batch.registerInZooKeeper in console.properties as true.

If however batch.registerInZooKeeper is set to false, then finch will not register the batch to Zookeeper even if the register property is set to true.

Steps to run the batch runner are as follows:

1. In Batch annotation if register = true then from console bin run the command to run the ServiceController: launch.bat ServiceController



2. User can update the ports in the yml configuration file provided by finch (batch-rest-config.yml). For more details please see the link <https://dropwizard.github.io/dropwizard/manual/configuration.html>.

5. Running Batch Job using Console

To run a batch job one has to use the following command from the console bin:

Command to run job

```
launch2.bat sampleDBtoCVSBatch -e FIN -f c:\messages.txt
```

finch provides authentication on batch. It can be configurable per batch. Following is the command to run batch for the authenticated user. To use authentication user need to do the above database setup. Once the command is submitted finch first checks if the username and password given match , if yes then it checks if the user has access to the batch by matching the roles configured for the batch in INF_BATCH_ROLE_PCPT.

Command to run job

```
launch2.bat sampleDBtoCVSBatch -e FIN -u <username> -p <password> -f c:\messages.txt
```

6. Running a Batch Job using UI

1. Create the appropriate menu for Batch Execution UI.
2. By choosing the batch UI menu it will populate the batch ID from INF_BATCH_MASTER table with the appropriate role in the batch UI.
3. Then open a separate command prompt from console bin and run launch.bat BatchRestServer

To run BatchRestServer 8081 port should be in open state as dropwizard internally uses 8081 as default admin port. Also the port entered in INF_BATCH_RUNNER_CONFIG table should be in open state

4. Now choose the batch name from the drop-down and it will populate the UI.
5. Start the batch runner from console. (See above to run the batch runner)
6. The UI shows the fields which are displayable true only. By default all fields are displayable.
7. Now press Submit button for the system conformation.then press the confirmation button for the user confirmation.
8. After successful batch execution it will generate a reference no.

6.1 Database Setup needed for BatchUI

Create the table INF_BATCH_RUNNER_CONFIG and Insert appropriate values. Below is the example.

	CONFIG_PK	SERVER_HOST	SERVER_PORT	SERVER_STATUS	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE	UPDATED_BY	UPDATE_DATE
1	localhost	8888	STARTED	28-SEP-13 03:30:49	28-SEP-13 03:30:49	FINCH-SAMPLE	28-SEP-13 03:30:49	FINCH-SAMPLE	28-SEP-13 03:30:49	

7. Batch Job Query UI

Steps are as follows

1. Create the appropriate menu for Batch Execution Query.
2. User can search by Batch ID,Start Date From-To,Job Execution Id,Status,Exit Code,User Id

BCHQR:Batch Query [09-Oct-2013]

Batch ID	<input type="text"/>	Start Date From-To
Exit Code	<input type="text"/>	User Id
Sort Criteria	<input type="text"/> ① <input type="button" value="▼"/> <input type="checkbox"/> DESC	② <input type="text"/>
<input type="button" value="PERSONALIZE"/> <input type="button" value="SAVE QUERY"/>		

3. User Also can use short criteria for Start Time,Job Execution Id,Status,Exit Code
4. User presses Submit button to get the result.

3.4.6.3.3 Batch Report Using Jasper

1. Introduction

This page describes the steps required to develop a console based report using finch infrastructure.

1.1 Summary

In finch application, two types of reports are generated:

- Online report
- Batch report

For both these type of reports, Jasper has been used as the reporting framework.

Purpose of using Jasper Report are as follows:

- It is open-source and free software. Can be used for commercial purpose.
- It is considered as an alternate solution to overcome Repro related design issues such as page break and line break issues.
- It is able to use data coming from any kind of data source and produce pixel-perfect documents .
- The generated reports can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel and Word.

Reference

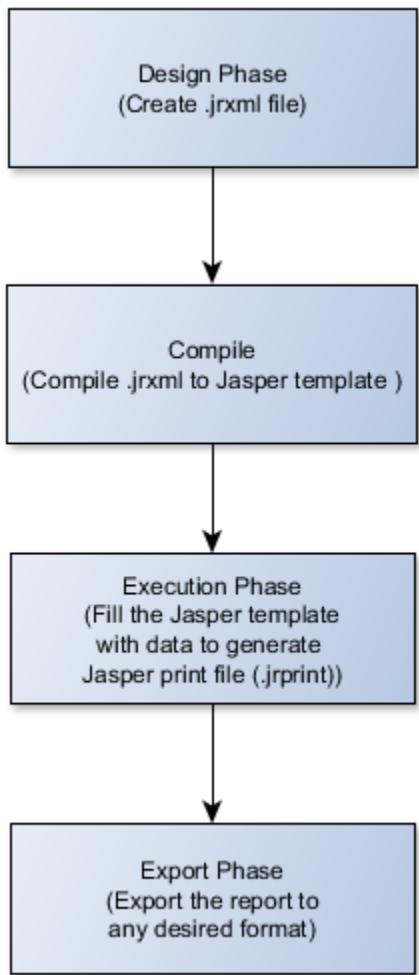
Refer to [Performance Report of Jasper](#) for more details on Jasper report performance

1.2 Scope

This document gives implementation details of Batch report generation using Jasper

2. Design

The main purpose of Jasper Reports is to create page oriented, ready to print documents in a simple and flexible manner. The flow chart depicting the typical work flow while creating Jasper Reports is shown below:



The life cycle for creating Jasper Reports has following distinct phases:

1. Designing the report - In this step the JRXML file is created, which is an XML document that contains the definition of the report layout. In finch, iReport tool has been used to create the report layout.
2. Compiling the report - In this step JRXML is compiled in a binary object called a Jasper file(*.jasper). Jasper files are actually needed in the application in order to run the reports. In finch, Spring supported classes for integration of JasperReports and JasperCompileManager.compileReport() have been used for compiling the .jrxml files for online and batch reports respectively.
3. Executing the report (Filling data into the report) - In this step data from the application is filled in the compiled report. The class net.sf.jasperreports.engine.JasperFillManager provides necessary functions to fill the data in the reports. A Jasper print file (*.jrprint) is created, which can be used to either print or export the report. In finch, Spring supported classes for integration of JasperReports and JasperFillManager.fillReport have been used for filling data into the online and batch reports respectively.
4. Exporting the report to desired format - In this step the Jasper print file, created in the previous step, is exported to any format using JasperExportManager. In finch, Spring supported classes for integration of JasperReports and JasperExportManager.exportReportToPdfStream() / JRExporter.exportReport() have been used for exporting the online and batch reports respectively to pdf/xls format. As Jasper provides various forms of exports, hence with the same input we can create multiple representations of the data.

2.1 Console Based Report Development Steps

Steps to develop console based reports have been mentioned below:

1. Create your jasper report template file(.jrxml file).

Recommended Report designing tool :

iReport

JasperStudio.

Sample jrxml File Expand

[source](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
  http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="Tread_Report" pageWidth="1402"
  pageHeight="1000" orientation="Landscape" columnWidth="1382" leftMargin="10" rightMargin="10"
  topMargin="10" bottomMargin="10">
  <property name="ireport.scriptlethandling" value="0"/>
  <property name="ireport.encoding" value="UTF-8"/>
  <property name="ireport.zoom" value="1.0"/>
  <property name="ireport.x" value="0"/>
  <property name="ireport.y" value="0"/>
  <import value="net.sf.jasperreports.engine.*"/>
  <import value="java.util.*"/>
  <import value="net.sf.jasperreports.engine.data.*"/>
  <style name="alternateStyle" isDefault="true" mode="Opaque" forecolor="#000000"
    backcolor="#CCCCCC" lineSpacing="Single" fontName="ARIALUNI" fontSize="10">
    <conditionalStyle>
      <conditionExpression><![CDATA[new Boolean($V{REPORT_COUNT}.intValue() % 2 ==
        0)]]></conditionExpression>
      <style backcolor="#FFFFFF" lineSpacing="Single"/>
    </conditionalStyle>
  </style>
  <parameter name="criteriaMap" class="java.util.Map"/>
  <parameter name="QUERY_SUBREPORT_PATH" class="java.lang.String"/>
  <parameter name="COMMON_SUBREPORT_TITLE_PATH" class="java.lang.String"/>
  <parameter name="COMMON_SUBREPORT_HEADER_PATH" class="java.lang.String"/>
  <parameter name="COMMON_SUBREPORT_FOOTER_PATH" class="java.lang.String"/>
  <parameter name="SUBREPORT_TITLE" class="java.lang.String"/>
  <parameter name="TITLE" class="java.lang.String"/>
  <parameter name="COMMON_REPORT_DATE" class="java.lang.String"/>
  <parameter name="COMMON_FORMATTED_REPORT_DATE" class="java.lang.String"/>
  <parameter name="FOOTER" class="java.lang.String"/>
  <field name="buySellFlag" class="java.lang.String">
    <fieldDescription><![CDATA[buySellFlag]]></fieldDescription>
  </field>
  <field name="accountBalanceType" class="java.lang.Long">
    <fieldDescription><![CDATA[accountBalanceType]]></fieldDescription>
  </field>
```

```

<field name="externalReferenceNo" class="java.lang.String">
<fieldDescription><![CDATA[externalReferenceNo]]></fieldDescription>
</field>
<field name="id" class="java.lang.String">
<fieldDescription><![CDATA[id]]></fieldDescription>
</field>
<field name="instrumentType" class="java.lang.String">
<fieldDescription><![CDATA[instrumentType]]></fieldDescription>
</field>
<field name="inventoryAccountNo" class="java.lang.String">
<fieldDescription><![CDATA[inventoryAccountNo]]></fieldDescription>
</field>
<field name="settlementCcy" class="java.lang.String">
<fieldDescription><![CDATA[settlementCcy]]></fieldDescription>
</field>
<field name="status" class="java.lang.String">
<fieldDescription><![CDATA[status]]></fieldDescription>
</field>
<field name="tradeCcy" class="java.lang.String">
<fieldDescription><![CDATA[tradeCcy]]></fieldDescription>
</field>
<field name="tradeDate" class="java.util.Date">
<fieldDescription><![CDATA[tradeDate]]></fieldDescription>
</field>
<field name="tradeDateStr" class="java.lang.String">
<fieldDescription><![CDATA[tradeDateStr]]></fieldDescription>
</field>
<field name="tradePk" class="java.lang.Long">
<fieldDescription><![CDATA[tradePk]]></fieldDescription>
</field>
<field name="tradeReferenceNo" class="java.lang.String">
<fieldDescription><![CDATA[tradeReferenceNo]]></fieldDescription>
</field>
<field name="tradeType" class="java.lang.String">
<fieldDescription><![CDATA[tradeType]]></fieldDescription>
</field>
<field name="valueDate" class="java.util.Date">
<fieldDescription><![CDATA[valueDate]]></fieldDescription>
</field>
<field name="valueDateStr" class="java.lang.String">
<fieldDescription><![CDATA[valueDateStr]]></fieldDescription>
</field>
<background>
<band splitType="Stretch"/>
</background>
<title>
<band height="40" splitType="Stretch">
<subreport>
<reportElement x="1" y="1" width="10" height="1"/>
<subreportParameter name="REPORT_RESOURCE_BUNDLE">
<subreportParameterExpression><![CDATA[$P{REPORT_RESOURCE_BUNDLE}]]></subreportParameterExpression>
</subreportParameter>
<subreportParameter name="TITLE">
<subreportParameterExpression><![CDATA[$P{TITLE}]]></subreportParameterExpression>
</subreportParameter>
<subreportParameter name="COMMON_FORMATTED_REPORT_DATE">

```

```

<subreportParameterExpression><![CDATA[$P{COMMON_FORMATTED_REPORT_DATE}]]></subreportParameterExpression>
</subreportParameter>
<subreportExpression class="java.lang.String"><![CDATA[$P{COMMON_SUBREPORT_TITLE_PATH} + ".jasper"]]></subreportExpression>
</subreport>
<subreport>
<reportElement x="1" y="39" width="10" height="1"/>
<subreportParameter name="REPORT_RESOURCE_BUNDLE">

<subreportParameterExpression><![CDATA[$P{REPORT_RESOURCE_BUNDLE}]]></subreportParameterExpression>
</subreportParameter>
<subreportParameter name="SUBREPORT_TITLE">

<subreportParameterExpression><![CDATA[$P{SUBREPORT_TITLE}]]></subreportParameterExpression>
</subreportParameter>
<dataSourceExpression><![CDATA[new net.sf.jasperreports.engine.data.JRBeanCollectionDataSource($P{criteriaMap}.entrySet())]]></dataSourceExpression>
<subreportExpression class="java.lang.String"><![CDATA[$P{QUERY_SUBREPORT_PATH} + ".jasper"]]></subreportExpression>
</subreport>
</band>
</title>
<pageHeader>
<band height="16" splitType="Stretch">
<subreport>
<reportElement x="0" y="15" width="0" height="0"/>
<subreportExpression class="java.lang.String"><![CDATA[$P{COMMON_SUBREPORT_HEADER_PATH} + ".jasper"]]></subreportExpression>
</subreport>
</band>
</pageHeader>
<columnHeader>
<band height="35" splitType="Stretch">
<textField isBlankWhenNull="false">
<reportElement key="staticText-3" mode="Opaque" x="87" y="0" width="138" height="35" backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$R{trade.label.trade.referenceno}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-4" mode="Opaque" x="0" y="0" width="87" height="35" backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
</box>

```

```

<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.trade.date}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-5" mode="Opaque" x="225" y="0" width="104" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.trdae.ccy}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-6" mode="Opaque" x="329" y="0" width="78" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.trade.type}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-7" mode="Opaque" x="407" y="0" width="76" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.value.date}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-8" mode="Opaque" x="483" y="0" width="143" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>

```

```

<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.account.balancetype}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-9" mode="Opaque" x="626" y="0" width="143" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.inventory.accountno}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-10" mode="Opaque" x="769" y="0" width="142" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.settlement.ccy}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-11" mode="Opaque" x="911" y="0" width="106" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.buysell.flag}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-12" mode="Opaque" x="1017" y="0" width="93" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>

```

```

<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.trade.status}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-13" mode="Opaque" x="1110" y="0" width="163" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.external.referenceno}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="staticText-14" mode="Opaque" x="1273" y="0" width="110" height="35"
backcolor="#999999"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI" size="11" isBold="true"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$R{trade.label.instrument.type}]]></textFieldExpression>
</textField>
</band>
</columnHeader>
<detail>
<band height="20" splitType="Stretch">
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="0" y="0" width="87"
height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{tradeDateStr}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">

```

```

<reportElement key="textField" style="alternateStyle" mode="Opaque" x="87" y="0" width="138" height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{tradeReferenceNo}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="911" y="0" width="106" height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{buySellFlag}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="769" y="0" width="142" height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{settlementCcy}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="626" y="0" width="143" height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{inventoryAccountNo}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="483" y="0" width="143"

```

```
height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$F{accountBalanceType}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="407" y="0" width="76"
height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{valueDateStr}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="1110" y="0" width="163"
height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression
class="java.lang.String"><![CDATA[$F{externalReferenceNo}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="329" y="0" width="78"
height="20"/>
<box topPadding="2" leftPadding="5" rightPadding="0">
<topPen lineWidth="1.0"/>
<leftPen lineWidth="1.0"/>
<bottomPen lineWidth="1.0"/>
<rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
<font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F{tradeType}]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
<reportElement key="textField" style="alternateStyle" mode="Opaque" x="1273" y="0" width="109"
height="20"/>
```

```

<box topPadding="2" leftPadding="5" rightPadding="0">
  <topPen lineWidth="1.0"/>
  <leftPen lineWidth="1.0"/>
  <bottomPen lineWidth="1.0"/>
  <rightPen lineWidth="1.0"/>
</box>
<textElement verticalAlignment="Top" lineSpacing="Single">
  <font fontName="ARIALUNI"/>
</textElement>
<textFieldExpression class="java.lang.String"><![CDATA[$F[instrumentType]]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
  <reportElement key="textField" style="alternateStyle" mode="Opaque" x="1017" y="0" width="93" height="20"/>
  <box topPadding="2" leftPadding="5" rightPadding="0">
    <topPen lineWidth="1.0"/>
    <leftPen lineWidth="1.0"/>
    <bottomPen lineWidth="1.0"/>
    <rightPen lineWidth="1.0"/>
  </box>
  <textElement verticalAlignment="Top" lineSpacing="Single">
    <font fontName="ARIALUNI"/>
  </textElement>
  <textFieldExpression class="java.lang.String"><![CDATA[$F[status]]]></textFieldExpression>
</textField>
<textField isBlankWhenNull="false">
  <reportElement key="textField" style="alternateStyle" mode="Opaque" x="225" y="0" width="104" height="20"/>
  <box topPadding="2" leftPadding="5" rightPadding="0">
    <topPen lineWidth="1.0"/>
    <leftPen lineWidth="1.0"/>
    <bottomPen lineWidth="1.0"/>
    <rightPen lineWidth="1.0"/>
  </box>
  <textElement verticalAlignment="Top" lineSpacing="Single">
    <font fontName="ARIALUNI"/>
  </textElement>
  <textFieldExpression class="java.lang.String"><![CDATA[$F[tradeCcy]]]></textFieldExpression>
</textField>
<break>
<reportElement x="0" y="19" width="100" height="1">
  <printWhenExpression><![CDATA[new java.lang.Boolean((int)($V{REPORT_COUNT}.intValue()%20==0))]]></printWhenExpression>
</reportElement>
</break>
</band>
</detail>
<columnFooter>
  <band height="21" splitType="Stretch">
    <textField>
      <reportElement x="1110" y="0" width="272" height="20" backcolor="#FFFFFF"/>
      <textElement textAlignment="Right" lineSpacing="Single"/>
      <textFieldExpression class="java.lang.String"><![CDATA[$V{PAGE_NUMBER}]]></textFieldExpression>
    </textField>
    <line>
      <reportElement x="1" y="20" width="1381" height="1" forecolor="#999999"/>
    </line>
  </band>
</columnFooter>

```

```
<pageFooter>
<band height="45" splitType="Stretch">
<subreport>
<reportElement x="1" y="0" width="97" height="4"/>
<subreportParameter name="REPORT_RESOURCE_BUNDLE">

<subreportParameterExpression><![CDATA[$P{REPORT_RESOURCE_BUNDLE}]]></subreportParameterExpression>
</subreportParameter>
<subreportParameter name="FOOTER">
<subreportParameterExpression><![CDATA[$P{FOOTER}]]></subreportParameterExpression>
</subreportParameter>
<subreportExpression class="java.lang.String"><![CDATA[$P{COMMON_SUBREPORT_FOOTER_PATH} + ".jasper"]]></subreportExpression>
</subreport>
</band>
</pageFooter>
<lastPageFooter>
<band height="45" splitType="Stretch">
<subreport>
<reportElement x="1" y="0" width="97" height="4"/>
<subreportParameter name="REPORT_RESOURCE_BUNDLE">

<subreportParameterExpression><![CDATA[$P{REPORT_RESOURCE_BUNDLE}]]></subreportParameterExpression>
</subreportParameter>
<subreportParameter name="FOOTER">
<subreportParameterExpression><![CDATA[$P{FOOTER}]]></subreportParameterExpression>
</subreportParameter>
<subreportExpression class="java.lang.String"><![CDATA[$P{COMMON_SUBREPORT_FOOTER_PATH} + ".jasper"]]></subreportExpression>
</subreport>
</band>
</lastPageFooter>
<summary>
<band splitType="Stretch"/>
</summary>
```

```
</jasperReport>
```

2. Compile the fileName.jrxml file which will create fileName.jasper file.
3. Put the jrxml file into appropriate place (e.g. \$sample-app\$sample-trd\$src\$main\$console\$report\$templates\$trd\$fileName.jrxml)
4. Create bean class in your application to generate the report* (e.g. here we create com.nrft.sample.trd.report.TrdReportView*).

TrdReportView.java

› Expand

[source](#)

```
package com.nrft.sample.trd.report;
import java.io.Serializable;
import java.util.Date;
import com.google.common.annotations.Beta;
import com.nrft.finch.inf.utils.DateUtils;
public class TrdReportView implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private long tradePk;
    private String tradeType;
    private String tradeReferenceNo;
    private String tradeCcy;
    private Date tradeDate;
    private Date valueDate;
    private String buySellFlag;
    private long accountBalanceType;
    private String inventoryAccountNo;
    private String settlementCcy;
    private String instrumentType;
    private String externalReferenceNo;
    private String status;

    public TrdReportView(long tradePk, String tradeType, String tradeReferenceNo,
        String tradeCcy, Date tradeDate, Date valueDate, String buySellFlag, long accountBalanceType, String
        inventoryAccountNo, String settlementCcy, String instrumentType, String externalReferenceNo, String
        status) {
        super();
        this.tradeType = tradeType;
        this.tradeReferenceNo = tradeReferenceNo;
        this.tradeCcy = tradeCcy;
        this.tradeDate = tradeDate;
        this.valueDate = valueDate;
        this.buySellFlag = buySellFlag;
        this.accountBalanceType = accountBalanceType;
        this.inventoryAccountNo = inventoryAccountNo;
        this.settlementCcy = settlementCcy;
        this.instrumentType = instrumentType;
        this.externalReferenceNo = externalReferenceNo;
        this.status = status;
        this.setId(tradeReferenceNo);
    }

    public String getTradeType() {
        return tradeType;
    }
}
```

```
}

public void setTradeType(String tradeType) {
    this.tradeType = tradeType;
}
public String getTradeReferenceNo() {
    return tradeReferenceNo;
}
public void setTradeReferenceNo(String tradeReferenceNo) {
    this.tradeReferenceNo = tradeReferenceNo;
}
public String getTradeCcy() {
    return tradeCcy;
}
public void setTradeCcy(String tradeCcy) {
    this.tradeCcy = tradeCcy;
}
public Date getTradeDate() {
    return tradeDate;
}
public void setTradeDate(Date tradeDate) {
    this.tradeDate = tradeDate;
}
public String getBuySellFlag() {
    return buySellFlag;
}
public void setBuySellFlag(String buySellFlag) {
    this.buySellFlag = buySellFlag;
}

public String getTradeDateStr() {
    return DateUtils.formatDate(this.tradeDate);
}
public Date getValueDate() {
    return valueDate;
}
public void setValueDate(Date valueDate) {
    this.valueDate = valueDate;
}

public String getValueDateStr() {
    return DateUtils.formatDate(this.valueDate);
}

public long getAccountBalanceType() {
    return accountBalanceType;
}
public void setAccountBalanceType(long accountBalanceType) {
    this.accountBalanceType = accountBalanceType;
}
public String getInventoryAccountNo() {
    return inventoryAccountNo;
}
public void setInventoryAccountNo(String inventoryAccountNo) {
    this.inventoryAccountNo = inventoryAccountNo;
}
public String getSettlementCcy() {
    return settlementCcy;
}
public void setSettlementCcy(String settlementCcy) {
```

```
this.settlementCcy = settlementCcy;
}
public String getInstrumentType() {
    return instrumentType;
}
public void setInstrumentType(String instrumentType) {
    this.instrumentType = instrumentType;
}

public String getExternalReferenceNo() {
    return externalReferenceNo;
}
public void setExternalReferenceNo(String externalReferenceNo) {
    this.externalReferenceNo = externalReferenceNo;
}
public String getStatus() {
    return status;
}
public void setStatus(String status) {
    this.status = status;
}
@Beta
// for grid rendering
private String id;

public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public long getTradePk() {
    return tradePk;
}
public void setTradePk(long tradePk) {
    this.tradePk = tradePk;
}
```

```
}
```

5. Create Report class*(e.g. here we create com.nrft.sample.trd.console.report.TradeReport* which extends com.nrft.fin.ch.inf.console.report.ConsoleReportLauncher class of finch).

TradeReport

```
package com.nrft.sample.trd.console.report;
import java.util.List;
import com.nrft.fin.ch.inf.console.report.ConsoleReportLauncher;
import com.nrft.fin.ch.inf.exception.FinchException;
import com.nrft.fin.ch.inf.startup.Operation;
import com.nrft.sample.trd.domain.service.TradeService;
import com.nrft.sample.trd.report.TrdReportView;
public class TradeReport extends ConsoleReportLauncher{
    /**
     * Returns report template path for the report.
     */
    @Override
    protected String getReportTemplatePath(String format) {
        return "trd/TradeReport.jrxml";
    }
    /**
     * returns report data
     */
    @Override
    protected List<TrdReportView> getData() throws FinchException {
        return Operation.getInstance().getReference(TradeService.class).getReportQueryResult();
    }
    /**
     * returns report ID
     */
    @Override
    protected String getReportId() {
        return "SATRD";
    }
}
```

6. Override getReportTemplatePath() method and return your jrxml file path relative to application-root directory. We recommend to use component-wise segregation for template files.
7. Override getData() which return List of bean defined is step 4.
8. Override getReportId() which return String where you will return the Report Id inserted into INF_REPORT table as string (Optional).
9. Edit the console-executable-mapping.properties file and enter the property to class the class you have created to compile and run the jrxml file to create report.

```
TRD.TradeReport = com.nrft.sample.trd.console.report.TradeReport
```

2.2 Run Console Report

To run the console report you have to follow the following steps

1. Extract sample-console.zip into the folder (e.g. sample-console)

2. open command prompt and goto the bin directory (e.g. D:\Project\finch\sample-console\bin)
3. now run the command to generate report (command:launch.bat TRD.TradeReport -e FIN -f pdf)

Now go back to the folder where you have extracted the sample-console.zip and check two folders are created 1.logs 2. report. In logs folder you can check all the logs and in report folder you can get the report you have generated.

2.3 Use of Tables for Report

Tables have been listed below:

- INF_FILE_DIRECTORY → One entry will must be there before generate any console report. It contains the path entity**

	FILE_DIRECTORY_PK	ENTERPRISE_ID	DIRECTORY_ID	PATH	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE	UPDATED_BY	UPDATE_DATE
1	1	FIN	REPORT	D:\Project\finch\sample-console\report	20-03-13	20-03-13	FINCH-SAMPLE	20-03-13	FINCH-SAMPLE	20-03-13

- INF_REPORT → This table contains few important entities like REPORT_ID, SHORT_NAME, FILE_NAME_PATTERN etc. The data will be inserted before you create report and the REPORT_ID value should be same as the value you return from the method getReportId() of the class com.nrft.sample.trd.console.report.TradeReport.

```

@Override
protected String getReportId() {
    return "SATRD";
}

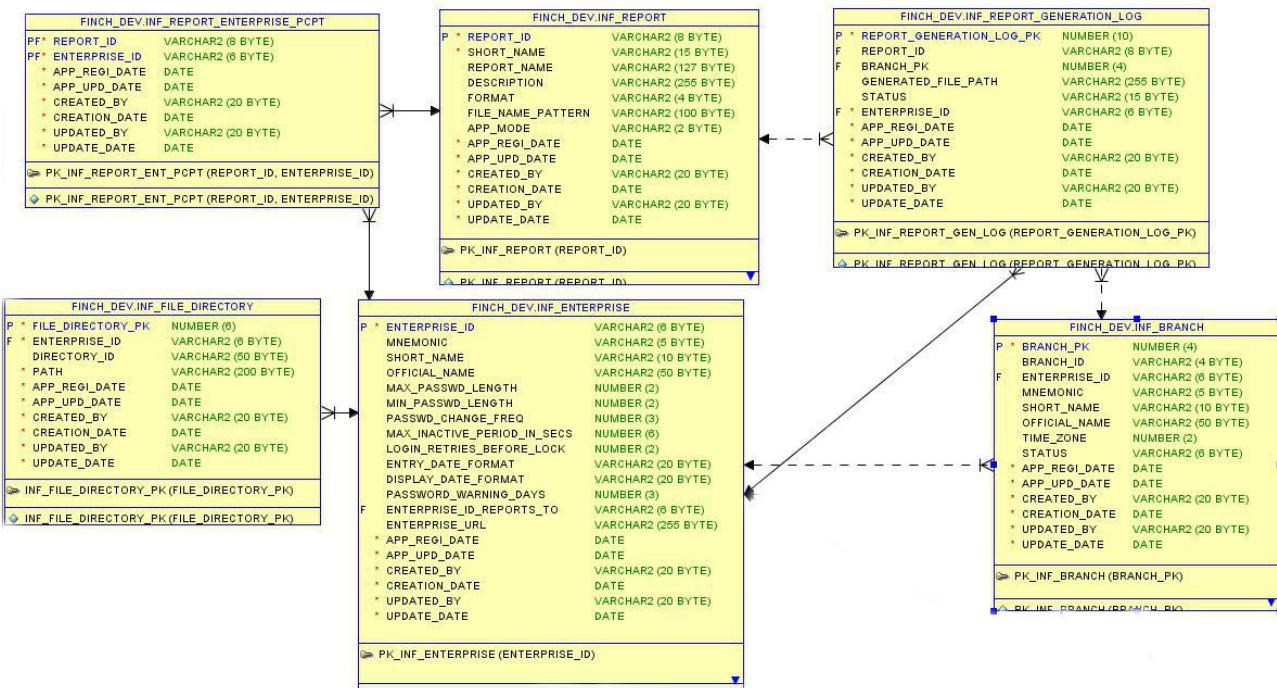
```

	REPORT_ID	SHORT_NAME	REPORT_NAME	DESCRIPTION	APP_MODE	APP_REGI_DATE	APP_UPD_DATE	CREATED_BY	CREATION_DATE	UPDATED_BY	UPDATE_DATE	FORMAT	FILE_NAME_PATTERN
1	SATRD	TradeReport	Trade Report	Sample Trade Report	(null)	11-02-13	11-02-13	test	11-02-13	test	11-02-13	xlsx	#{reportId}-#{appDate}-#{ent...}

- INF_REPORT_ENTERPRISE_PCPT
- INF_REPORT_GENERATION_LOG → Whenever you generate any report one entry will be insert in this table.

2.4 Relationship Between Tables

The following image displays the relationships between the tables.



3. Document History

Version	Date	Purpose	Author
1.0	03-Feb-2015	Initial document	rahulc

3.4.6.3.4 Password Encryption Batch

Introduction

This document describes how to configure finch provided batch for encrypting and resetting user password.

Background

finch provides two batch for encrypting and resetting user password. finch now provides user password to encrypted as stored in database. For existing application who have their user password as plain text in database can encrypt existing password and reset password by executing these batches.

Configuration

Add following bean definition in application-context.xml for console

application-context.xml

```
<bean id="passwordEncoder" class="org.springframework.security.crypto.password.NoOpPasswordEncoder" />
```

Changes in security-batch.xml file

change the class attribute of bean finchAuthenticationProvider to com.nrft.fin.ch.inf.security.AuthenticationProvider and add a new property name="passwordEncoder"

security-batch.xml

```
<beans:bean id="finchAuthenticationProvider" class="com.nrft.fin.ch.inf.security.AuthenticationProvider">
    <beans:property name="userDetailsService" ref="userDetailsService"/>
    <beans:property name="passwordEncoder" ref="passwordEncoder"/>
</beans:bean>
```

add a new bean definition for passwordEncoder

security-batch.xml

```
<beans:bean id="passwordEncoder"
    class="org.springframework.security.crypto.password.NoOpPasswordEncoder" />
```

change the class attribute of bean userDetailsService to com.nrft.fin.ch.inf.security.FinchAuthenticationService

security-batch.xml

```
<beans:bean id="userDetailsService"
    class="com.nrft.fin.ch.inf.security.FinchAuthenticationService">
    <beans:property name="dataSource" ref="dataSource" />
    <beans:property name="usersByUsernameQuery"
        value="select e.user_id, e.appl_passwd, e.status, e.locked, e.start_date, e.end_date from inf_employee e
        where e.user_id = ?" />
    <beans:property name="usersByUserAndEnterpriseQuery"
        value="select e.user_id,e.appl_passwd,e.status,e.locked,e.start_date,e.end_date FROM inf_employee
        e,inf_branch b,inf_enterprise ie where e.default_branch_pk = b.branch_pk and b.enterprise_id =
        ie.enterprise_id and e.user_id = and ie.enterprise_id = ?" />
    <beans:property name="authoritiesByUsernameQuery"
        value="select e.user_id, r.application_role_name from inf_employee e, inf_emp_appln_role_participant erp,
        inf_application_role r where e.employee_pk = erp.employee_pk and erp.appl_role_pk = r.appl_role_pk and
        .status='NORMAL' and e.user_id = ?" />
</beans:bean>
```

Add following values in console-executable-mapping.properties file

```
ResetPassword=com.nrft.fin.ch.inf.console.utility.DatabasePasswordResetBean
EncryptPassword=com.nrft.fin.ch.inf.console.utility.DatabasePasswordSecurerBea
```

Batch Details

For details about the tow batches please refer to the [link](#)

Password Encoder

It is a interface provided by spring security to encode password. For details follow the [link](#)

3.4.7 Unit Testing

1. Introduction

In finch, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process.

2. Prerequisite

- Technologies used :-

- JUnit
- AssertJ
- Assert-guava
- H2database
- DBunit
- ActiveMq

NOTE:- These libraries are already present in the finch repositories.

- Brief Knowledge

The unit tester must have some basic knowledge and understanding of

- JUnit
- DBUnit
- Spring TestContext Framework

3. Candidates

The different candidates which are needed to be unit tested are :-

- Controllers
- Console
- Services
- Repository
- Report

Test Class Hierarchy

com.nrft.fin.ch.inf.domain.repository.AbstractDataDrivenTests

This is an abstract class which should be extended to write tests that need both database access and Operation initialization. It sets the Junit test runner class and registers various TestExecutionListeners. It has methods to initialize Operation before each test and destroy it after completion of each test. The sub-classes of this class must create and configure an ApplicationContext and configure a data source.

com.nrft.fin.ch.inf.domain.repository.AbstractCoreDataDrivenTests

This is an abstract class in the module finch-core. It creates and configures an ApplicationContext using the annotation @ContextConfiguration and sets the data source using the annotation @DbUnitConfiguration. This class should be extended to write tests for any class in the finch-core module. The sub-classes must have code for adding and deleting tuples from the configured data source.

com.nrft.fin.ch.dbd.domain.repository.AbstractDashboardDataDrivenTests

This is an abstract class in the module finch-dbd. Its function is similar to that of AbstractCoreDataDrivenTests above.

com.nrft.fin.ch.inf.domain.repository.AbstractBatchDataDrivenTests

This is an abstract class in the module finch-batch. Its function is similar to that of AbstractCore

DataDrivenTests above.

- com.nrft.fin.ch.inf.domain.repository.AbstractConsoleDataDrivenTests

This is an abstract class in the module finch-console-infra. Its function is similar to that of AbstractCoreDataDrivenTests above.

- com.nrft.fin.ch.inf.domain.repository.AbstractWebDataDrivenTests

This is an abstract class in the module finch-web-infra. Its function is similar to that of AbstractCoreDataDrivenTests above.

- com.nrft.fin.ch.inf.push.actors.AbstractNotificationDataDrivenTests

This is an abstract class in the module finch-notification. Its function is similar to that of AbstractCoreDataDrivenTests above.

- com.nrft.fin.ch.inf.AbstractOperationInitTests

This is an abstract class which should be extended to write tests that require Operation initialization but do not require database access. It sets the Junit test runner class. It has methods to initialize Operation before each test and destroy it after completion of each test. The subclasses of this class must create and configure an ApplicationContext.

- com.nrft.fin.ch.inf.AbstractCoreOperationInitTests

This is an abstract class in the module finch-core. It creates and configures an ApplicationContext using the annotation @ContextConfiguration. This class should be extended to write tests for the finch-core module.

com.nrft.fin.ch.inf.batch.AbstractBatchOperationInitTests

This is an abstract class in the module finch-batch. Its function is similar to that of AbstractCoreOperationInitTests above.

com.nrft.fin.ch.inf.scheduler.AbstractSchedulerOperationInitTests

This is an abstract class in the module finch-batch. Its function is similar to that of AbstractCoreOperationInitTests above.

com.nrft.fin.ch.inf.web.controller.AbstractControllerTests2

This is an abstract class that should be extended to write tests for Spring MVC Controllers. It creates database tables from Entity Java classes and populates the tables with provided test data set. Tests written by extending this class may be considered as integration test because they cover Spring Application Context, Spring MVC Configuration and Spring Security context in one place. It sets the JUnit test runner class. It provides a MockServletContext and MockMvc instance. The subclasses must create and configure an ApplicationContext instance, have a method named controller() which will return a reference to an object of the Controller class being tested.

com.nrft.fin.ch.inf.web.controller.AbstractWebControllerTest

This is an abstract class in the module finch-web-infra. Using the annotations @ContextConfiguration and @WebAppConfiguration it provides an instance of WebApplicationContext. Using the annotation @DbUnitConfiguration it sets a data source. The sub-classes of this class must have code for adding tuples to and deleting tuples from the configured data source. The sub-class must have a method named controller() which will return a reference to an object of the Controller class being tested. This class must be extended to write tests for the finch-web-infra module.

com.nrft.fin.ch.inf.web.controller.AbstractUploadControllerTests

This an abstract class in the module finch-upload. Its function is similar to that of

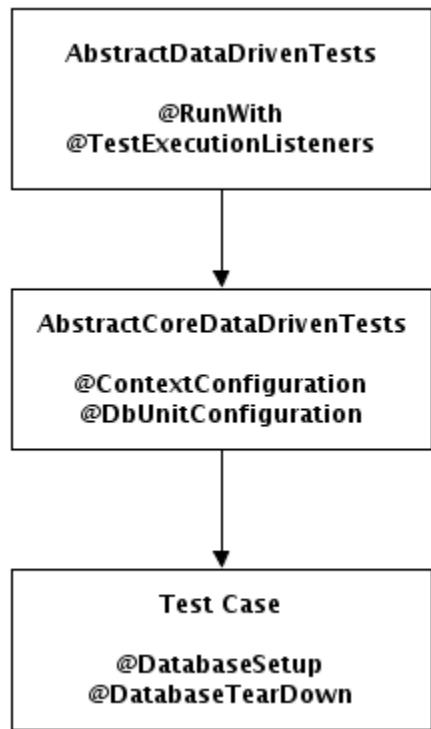
AbstractWebControllerTests above.

[-] com.nrft.fin.ch.dbd.controller.AdstractDashboardControllerTests

This an abstract class in the module finch-dbd. Its function is similar to that of AbstractWebControllerTests above.

4. General Setup Guidelines

Unit Test which involve database related operations must incorporate the following structure (But require Operation initialization):



- **AbstractDataDrivenTests** is the Base class for data driven JUnit test class in finch. It performs Operation initialization and destroy.

AbstractDataDrivenTests

› Expand

```
@RunWith(SpringJUnit4ClassRunner.class)
@TestExecutionListeners({DependencyInjectionTestExecutionListener.class,
    DirtyContextTestExecutionListener.class,
    TransactionalTestExecutionListener.class,
    TransactionDbUnitTestExecutionListener.class })
public abstract class AbstractDataDrivenTests {

    @Before
    public void beforeDataDrivenTest() throws FinchException {
        Operation.init(getStartupConfiguration());
    }

    @After
    public void afterDataDrivenTest() throws FinchException {
        Operation.destroy();
    }

    protected StartupConfiguration getStartupConfiguration() {
        return StartupConfigurations.defaultBatchConfigBuilder(getConfigURI())
            .enterpriseld(getEnterpriseld())
            .springContext(getApplicationContext()).build();
    }

    protected abstract String getEnterpriseld();

    protected abstract String getConfigURI();

    protected abstract ApplicationContext getApplicationContext();
}
```

[source](#)

- `AbstractCoreDataDrivenTests` is the abstract class for data driven JUnit test class in `finch core`. It sets the Context configuration & DbUnit configurations.

AbstractCoreDataDrivenTests

› Expand

```
@ContextConfiguration(locations = {"classpath: META-INF/config/application-context-core-h2.xml"})
@DbUnitConfiguration(databaseConnection = "GLOBAL")
public abstract class AbstractCoreDataDrivenTests extends AbstractDataDrivenTests {

    @Autowired
    protected ApplicationContext applicationContext;

    @Override
    protected String getEnterpriseld() {
        return "FIN";
    }

    @Override
    protected String getConfigURI() {
        return "classpath: META-INF/config/application-context-core-h2.xml";
    }

    @Override
    protected ApplicationContext getApplicationContext() {
        return applicationContext;
    }
}
```

- All Data Driven Test class extends the AbstractCoreDataDrivenTests class. Every test case has to setup their @DatabaseSetup and @DatabaseTearDown. For example EmployeeRepositoryTests is given below.

EmployeeRepositoryTests

› Expand

```
@DatabaseSetup(value = "classpath: data/finch-core-inf.xml", type = DatabaseOperation.INSERT)
@DatabaseTearDown(value = "classpath: data/finch-core-inf-del.xml", type =
DatabaseOperation.DELETE_ALL)
public class EmployeeRepositoryTests extends AbstractCoreDataDrivenTests {
    @Autowired
    private EmployeeRepository empRepo;
    @Autowired
    private LanguageRepository langRepo;
    @Test
    @Transactional(value = "global-tm", readOnly = true)
    public void getEmployeeByUserId() throws FinchException {
        assertThat(empRepo.getEmployeeByUserId("finch")).isPresent();
        assertThat(empRepo.getEmployeeByUserId("fake")).isAbsent();
    }
    @Test
    @Transactional(value = "global-tm", readOnly = true)
    public void getEmployeeByUserIdWithStatus() throws FinchException {
        assertThat(empRepo.getEmployeeByUserId("finch")).isPresent();
        assertThat(empRepo.getEmployeeByUserId("finch", "Normal")).isPresent();
        assertThat(empRepo.getEmployeeByUserId("password", "cancel")).isPresent();
    }
    @Test
    @Transactional
    @DatabaseSetup(value = "classpath: data/finch-core-inf-emp.xml", type = DatabaseOperation.DELETE)
    public void newEmployee() throws FinchException {
```

```

Employee e = new Employee();
e.setLastName("last");
e.setFirstName("first");
e.setMiddleInitial("m.");
e.setSuffix("Mr.");
e.setTitle("Title");
e.setUserId("testuser");
e.setApplPasswd("testuser");
e.setStartDate(LocalDate.now().toDate());
e.setEndDate(new LocalDate(2050, 12, 31).toDate());
e.setLastPasswordChangeDate(LocalDate.now().toDate());
e.setWrongPasswordCount(new BigDecimal(5));
e.setStatus("NORMAL");
e.setLocked("N");
e.setMail("testuser@example.com");
e.setEmployeeOpenDate(LocalDate.now().toDate());
e.setEmployeeOpenedBy("junit-test");
EmployeeNameXref empNameXRef = new EmployeeNameXref();
// get Language
Language lang = langRepo.find(Language.class, "EN");
empNameXRef.setLanguage(lang);
empNameXRef.setFirstName("firstname-en");
empNameXRef.setLastName("lastname-en");
e.setEmployeeNameXrefs(Sets.newHashSet(empNameXRef));
empRepo.write(e);
Optional<Employee> optEmp = empRepo.getEmployeeByUserId("testuser");
Employee emp = optEmp.get();
int previouudVal = emp.getConcurrencyCheck();
emp.setTitle("AB");
if (Operation.getInstance().isCurrentContextSet()) {
    Operation.getInstance().getContext()
        .setCallerIdentity(Identity.systemUser());
}
empRepo.update(emp);
optEmp = empRepo.getEmployeeByUserId("testuser");
assertThat(optEmp.get().getConcurrencyCheck()).isGreaterThan(
    previouudVal);
assertThat(optEmp.get().getEmployeeNameXrefs()).hasSize(1);
empRepo.lock(emp, LockModeType.PESSIMISTIC_WRITE);
empRepo.unlock(emp);
assertThat(optEmp.get().getUpdatedBy()).isEqualTo(
    Identity.systemUser().getUserId());
}
@Test
@Transactional(readOnly = true)
public void getEnterpriseldByUserId() {
    assertThat(empRepo.getEnterpriseldByUserId("finch")).isPresent();
    assertThat(empRepo.getEnterpriseldByUserId("fake")).isAbsent();
}
@Test
@Transactional(readOnly = true)
public void getEmployeeByEmployeePk() {
    assertThat(empRepo.getEmployeeByEmployeePk(1L)).isPresent();
    assertThat(empRepo.getEmployeeByEmployeePk(10L)).isAbsent();
}
@Test
@Transactional(value = "global-tm", readOnly = true)
public void getEmployees() {
    List<Employee> emps = empRepo.getEmployees("FIN");
}

```

```

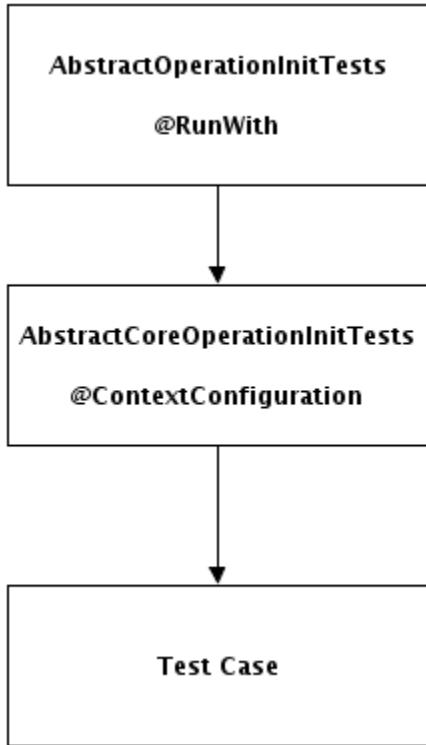
        assertThat(emps).hasSize(emps.size());
        emps = empRepo.getEmployees("GIN");
        assertThat(emps).isEmpty();
    }
    @Test
    @Transactional(readOnly = true)
    public void getApplicationRoleAndBranch() {
        assertThat(empRepo.getApplicationRoleAndBranch(1L)).hasSize(2);
        assertThat(empRepo.getApplicationRoleAndBranch(10L)).isEmpty();
    }
    @Test
    @Transactional(readOnly = true)
    public void getRoleNamesForEmployee() throws FinchException {
        assertThat(empRepo.getRoleNamesForEmployee("finch", "FIN")).hasSize(2);
    }
    @Test
    @Transactional(readOnly = false)
    public void updateEmployee() throws FinchException {
        Employee employee = empRepo.getEmployeeByUserId("finch").get();
        employee.setFirstName("FNCHT");
        employee.setLastName("Empl");
        empRepo.updateEmployee(employee);
        assertThat(empRepo.getEmployeeByUserId("finch").get().getFirstName())
            .isEqualTo("FNCHT");
    }
    @Test
    @Transactional(readOnly = false)
    public void saveEmployee() throws FinchException {
        Employee employee = empRepo.getEmployeeByUserId("finch").get();
        employee.setFirstName("FNCHT");
        employee.setLastName("Empl");
        empRepo.saveEmployee(employee);
        assertThat(empRepo.getEmployeeByUserId("finch").get().getFirstName())
            .isEqualTo("FNCHT");
    }
    @Test
    @Transactional(readOnly = false)
    public void cancelEmployee() throws FinchException {
        Employee employee = empRepo.getEmployeeByUserId("finch").get();
        empRepo.cancelEmployee(employee);
        assertThat(empRepo.getEmployeeByUserId("finch")).isAbsent();
    }
    @Test
    @Transactional(value = "global-tm", readOnly = false)
    public void removeApplicationRoleAndBranch() throws FinchException {
        EntityManager entityManager = empRepo.getEntityManager();
        EmpApplnRoleParticipantPK pk = new EmpApplnRoleParticipantPK();
        pk.setApplRolePk(1);
        pk.setEmployeePk(1);
        EmpApplnRoleParticipant empRP = entityManager.find(
            EmpApplnRoleParticipant.class, pk);
        List<EmployeeBranchRoleDTO> branchRoleDTOs = empRepo
            .getApplicationRoleAndBranch(1L);
        assertThat(branchRoleDTOs).hasSize(2);
        empRepo.removeApplicationRoleAndBranch(empRP);
        branchRoleDTOs = empRepo.getApplicationRoleAndBranch(1L);
        branchRoleDTOs.get(0).setAcRestrictionFlag("Y");
        assertThat(branchRoleDTOs).hasSize(1);
        assertThat(branchRoleDTOs.get(0).getBranch()).isEqualTo("FBR");
    }
}

```

```
assertThat(branchRoleDTOs.get(0).getBranchPk().toString()).isEqualTo("1");
assertThat(branchRoleDTOs.get(0).getRole()).isEqualTo("ROLE2");
assertThat(branchRoleDTOs.get(0).getRolePk().toString()).isEqualTo("2");
assertThat(branchRoleDTOs.get(0).getEmployeePk().toString()).isEqualTo("1");
assertThat(branchRoleDTOs.get(0).getAcRestrictionFlag()).isEqualTo("Y");
}
@Test
public void testGetMessageResourceBundle() throws FinchException {
    Component comp = Component.getComponent("INF").get();
    ResourceBundle rb = ComponentUtils.getMessageResourceBundle(comp, null);
    assertThat(rb.isNotNull());
    Operation.getInstance().getContext().setCallerIdentity(Identities.webUser("finch", "dbunit-web"));
    rb = ComponentUtils.getMessageResourceBundle(comp, null);
```

```
        assertThat(rb.isNotNull());
    }
}
```

Unit Test which does not involve database related operations must incorporate the following structure (But require Operation initialization) :



- AbstractOperationInitTests is the Base class for all those test that require Operation initialization before execution of test case. No database is initialized. So it can be used where Operation needs to be initialized without any database

AbstractOperationInitTests

› Expand

```
@RunWith(SpringJUnit4ClassRunner.class)
public abstract class AbstractOperationInitTests {
    @Before
    public void beforeOperationInitTest() throws FinchException {
        Operation.init(getStartupConfiguration());
    }

    @After
    public void afterOperationInitTest() throws FinchException {
        Operation.destroy();
    }

    protected StartupConfiguration getStartupConfiguration() {
        return StartupConfigurations.defaultBatchConfigBuilder(getConfigURI())
            .enterpriseld(getEnterpriseld())
            .springContext(getApplicationContext()).build();
    }

    protected abstract String getEnterpriseld();

    protected abstract String getConfigURI();

    protected abstract ApplicationContext getApplicationContext();
}
```

[source](#)

- AbstractCoreOperationInitTests is the base Core JUnit test class for test that require Operation initialization before execution of test case in finch-core module. Only the context is configured here.

AbstractCoreOperationInitTests

› Expand

```
@ContextConfiguration(locations = { "classpath: META-INF/config/application-context-core-h2.xml" })
public abstract class AbstractCoreOperationInitTests extends AbstractOperationInitTests {

    @Autowired
    protected ApplicationContext applicationContext;
    protected String getEnterpriseld() {
        return "FIN";
    }

    protected String getConfigURI() {
        return "classpath: META-INF/config/application-context-core-h2.xml";
    }

    protected ApplicationContext getApplicationContext() {
        return applicationContext;
    }
}
```

[source](#)

- All Non Data Driven Test class extends the AbstractCoreOperationInitTests class(which also requires Operation initialization before execution of test case). For example ComponentTests is given below.

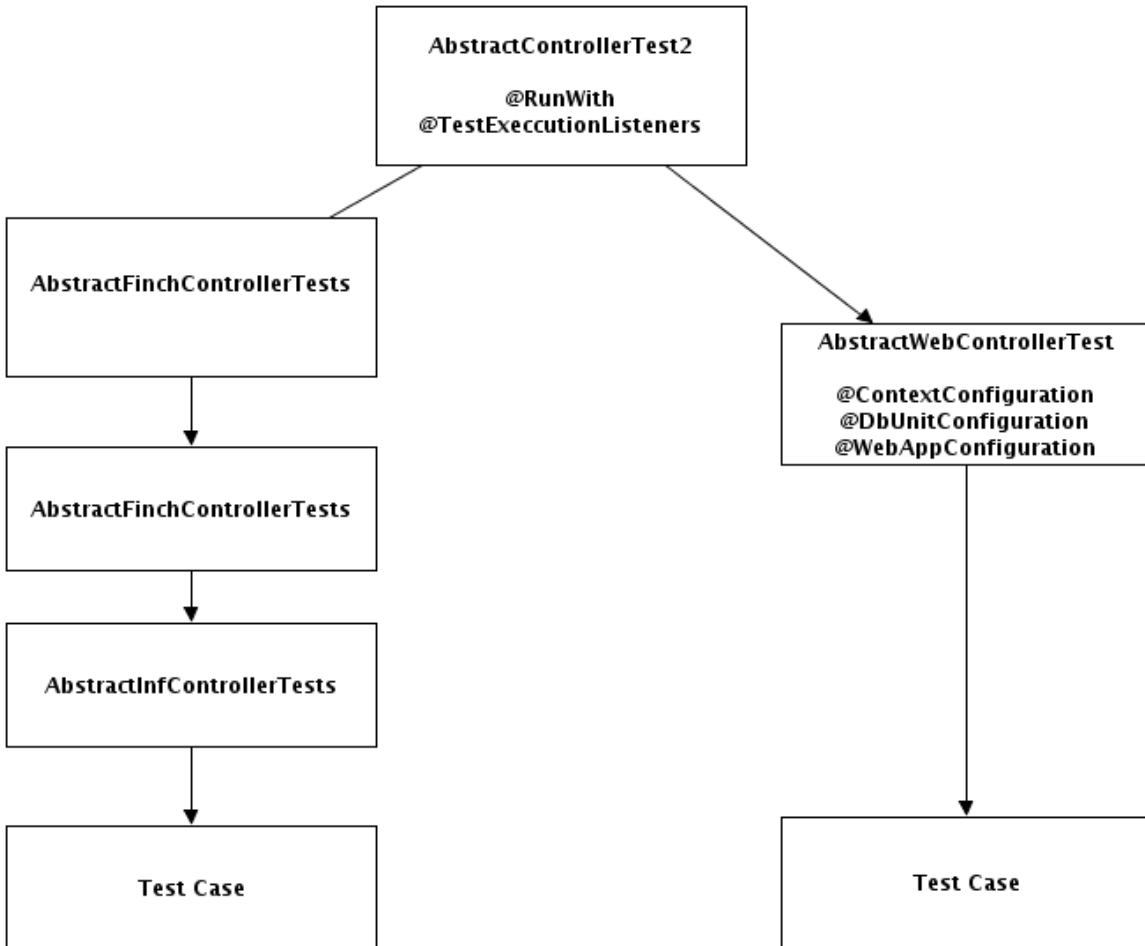
EmployeeRepositoryTests

› Expand

```
public class ComponentTests extends AbstractCoreOperationInitTests {  
    @Test  
    public void getComponent() {  
        Optional<Component> com = Component.getComponent("INF");  
        assertThat(com).isPresent();  
        com = Component.getComponent("DBD");  
        assertThat(com).isAbsent();  
    }  
  
    @Test  
    public void getAllComponent() {  
        Collection<Component> coms = Component.getAllComponent();  
        assertThat(coms).hasSize(2).areExactly(1, new Condition<Component>() {  
            @Override  
            public boolean matches(Component value) {  
                return value.getName().equals("INF");  
            }  
        });  
    }  
  
    @Test  
    public void getParent() {  
        Optional<Component> com = Component.getComponent("INF");  
        assertThat(com.get().getParent()).isNull();  
    }  
  
    @Test  
    public void getPackageName() {  
  
        assertThat(Component.getComponent("INF").get().getPackageName()).isEqualTo("com.nrift.finch.inf");  
        Component comp = new Component("TEST");  
        assertThat(comp.getPackageName()).isEqualTo("com.nrift.finch.test");  
    }  
  
    @Test  
    public void testComponent() {  
        Component comp = new Component("TEST",null,Component.getComponent("INF").get());  
        String packageName = comp.getPackageName();  
        assertThat(packageName).isNotNull();  
    }  
}
```

[source](#)

Unit Test which involve controllers must incorporate the following structure:



- AbstractControllerTest2 is the Abstract base class for any Spring MVC controller test; can be treated as integration test for Spring MVC web application because it covers application Spring context, String MVC configuration and Spring Security context in one place. It creates database table from Entity Java classes and populates the tables with provided test data set.

AbstractControllerTests2

```

@RunWith(SpringJUnit4ClassRunner.class)
@TestExecutionListeners({DependencyInjectionTestExecutionListener.class,
                      DirtyContextTestExecutionListener.class,
                      TransactionalTestExecutionListener.class,
                      DbUnitTestExecutionListener.class })
public abstract class AbstractControllerTests2 {

    @Resource
    protected FilterChainProxy fcp;

    @Autowired
    protected MockServletContext sc;
    protected MockMvc mvc;

    protected OperationContextListener scl;

    @Before
    public void init() {
        sc.addInitParameter("defaultHtmlEscape", getDefaultHtmlEscape());
    }
}

```

» [Expand](#)

[source](#)

```
sc.addInitParameter("contextConfigLocation", getContextConfigLocation());
sc.addInitParameter("enterpriseld", getEnterpriseld());
sc.addInitParameter("configURI", getConfigURI());
sc.setContextPath(getContextPath());
doInit();
scl = new OperationContextListener();
scl.contextInitialized(new ServletContextEvent(sc));
mvc = getMockMvc();
}

{@After
public void destroy() {
    scl.contextDestroyed(new ServletContextEvent(sc));
    Operation.destroy();
}

protected StartupConfiguration getStartupConfiguration() {
    return StartupConfigurations.defaultOnlineConfigBuilder(getConfigURI())
        .enterpriseld(getEnterpriseld())
        .contextRoot(getContextPath())
        .springContext(getApplicationContext()).build();
}

protected MockMvc getMockMvc() {
    return MockMvcBuilders.standaloneSetup(controller()).addFilter(fcp).build();
}

protected String getDefaultHtmlEscape() {
    return "true";
}

protected void doInit() {

protected abstract Object controller();

protected abstract String getEnterpriseld();

protected abstract String getConfigURI();

protected abstract ApplicationContext getApplicationContext();
protected abstract String getContextConfigLocation();
```

```
    protected abstract String getContextPath();  
}
```

- AbstractFinchControllerTests is the Abstract base class for any finch Spring MVC controller

```
AbstractCoreOperationInitTests  
@ContextConfiguration(locations = {"classpath: META-INF/config/application-context-h2_2.xml",  
                           "classpath: web/WEB-INF/config/webmvc-config_2.xml",  
                           "classpath: META-INF/config/security-context_2.xml"})  
@WebAppConfiguration("src/main/web")  
@DbUnitConfiguration(databaseConnection = "GLOBAL")  
public abstract class AbstractFinchControllerTests extends AbstractControllerTests {  
}
```

› Expand

[source](#)

- AbstractInfControllerTests is the Abstract base class for any finch Spring MVC controller

```
AbstractInfControllerTests  
@DatabaseSetup("classpath: data/finch-inf-test-data_2.xml")  
public abstract class AbstractInfControllerTests extends AbstractFinchControllerTests {  
}
```

› Expand

[source](#)

- AbstractWebControllerTest is the Abstract base class for any finch Spring MVC controller

EmployeeRepositoryTests

› Expand

```
@ContextConfiguration(locations = {"classpath: META-INF/config/application-context-h2_2.xml",
    "classpath: web/WEB-INF/config/webmvc-config_2.xml",
    "classpath: META-INF/config/security-context_2.xml"})
@DbUnitConfiguration(databaseConnection = "GLOBAL")
@WebAppConfiguration("src/main/web")
public abstract class AbstractWebControllerTest extends AbstractControllerTests2 {
    @Autowired
    private ApplicationContext context;
    protected String getContextConfigLocation() {
        return "classpath: META-INF/config/security-context_2.xml";
    }
    protected String getEnterpriseId() {
        return "FIN";
    }
    protected String getConfigURI() {
        return "classpath: META-INF/config/application-context-h2_2.xml";
    }
    @Override
    protected String getContextPath() {
        return "finch-web";
    }
    @Override
    protected ApplicationContext getApplicationContext() {
        return context;
    }
}
```

Source

3.4.7.1 Testing Guidelines

3.5 Testing Phase

3.5.1 Functional Testing

Introduction

QA team is responsible for testing the Application functionalities. The Functional Testing Scope is still not fully defined for finch. The goal of functional testing is to detect faults that can only be exposed by testing the entire integrated system or some major part of it based on some business scenario.

The functional Testing is same as of Black-Box (may vary from project to project as per requirement) testing, which is generally prepared based on the Business Requirement Specification. Usually It covers the following:

- Business scenarios (only applicable for finch client for business logic implemented from their end)
- Operation flow

The objective of this test is to ensure that each element of the application meets the functional requirements of the business as outlined in the Use Cases or Requirement document. The major points of the functional test cover entry of the input, processing the input and

processing in downstream components. The functional test includes various patterns based on business scenario and operation. In some projects, cases of storage (database), generating output messages (interfacing between components) , menu access control at user level, multilevel and concurrency scenarios are considered during the functional test phase.

Scope of Testing

Functional testing scope is defined in the following areas for finch:

- Authentication and authorization
- Basic infrastructural data model
- Abstraction for business service, DAO layer
- Dashboard
- Abstraction for Trade Entry, Amend, Cancel UI
- Abstraction for Trade Query
- Abstraction for console and online reports
- Abstraction for console batch and service processes
- File Upload

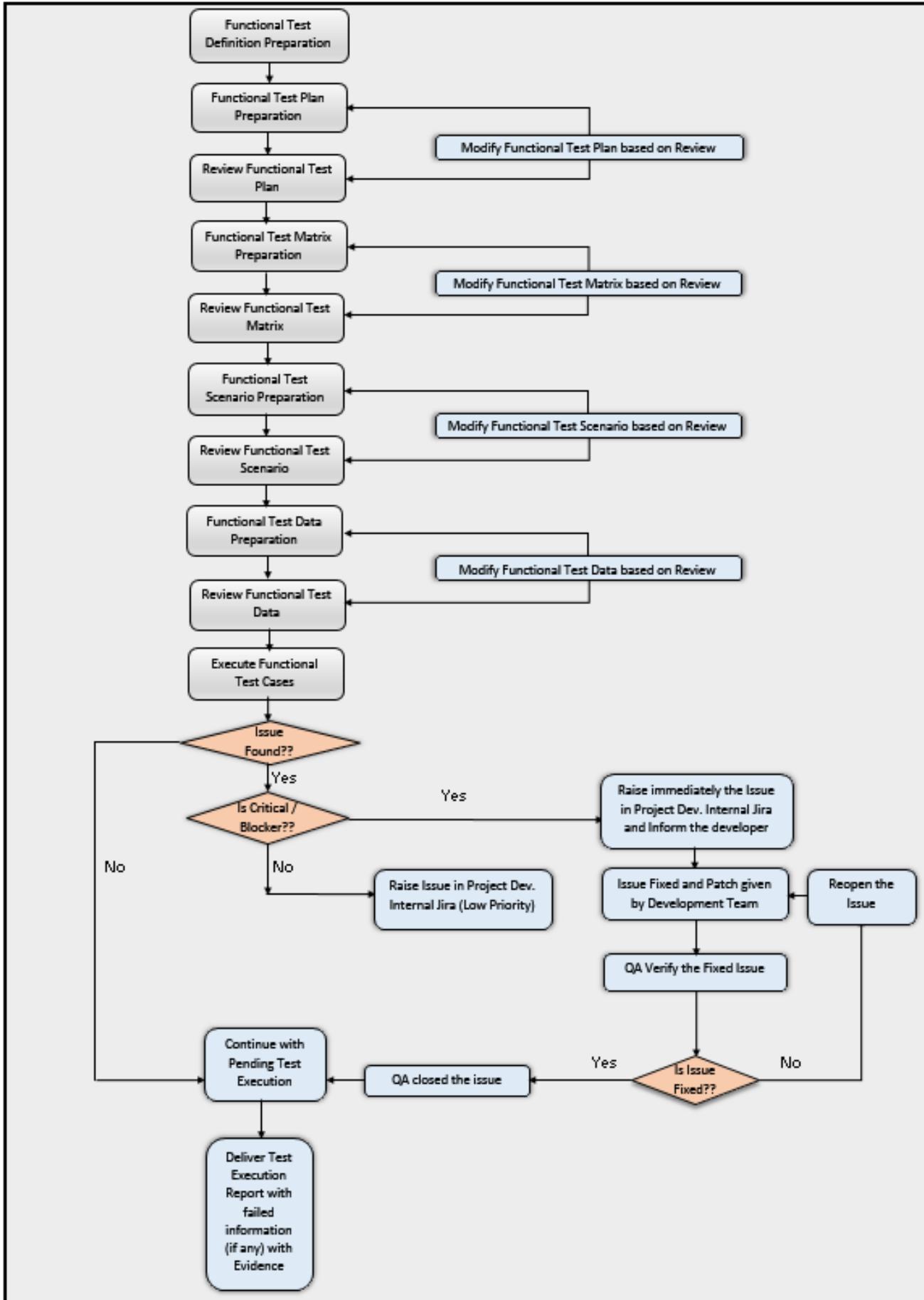
There is no such Business Scenario Specification implemented from finch end. Business scenario is to be implemented from finch client's end as per the requirement.

Functional Test Life Cycle

After the business logic is implemented from the Project end (as per requirement), the following set of activities are considered during functional testing phase:

- Functional Test Plan Preparation
- Test Definition Preparation
- Functional Test concept
- Test Matrix Preparation (If required)
- Test Scenarios Preparation
- Test Data Preparation
- Test Execution
- Test Evidence
- Test Evaluation / Review
- Logging of Issues (Bug Report)
- Fixing of Issues
- Retesting of Issues

Functional Testing Life Cycle Diagram



Functional Test Plan

A strategy document is prepared by QA team containing the details of systematic approach those are used to test an application. The plan contains a detailed understanding of the eventual workflow during functional testing.

Test Definition Preparation

Document containing guideline intended to establish the quality, performance or reliability of the application.

Functional Test Concept

Document containing the details on business scenario and operation.

Test Matrix

Document prepared by BA/QA team containing all the possible combinations of business scenario and operation in matrix format.

Test Scenarios

Document to be prepared from Test Matrix, which contains step by step details of each functional test scenario.

Test Data

Document containing data set, which is used to test the application to confirm the expected result.

Test Execution

QA team involves in test execution and in the actual testing activities to confirm the expected result based on test scenarios.

Test Evidence

QA team provides test evidences for failed test cases with screenshots and proper step to reproduce it in JIRA as logged issue. In some cases passed test cases are also considered as evidences.

Test Evaluation/Review

Concerned PIC evaluates the completeness of test suite consisting of test cases and results. This activity is conducted to ensure that the most important functional aspects and business scenarios have been fully covered during testing.

Logging of Issues

QA team records bug/defects in JIRA with test result and proper steps if any deviation from expected result found during testing. In case of Critical/Blocker issue immediate action should be taken from project development team.

Fixing of Issues

Developer team works on the logged bugs/defects according to the priority and update the status in JIRA after the issue is resolved.

Retesting of Issues

QA team retests the logged bugs/defects after the issues have been fixed by developer and update the status of logged issues accordingly in JIRA till closure.

Pre-Assumption

- Screens to be unit tested by developers are complete.
- All major issues found during unit testing and QA testing (in each sprint) are resolved and fixed.
- Functional Test Cases to be prepared based on Use Cases/Requirement document available in Confluence.
- Followings are the responsibilities of PIC of the respective Project
 - Functional Test Plan preparation
 - Test Matrix preparation(wherever applicable)
 - Test Scenarios preparation
 - Test Data preparation
 - Test Execution
 - Test Evidence preparation
 - Logging of Issues
 - Retesting after Fixing of Issues
- Followings are the responsibilities of concerned PIC, which should be decided by the respective PM:
 - Test Definition document preparation
 - Functional Test concept document preparation
 - Test Evaluation review

Features Out of Testing Scope

System Integration testing (SIT) feature is out of testing scope. Even though this is being unplanned and needs to be carried out by the tester considering the following aspects for finch:

- Data state with integration layer
- Data state within the database layer
- Data state within the Application layer

For the time being this is not planned.

Note

- Only some basic feature for Multi Enterprise Testing (like accessing menu/screen..etc.) would be under the scope of testing.
Exhaustive data testing will be out of the scope
- Multi Office will be out of the scope of testing (As it is out of finch scope)

Test Input

Existing usecase and requirement document available in confluence for the respective project.

Schedule

The schedule for functional testing is to be decided by PM as per requirement .

Test Deliverable for Functional Testing

Following items are included with functional test deliverable, which may vary as per Project requirement.

- Testcase having pass/fail status with proper steps and screenshot evidence.
- Bug Report for fail test cases with proper steps and screenshot evidence - At the time of functional testing, if a tester finds a bug, a Jira issue is to be raised with priority as any one of the following options:
 - Blocker
 - Critical
 - High
 - Major
 - Minor
- The priority is determined by the tester based on the nature of the issue. If due to the bug the application became unstable or it causes loss of data or it sets incorrect values or further testing is not possible etc., the tester may set the priority as Critical or Blocker. Tester may also consult the leads or manager before setting the priority. There is no such predefined guideline for raising the

issue as critical or non-critical. All the issues raised by the tester during functional testing will be communicated to the development team.

- Test Summary Report - The main items in Test Summary Report are as follows(may vary as per requirement):
 - Number of Testcases Remaining
 - Number of Testcases Executed
 - Number of Testcases Confirmed (Reviewed)
 - Total No. of Bugs Detected
 - Total No. of Bugs Resolved

The Test Summary report generally provided to the client based on some intervals.

Incident Report for Functional Testing

The following course of actions are considered during preparation of Incident Report:

- If any issue found during testing, a bug report is prepared and attached to the JIRA issue for the Bug. The bug is then assigned to the Development Lead.
- The developer does the root cause analysis for all the issues.
- Tester lists down all the bug issues found during particular cycle of the functional testing in the incident report.
- The event, cause and root cause information should be added for all the issues in the incident report.
- Updated Incident Report is shared with Client in regular interval.

Revision History

Documentation Control log.

Version	Release Date	Description of Changes	Author
1.0	11/07/2014	Initial Version	Saumya Dey

3.5.2 Non Functional Testing

Introduction

The purpose of finch2 is to create own framework with the latest art of technology and industry standards for quick bootstrapping every project benefiting the proven technology stack & best practices.

This Non Functional Testing will mainly focus on finch framework related testing from the client perspective.

Screen Layout , Configuration changing , Console / Batch and Dashboard features are the important parts with the other features of finch framework functionalities.

The testing will mainly focus on Internet Explorer 8.0 with the other browser like Firefox and Chrome.

Presently it caters to the requirements of the following clients:

- elibSys
- Mercury
- Iris
- EIG

Definition

The purpose of this test document is to address the testing scope, identifying testing phases, QA resources role and responsibilities, schedule and defect management for the FINCH2 Project.

Standards

No Critical ,Blocker or Major errors can be expected before release.

Critical ,Blocker and Major issue should be fixed with highest priority after reported in Finch Support JIRA or Internal Finch Dev. JIRA by Finch client or internal team respectively.

UI related standard, usability error checking / testing should also be a part of testing for the Finch users.

QA Role

- Testcases preparation based on the Finch updated feature (if any) referring the use cases or other related documents available in confluence for current Sprint (to be taken up before the release from Finch in each Sprint)
- Prepared Testcase execution on the Finch delivered build on sample-app. This activity would be taken up after the QA release from Finch in each Sprint (applicable for Finch Internal QA).
- Prepared Testcase execution on Finch features with the integrated project features. This activity would be taken up after the public release from Finch in each Sprint (applicable for the respective Finch Client Projects).
- Retest the Fixed issues.
- Build Verification/Smoke Testing on the day of release
- Test Plan preparation and update (if required).

Strategy

The project being in agile mode, testing for this project would, therefore be through agile methodology. Generally each iteration is composed of several sprints (not defined). Each sprint duration will be of 2 weeks.

Test Cycle

Let us consider sprint releases to be SPRINT1, SPRINT2… SPRINTn-1, SPRINTn

DEV-SP1 = Development First Sprint Release

DEV-SP2 = Development Second Sprint Release

DEV-SP3 = Development Third Sprint Release

Finch Manual QA-SP1 = QA First Sprint Release to Internal Manual Team

Finch Manual QA-SP2 = QA Second Sprint Release to Internal Manual Team

Finch Manual QA-SP3 = QA Third Sprint Release to Internal Manual Team

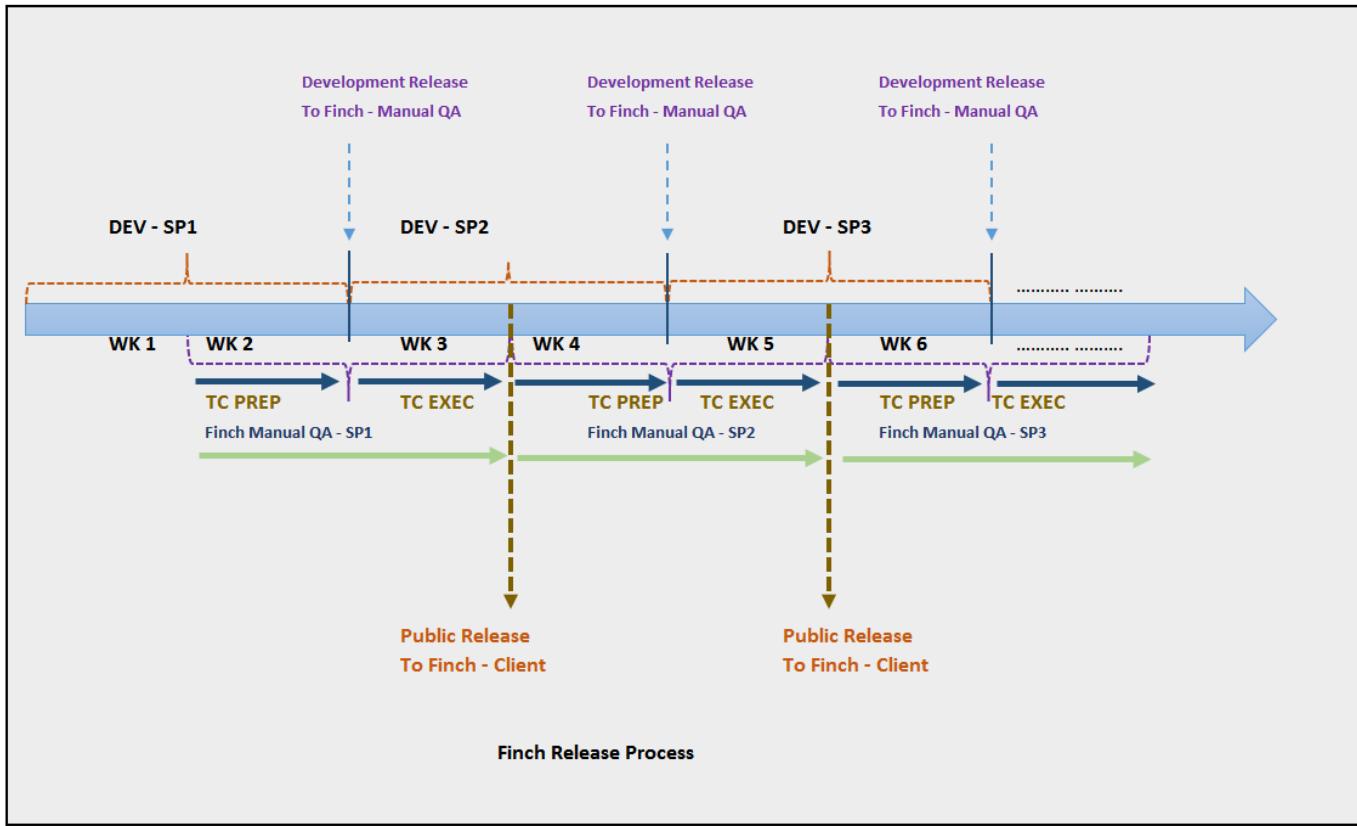
.....
W1 = 1st Week

W2 = 2 nd Week

...

...

Wk = K the Week
.....



Test Cycle Strategy

Manual testing duration is usually conducted for 2 weeks. Manual test case design begins during the period when development is in progress (normally from the second week onward in each sprint development phase). Before preparing test cases on current version/release, the usecases, requirement document (if any) and User Guide in Confluence should be properly updated (which is the input for those test cases).

Test cases are required to be reviewed by peers or superiors. In regular interval QA Checklist for Non-Functional UI Testing should be consulted and updated to ensure that application is following the industry standard guideline for web user interface testing.

All the issue verification for the fixed issues should be assigned to Internal Finch QA members wherever manual testing scope available.

Once the internal ‘Development Release to To Finch - Manual QA’ (e.g. DEV-SP1) is available, the manual test execution will begin with the prepared test cases on the delivered build to ensure the stability by Finch Internal Manual QA Team.

Internal Manual QA team will assure that the ‘Development Release to To Finch - Manual QA’ (delivered by Development

Team) by executing previously prepared Regression Test Cases fully/partially (by executing at least High and Mid Priority Test Cases) and doing Ad-hoc testing.

The issues found during testing will be mentioned against the test cases executed or new test cases will be added against the found issues in Jira (for which no manual test case exists).

On the last day of Manual Testing cycle smoke testing would be carried out on Final fixes of QA Release and Public Release will be assured with proper Release Note in confluence from Finch Internal QA's end.

The manual testing will end for that sprint(e.g. Finch Manual QA - SP1) with a 'Public Release To Finch - Client'.

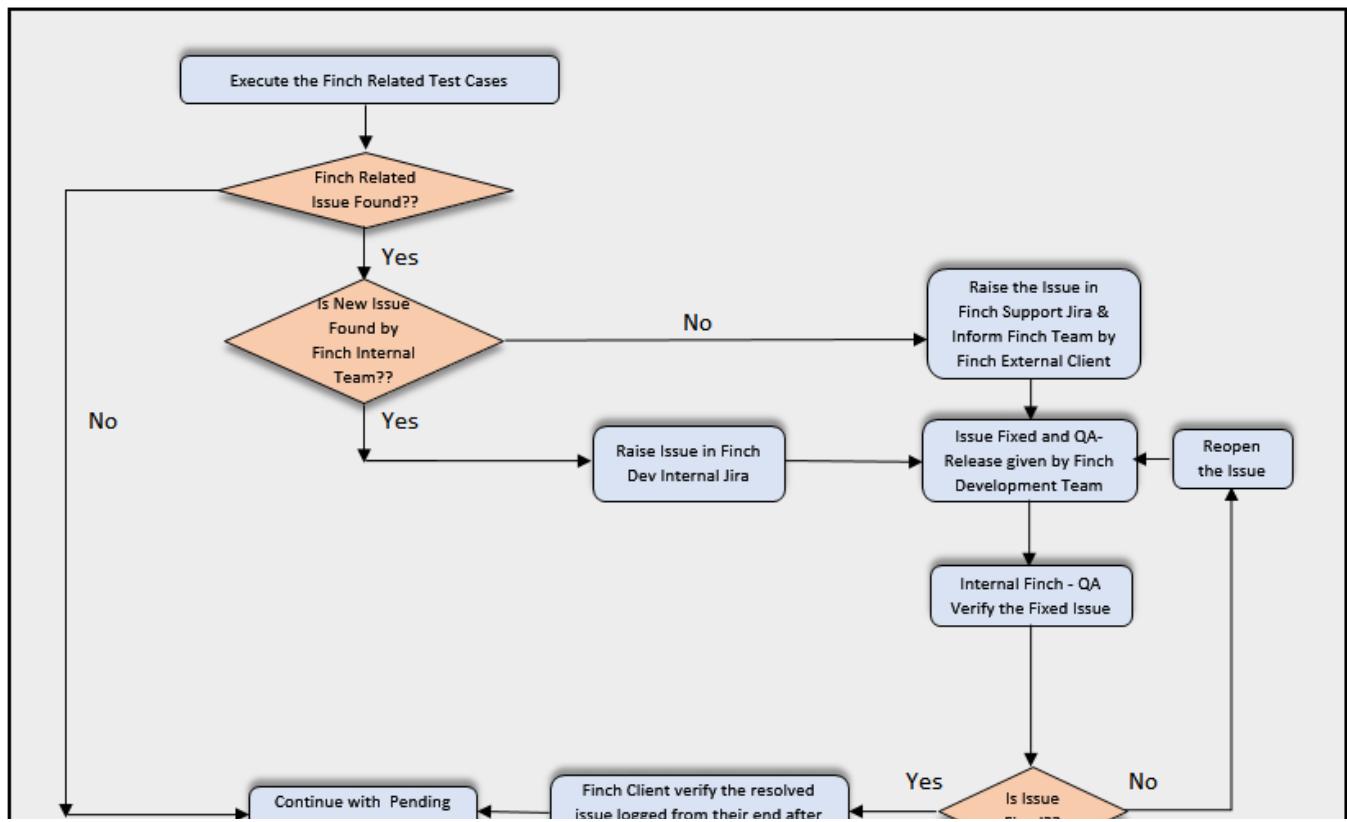
Once the Public Release is available to the 'Finch - Client' , they will do testing from their end after proper integration of sample-app to their system.

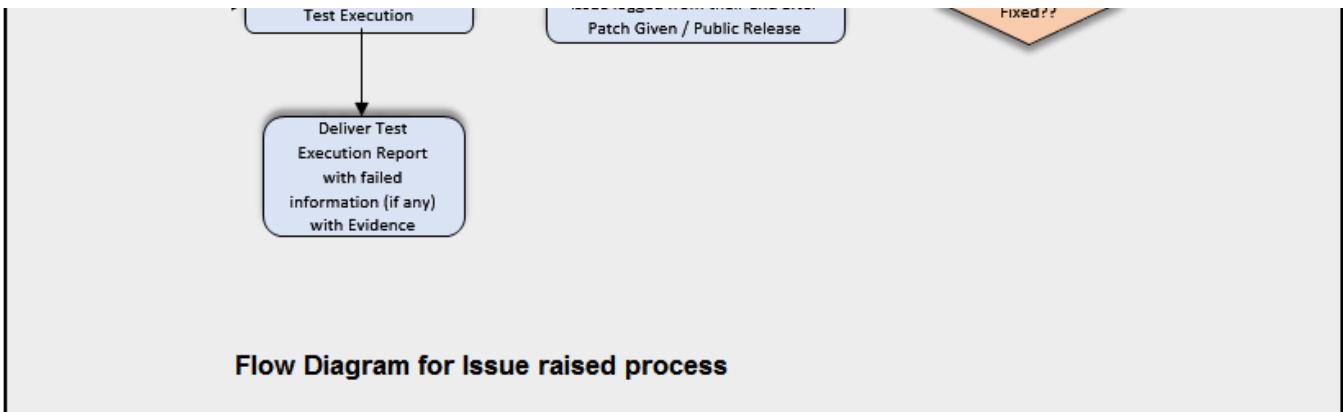
Defect Reporting

QA will track logged defects through JIRA resulting from the test execution.

The following activities are to be carried out during reporting defects:-

- Raise issue in JIRA with test result.
- Issue might be found by internal finch QA team or finch external client.
- If the Issue(s) found by the finch external client (sample-app) user then report the Issue (defects/bugs/improvement) with proper test steps and screen shots to development team by defining priority and assigning it to finch Project Lead (in finch supported Jira).
- Internal finch QA will log Issue(s) and assign to the finch Lead through finch Development JIRA (internal).
- Before Public Release given to finch external client, finch QA will retest the fixed bugs (wherever manual testing scope available) after they are fixed by developer.
- All medium and high severity bugs of last sprint must be fixed in the current sprint release.
- After Public Release, client will do the test execution from their end for the fixed issue(s) after proper integration of finch sample-app to their system.
- The entire flow has been depicted by the following diagram.





Scope of Non Functional Testing

Behavior of application on QA environment will be tested. The following items would come under the non functional testing scope.

Unit Testing

Unit testing will be carried out by developers according to the requirements implemented in the application.

Unit Interface Testing

Testing the various graphical user interfaces, how user interacts with application, testing application elements like fonts, layouts, navigation, button, menus, dialog boxes, icons, images, colors etc.

Boundary Value Testing

BVT to be executed for known data range for the fields as per sprint specific requirements across the application.

Integration Testing

Integration testing will also be carried out by the tester including all major functions.

Regression Testing

Regression testing will be carried out partially manually by executing at least high and mid priority test cases and gradually it covered through automation to ensure that all the functionalities implemented earlier are as they were (also for fixed bug/bugs).

System Integration Testing (SIT)

Even though this being unplanned needs to be carried out by the tester considering the following aspects:

- Data state with integration layer
- Data state within the database layer
- Data state within the Application layer

For the time being this is not planned.

Performance Testing

Huge volume of data need to be processed through the application to find out the performance of the application.

For the time being this is not planned.

Testing Data

The fields in the application are populated with the DB static data properly or not, this also needs to be checked.

Note

- Only some basic feature for Multi Enterprise Testing (like accessing menu/screen..etc.) would be under the scope of testing.
- Complete Data testing will be out of the scope.
- Multi Office will be out of the scope of testing (As it is out of finch scope).

Test Items

- User authentication
- Concurrency
- Dashboard
- Menu panel and screen involving Trade, Sys control & Reference
- Settings/Preferences
- Various widgets
- New Workspaces
- Holiday Calendar
- Few Configuration related changes
- Console/Batch Related tests

List of items are to be tested as part of testing (as of now):

Module Name	Functionality
User Authentication	User authorization & ensuring that unauthorized user having no access to the application.
Concurrency	Application Behavior when same/two user simultaneously modifying the same data
Dashboard	Various functionalities like creation of different widgets like Menu Shortcut, Business Feed, etc.
Menu panel	Various functionalities involving Trade, Sys Control & Reference
Settings/Preferences	User settings in terms of core & Dashboard with respect to the application.
Widgets	Creation & their functionalities
Workspaces	Creation & usage of the same
Configuration	Some configuration (xml,excel,csv or some other) file related minor changes
Console/Batch	Verify the console / batch behavior

Checklist for Non Functional Testing

Standard Checklist for testing can be found under the following link of Best Practice portal:-> <http://portal.nrifintech.com/portal/display/BET/2.+Standard+Checklist>

On periodic basis QA will execute the Checklist (items selectively from the available checklists , whatever is applicable for finch) for non-functional testing to ensure that application is following the industry standard guideline for web user interface testing.

Features out of Testing Scope

Load & performance testing is currently out of scope.

Test Constraint

Compiling of all source files in the testing machine.

Revision History

Documentation Control log.

Version	Release Date	Description of Changes	Author
1.0	11/07/2014	Initial Version	Saumya Dey

3.5.3 Automation

Automation testing in Finch Framework Using QTP

Introduction

Every software development group tests its products before delivery, yet delivered software always has defects. Test engineers strive to catch them before the product is delivered but they always creep in and they often reappear, even with the best manual testing processes. Automated software testing is the best way to increase the effectiveness, efficiency and coverage of software testing. Manual tests are repeated often during development cycles for source code changes and other situations like multiple operating environments and hardware configurations.

An automated testing tool is able to execute the predefined actions, compare the results to the expected behavior and report the success or failure of these manual tests to a test engineer. Once automated tests are created they can easily be repeated and they can be extended to perform tasks, which are impossible with manual testing. Automated software testing has long been considered critical for big software development organizations but is often thought to be too expensive or difficult for smaller companies to implement.

Purpose of Automation

- Automates software testing saves time and money
- Testing improves accuracy
- Increases Test Coverage

Objective of Automation Framework

Framework for Quick Test Professional (QTP) driven automation is designed and developed by NRI FT with the objective of automated testing web applications like finch, fam with minimum changes. It can be used for regression and sanity testing. The design approach is configurable. Thus if any new application / screen needs to be tested with this framework, then it can be implemented with configuration file changes (without modifying the code).

Technical Features of Automation Framework

- Multiple System / Application handling capability: Configuration file changes will be required for the same scripts to be executed for any finch web applications.
- **Keyword Driven Framework:** Keyword Driven Framework is a type of automation framework where for a particular test case, you would first identify a set of Keywords and then associate an action (or function) with each of these keywords. For e.g. ‘Click Button’ is a keyword which is associated with the function ‘Click_Button()’ which clicks a button during execution.
- **Descriptive Programming is used:** Descriptive Programming based test scripts are developed to reuse the same scripts for different applications. Descriptive Programming is a mechanism for creating tests where you use “Programmatic Description” of objects instead of recording them.

Guideline of Using QTP Framework

Deployment Instruction:

1. Checkout from CVS
2. Extract the QueryScreen.zip file

3. Extract the driver_script.zip file
4. Change the project name (e.g. FINCH) in Appliance.xml file and change the corresponding Node Value to Yes

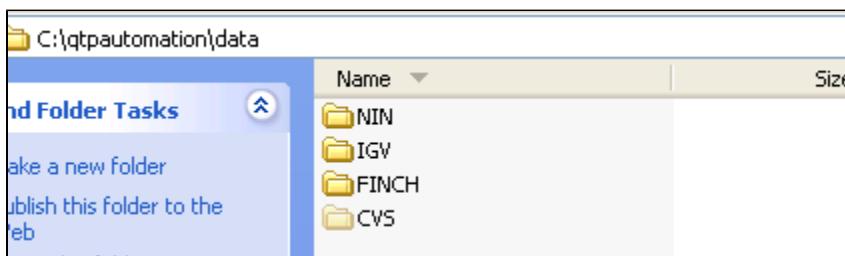
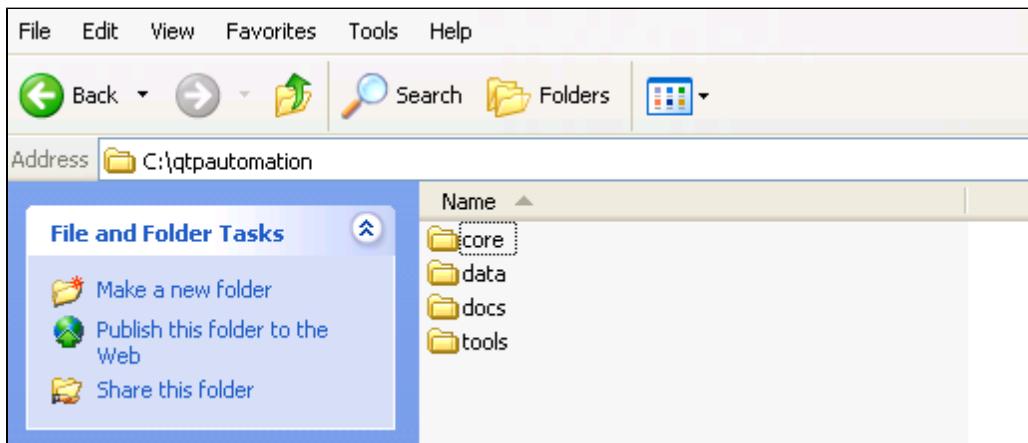
Environment setup instructions:

1. Windows XP 32 bit machine is required
2. QTP 11 should be installed in the machine
3. IE8 should be installed in the machine

Guidelines for preparing test cases in QTP

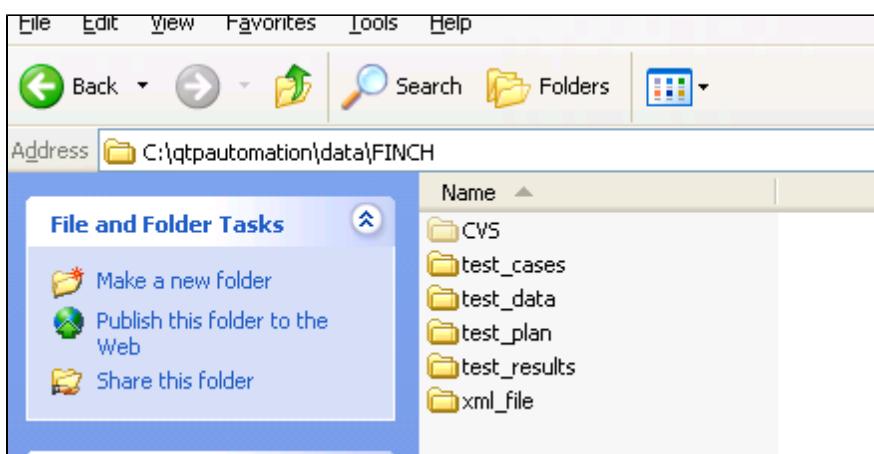
If the project starts from the scratch.

1. First create a folder with project name under data folder



(For an e.g. project name is Finch, folder is created with name FINCH)

2. Within the project folder create folder for test case, xml file, test data file, test plan file and test results folder

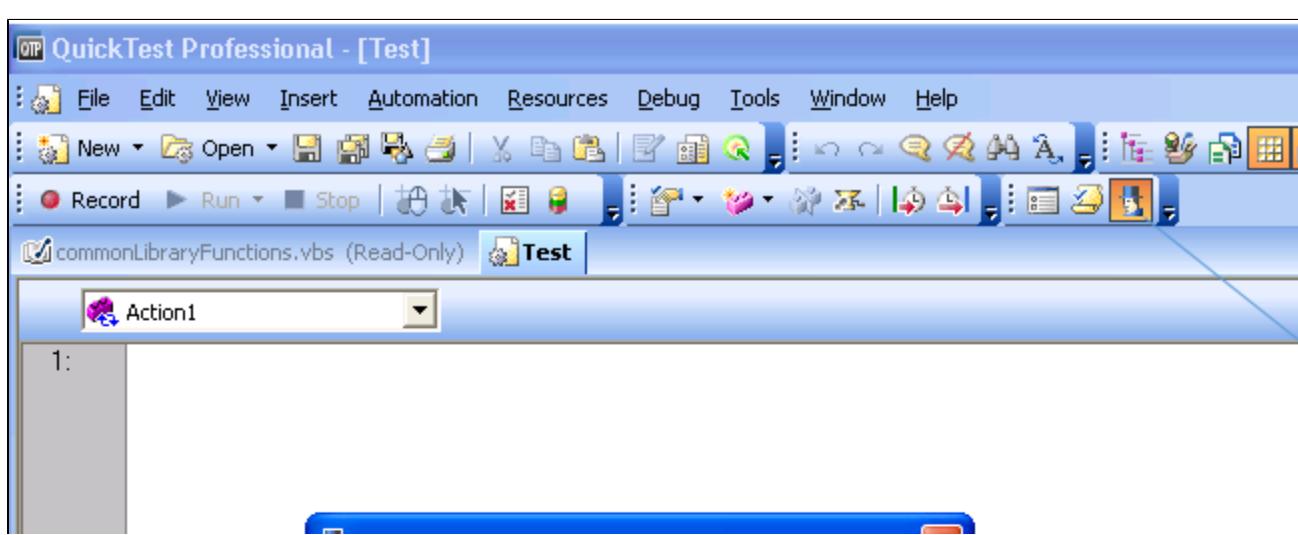


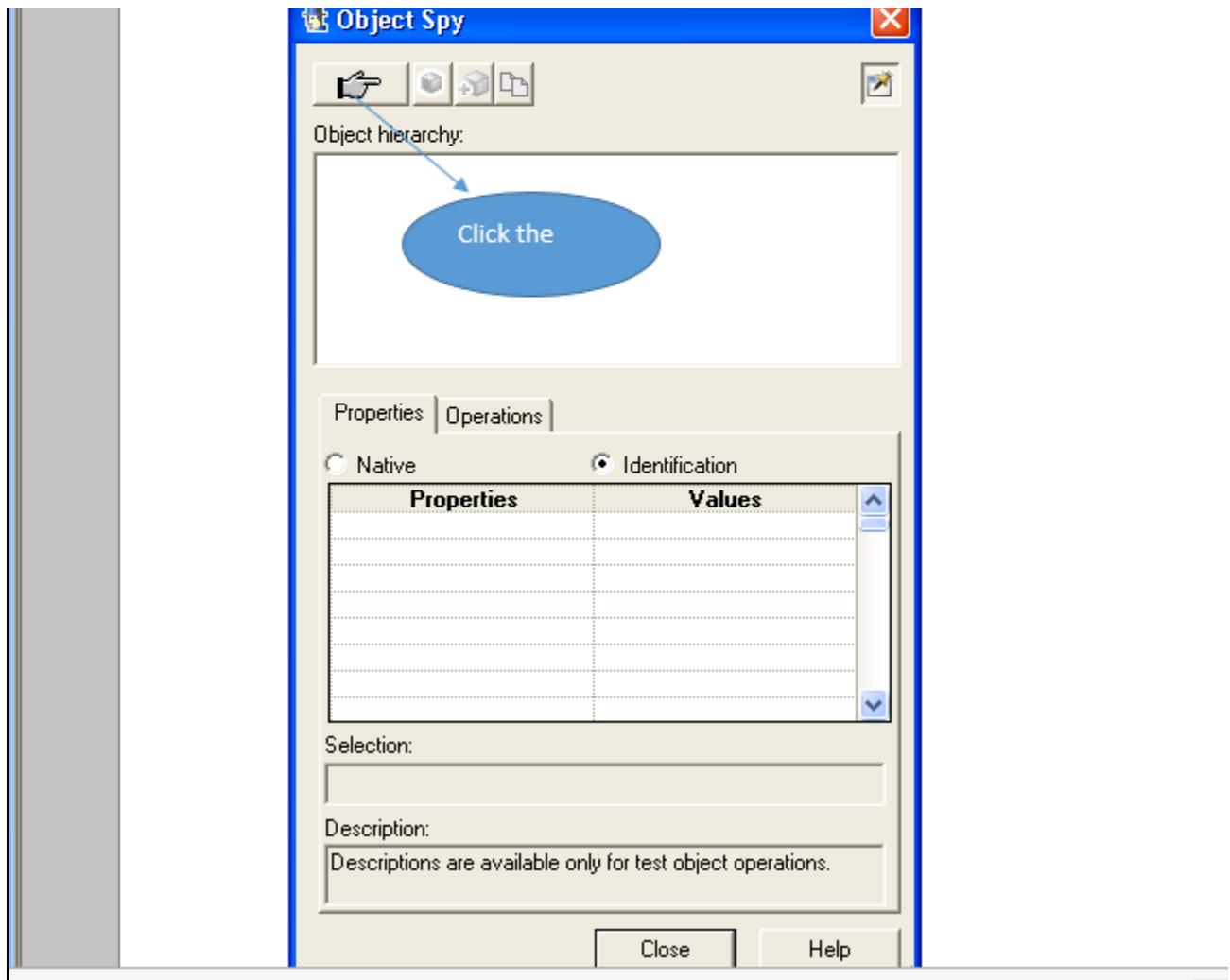
- test_cases folder should contain all the test cases that would be prepared for finch
- test_data folder should contain all the test data which will be required for executing the test cases
- xml_file folder should contain all the screen specific xml file and one master xml file(Screen specific xml file contains all the object property value which is required to identify an object in screen and master xml file contains all basic screen properties for an application)
- test_plan folder contain an excel file which is required for executing the desired test cases.
- test_results folder contains all the test results after test case execution is complete

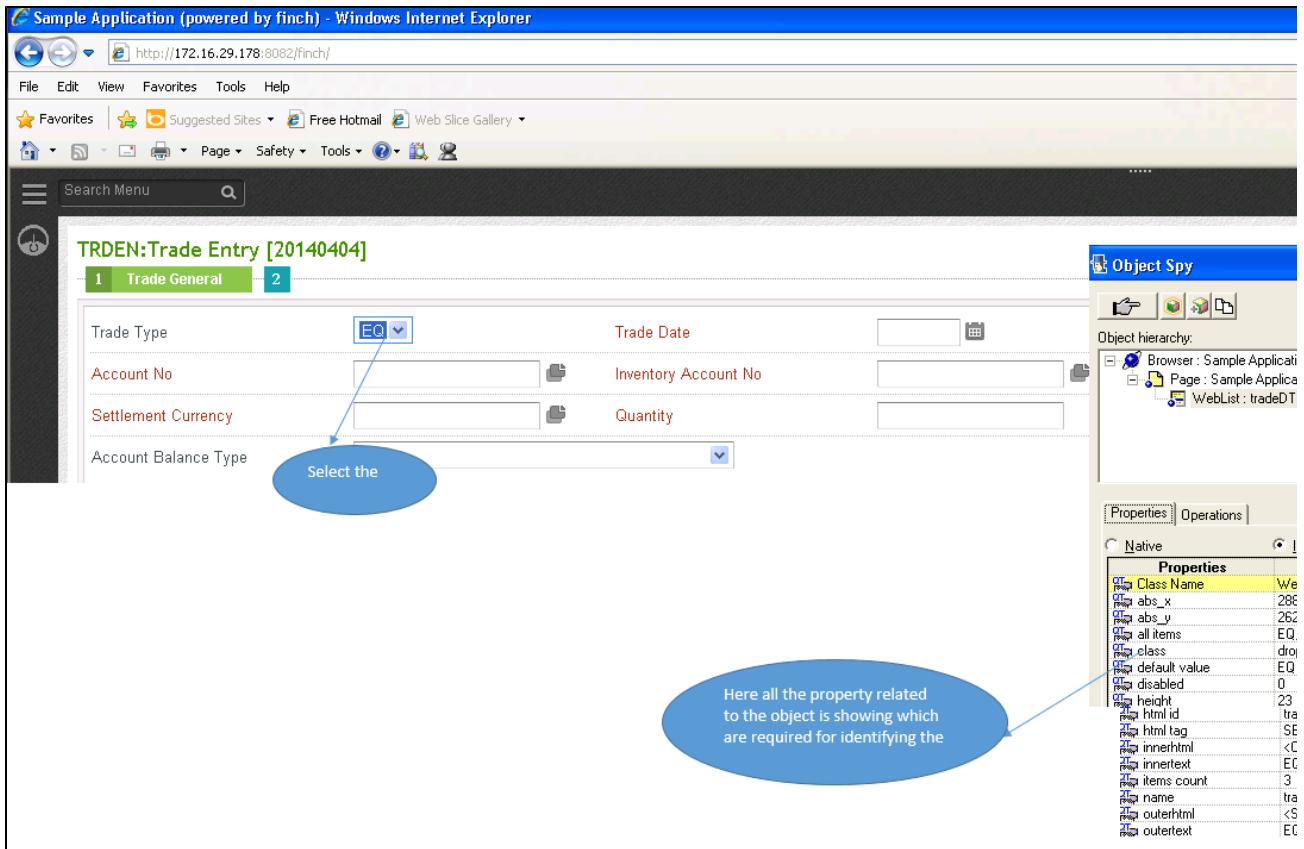
How to Prepare Master XML file and Screen Specific xml File

First create an xml file with project name(for e.g. finch.xml)

1. Go to any screen
2. Go to QTP
3. Click on object spy
4. Identify the object







5. Add the class and property for each object in Master xml (e.g finch.xml) file

```
<?xml version="1.0" encoding="UTF-8"?>
<objects>
<object1 strClassname="WebButton" strPropertyName="name">Button</object1>
<object2 strClassname="WebEdit" strPropertyName="html id">TextBox</object2>
<object3 strClassname="Weblist" strPropertyName="name">dropdownList</object3>
<object4 strClassname="WebCheckBox" strPropertyName="name" strPropertyName1="value">Check</object4>
<object5 strClassname="WinButton" strPropertyName="text">DialogButton</object5>
<object6 strClassname="Link" strPropertyName="href" strPropertyName1="outertext">MenuLink</object6>
<object7 strClassname="WebElement" strPropertyName="class">ButtonUp</object7>
</objects>
```

6. Add the property value in screen specific xml file(for e.g. trdEntry.xml)

Mapping between master and screen specific XML file:

Finch.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objects>
    <object1 strClassname="WebButton" strPropertyName="name">Button</object1>
    <object2 strClassname="WebEdit" strPropertyName="html id">TextBox</object2>
    <object3 strClassname="Weblet" strPropertyName="name">DropDownList</object3>
    <object4 strClassname="WebCheckBox" strPropertyName="name" strPropertyName1="value">C
    <object5 strClassname="WinButton" strPropertyName="text">DialogButton</object5>
    <object6 strClassname="Link" strPropertyName="href" strPropertyName1="outerText">Menu
    <object7 strClassname="WebElement" strPropertyName="class">ButtonUp</object7>
```

trdEntry.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objects>

    <browser title="Sample Application \ (powered by finch\).* ">Browser</browser>
    <page title="Sample Application \ (powered by finch\)">Page</page>
    <DashBoard class="dashboardName-inner">DashBoard</DashBoard>
    <MenuLink href="trd/entry/init\?screenId\=TRDEN&versionNo\=1&componentId\=1">MenuLink</MenuLink>
    <Verify_Label strPropertyValue="LABEL" strPropertyValue1="Trade Type|Trade Date|Value Date|Account No|Inventory Account No|Security Code|Buy Sell Orientation|Execution Market|Account Balance Type|Next|Submit|Confirm|OK|Back|Reset">Verify_Label</Verify_Label>
    <TextBox id="tradeDate">Trade Date</TextBox>
    <TextBox id="valueDate">Value Date</TextBox>
    <TextBox id="accountNo">Account No</TextBox>
    <TextBox id="invAccountNo">Inventory Account No</TextBox>
    <TextBox id="securityCode">Security</TextBox>
    <DropdownList name="tradeDTO.tradeType">Trade Type</DropdownList>
    <DropdownList name="tradeDTO.buySellOrientation">Buy_Sell_Orientation</DropdownList>
    <DropdownList name="tradeDTO.executionMarket">Execution_Market</DropdownList>
    <DropdownList name="tradeDTO.accountBalanceTypeStr">Account_Balance_Type</DropdownList>
    <Button name="Next" intPos="0">Next</Button>
    <Button name="Submit" intPos="0">Submit</Button>
    <Button name="Confirm" intPos="0">Confirm</Button>
    <Button name="OK" intPos="0">OK</Button>
    <Button name="Back" intPos="0">Back</Button>
    <Button name="Reset" intPos="0">Reset</Button>
```

How to Prepare Test Case

Attributes of test cases

- Depending on the ToBeExecuted flag, step will be executed (If the flag is yes step will be executed, if the flag is no step will not be executed)
- Mention the TestCaseID in test case id field.
- In the Action column mention the action required to perform a step (e.g. such as open Browser it will Open the browser, Click button it will click a button)
- If required mention the screen specific xml file name in XMLFileName column
- Node name is used for identifying the type of object (If required)
- Node value is used for identifying the screen objects corresponding to the specified node name (If required)
- Node attribute contains the property required for identifying the object (If required)
- Command string field is required for sending the command for console or in some cases sending the name of the file to be saved at run time (If required)

Sample test case

ToBeExecuted	TestCaseID	Action	XMLFileName	NodeName
Yes	TC_TradeEntry_ATC001	Open Browser		
Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox
Yes	TC_TradeEntry_ATC001	Click Button	Login	Button
Yes	TC_TradeEntry_ATC001	DashboardExist	trdTradeEntry	DashBoard
Yes	TC_TradeEntry_ATC001	Click Menu	trdTradeEntry	MenuLink

How the XML files are mapped in test case

ToBeExecuted	TestCaseID	Action	XMLFileName
Yes	TC_TradeEntry_ATC001	Open Browser	
Yes	TC_TradeEntry_ATC001	Set Value	Login
Yes	TC_TradeEntry_ATC001	Set Value	Login
Yes	TC_TradeEntry_ATC001	Click Button	Login
Yes	TC_TradeEntry_ATC001	DashboardExist	trdTradeEntry
Yes	TC_TradeEntry_ATC001	Click Menu	trdTradeEntry

trdTradeEntry.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<objects>

<browser title="Sample Application \powered by finch\.*">Browser</browser>
<page title="Sample Application \powered by finch\)">Page</page>
<DashBoard class="dashboardName-inner">DashBoard</DashBoard>
<MenuLink href="trd/entry/init\?screenId\=TRDEN&versionNo\=1&componentId\=TRD" >
<Verify_Label strPropValue="LABEL" strPropValue1="Trade Type|Trade Date|Value Date|Buy|!

```

```

<TextBox id="tradeDate">Trade_Date</TextBox>
<TextBox id="valueDate">Value_Date</TextBox>
<TextBox id="accountNo">Account_No</TextBox>
<TextBox id="invAccountNo">Inventory_Account_No</TextBox>
<TextBox id="securityCode">Security</TextBox>
<DropdownList name="tradeDTO.tradeType">Trade_Type</DropdownList>
<DropdownList name="tradeDTO.buySellOrientation">Buy_Sell_Orientation</DropdownList>
<DropdownList name="tradeDTO.executionMarket">Execution_Market</DropdownList>
<DropdownList name="tradeDTO.accountBalanceTypeStr">Account_Balance_Type</DropdownList>
<Button name="Next" intPos="0">Next</Button>
<Button name="Submit" intPos="0">Submit</Button>
<Button name="Confirm" intPos="0">Confirm</Button>
<Button name="OK" intPos="0">OK</Button>
<Button name="Back" intPos="0">Back</Button>
<Button name="Reset" intPos="0">Reset</Button>

```

Prepare Automation Test Case

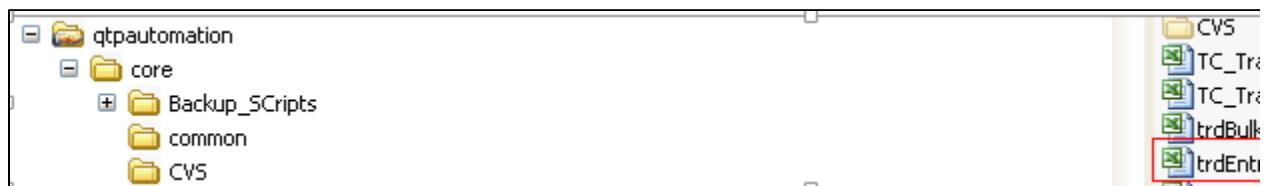
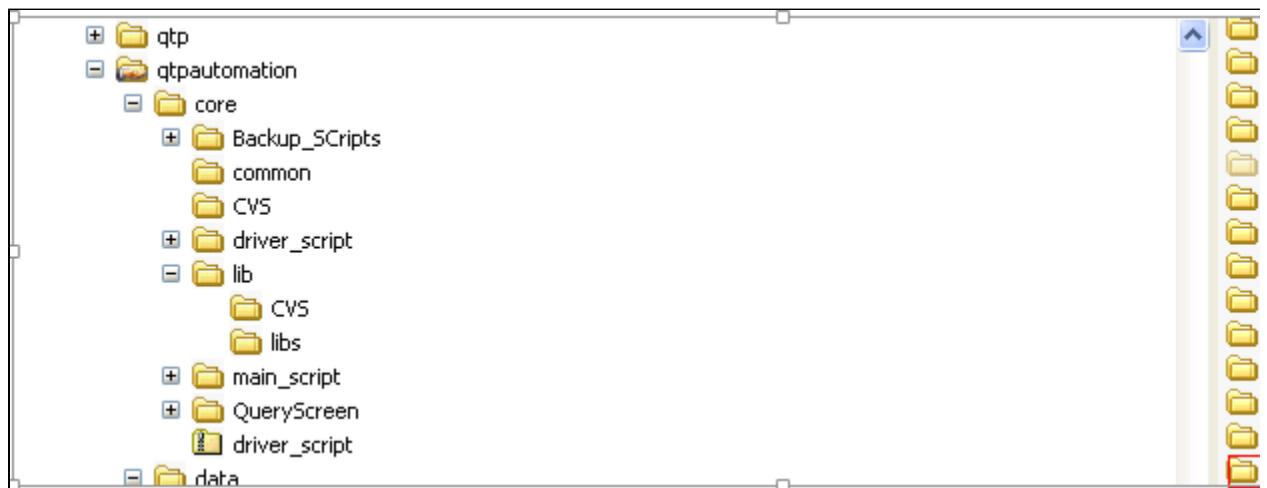
- Open template.xls file
- Using the template.xls file, create the test case file
- Mention the ToBeExecuted flag (Yes/No)
- Mention the TestCaseID
- Mention the XMLFileName
- Mention the Action which is required to perform a step
- Mention the NodeName ,NodeValue and NodeAttribute if it is required to perform an action

Sample Test Case for Login

ToBeExecuted	TestCaseID	Action	XMLFileName	NodeName
Yes	TC_Login_ATC001	Open Browser		
Yes	TC_Login_ATC001	Set Value	Login	TextBox
Yes	TC_Login_ATC001	Set Value	Login	TextBox
Yes	TC_Login_ATC001	Click Button	Login	Button
Yes	TC_Login_ATC001	DashboardExist	Dashboard	DashBoard
Yes	TC_Login_ATC001	Verify After Login	Dashboard	VerifyAfterLogin
Yes	TC_Login_ATC001	Click Button	Dashboard	ButtonUp
Yes	TC_Login_ATC001	Close Browser		

Prepare Test Data

1. Create a folder in test_data folder in module name(Suppose test data is required to be prepared for trade entry screen,create a folder name trd)
2. Within the trd folder create a xls file for trade entry test data(where all the data required for trade entry should be kept in for an e.g. trdEntry.xls)



Sample TradeEntry test case

ToBeExecuted	TestCaseID	Action	XMLFileName	NodeName
Yes	TC_TradeEntry_ATC001	Open Browser		
Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox
Yes	TC_TradeEntry_ATC001	Click Button	Login	Button
Yes	TC_TradeEntry_ATC001	DashboardExist	trdTradeEntry	DashBoard
Yes	TC_TradeEntry_ATC001	Click Menu	trdTradeEntry	MenuLink
Yes	TC_TradeEntry_ATC001	Verify Controls And Values	trdTradeEntry	DropdownList
Yes	TC_TradeEntry_ATC001	Select Value	trdTradeEntry	DropdownList
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Select Value	trdTradeEntry	DropdownList
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Select Value	trdTradeEntry	DropdownList
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Set Value	trdTradeEntry	TextBox
Yes	TC_TradeEntry_ATC001	Select Value	trdTradeEntry	DropdownList
Yes	TC_TradeEntry_ATC001	KeyBoardShortCut	trdTradeQuery	Enter_Shortcut_Key
Yes	TC_TradeEntry_ATC001	Verify Field Data	trdTradeEntry	Verify_Field_Data

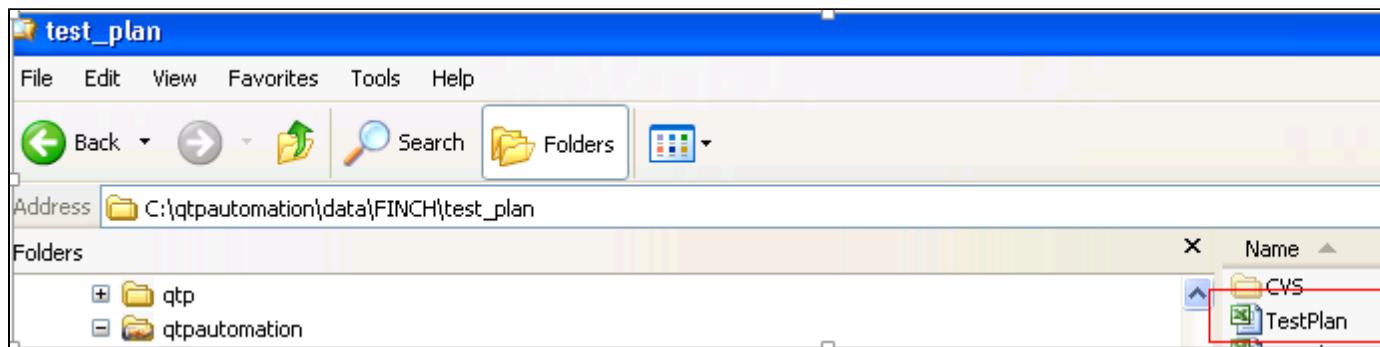
Sample Test data corresponding to test case id

While preparing the test data file, mention the TestCaseID and mention the node value in the column header and put the test data in the specified column, test data is selected depending on the TestCaseID and NodeValue

Microsoft Excel - trdEntry															
Type a question for help															
A2	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	TestCaseID	Trade_Date	Value_Date	Account_No	Inventory_Account_No	Security	Quantity	Trade_Currency	Settlement_Currency	Price	External_Ref_No	Trade_Type	Buy_Sell_Orientation	Execution_Market	Account_Balance_Type
2	TC_TradeEntry_ATC001	20131204	20131206	121212004	1401014	IBM	30	JPY	INR	20	1234 FI	Client Buy	BR001	01-Cash collateral for margin trades	

Prepare Test Plan

Within project specific folder(e.g. Finch) create a folder "test_plan" and within the folder create a file "TestPlan.xls"



Sample TestPlan.xls file

	A	B	C	D	E	F
1	ToBeExecuted	TestName	ActionName	TestCaseName	TestDataName	CheckPointFlag
2	Yes	QueryScreen	QueryScreen	TC_Trade_Entry_ATC001	trdEntry	
3	No	QueryScreen	QueryScreen	TC_Account_Query_ATC004	acnAccountQuery	
4	No	QueryScreen	QueryScreen	TC_Account_Query_ATC005	acnAccountQuery	
5	No	QueryScreen	QueryScreen	TC_Account_Query_ATC006	acnAccountQuery	

Attributes in TestPlan.xls file

- Depending on the ToBeExecuted flag test case will be executed.
- TestName and ActionName column refers to the QTP test name and action name
- TestCase name column contains the testcase id which is required to be executed
- TestDataName column contains the datasheet required for executing the test case
- CheckPointFlag is required for database checkpoint and xml checkpoint (If the CheckPoint flag is Yes then database checkpoint and xml checkpoint works,data need to be prepared for it)

Mapping Between the Test Case Test Data and Test Plan File

Microsoft Excel - TestPlan

This table contains test plan data with columns: A, B, C, D, and E.

	A	B	C	D	E
1	ToBeExecuted	TestName	ActionName	TestCaseName	Te...
2	Yes	QueryScreen	QueryScreen	TC_Trade_Entry_ATC001	trd...
3	No	QueryScreen	QueryScreen	TC_Account_Query_ATC004	acr...
4	No	QueryScreen	QueryScreen	TC_Account_Query_ATC005	acr...
5	No	QueryScreen	QueryScreen	TC_Account_Query_ATC006	acr...

Microsoft Excel - trdEntry

This table contains trade entry data with columns: A, B, C, D, E, and F.

	A	B	C	D	E	F
1	TestCaseID	Trade_Date	Value_Date	Account_No	Inventory_Account_No	Security
2	TC_TradeEntry_ATC001	20131204	20131206	121212004	1401014	IBM

Microsoft Excel - TC_TradeEntry_ATC001

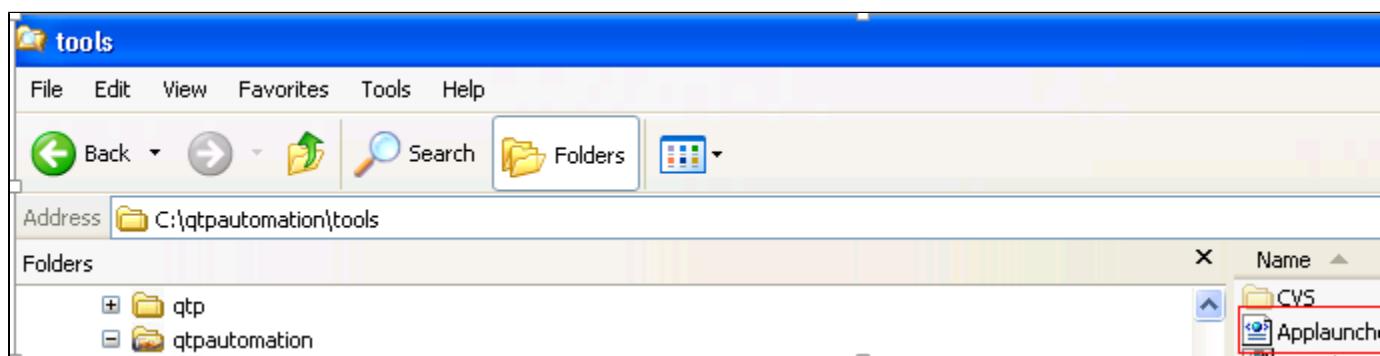
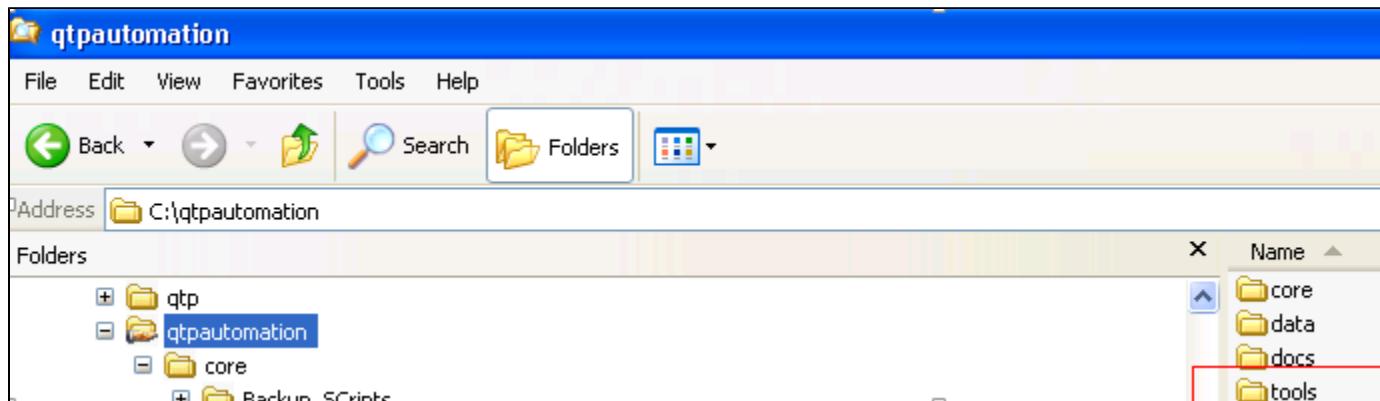
This table contains specific actions for the trade entry case with columns: A, B, C, D, and E.

	A	B	C	D	E
1	ToBeExecuted	TestCaseID	Action	XMLFileName	NodeName
2	Yes	TC_TradeEntry_ATC001	Open Browser		
3	Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox
4	Yes	TC_TradeEntry_ATC001	Set Value	Login	TextBox

5 Yes	TC_TradeEntry_ATC001 Click Button	Login	Button
---------	-------------------------------------	-------	--------

How to Execute the Test Case

1. Before executing the test case go to tools folder
2. Open the Applauncher.xml file
3. Create test_results folder



Applauncher.xml file

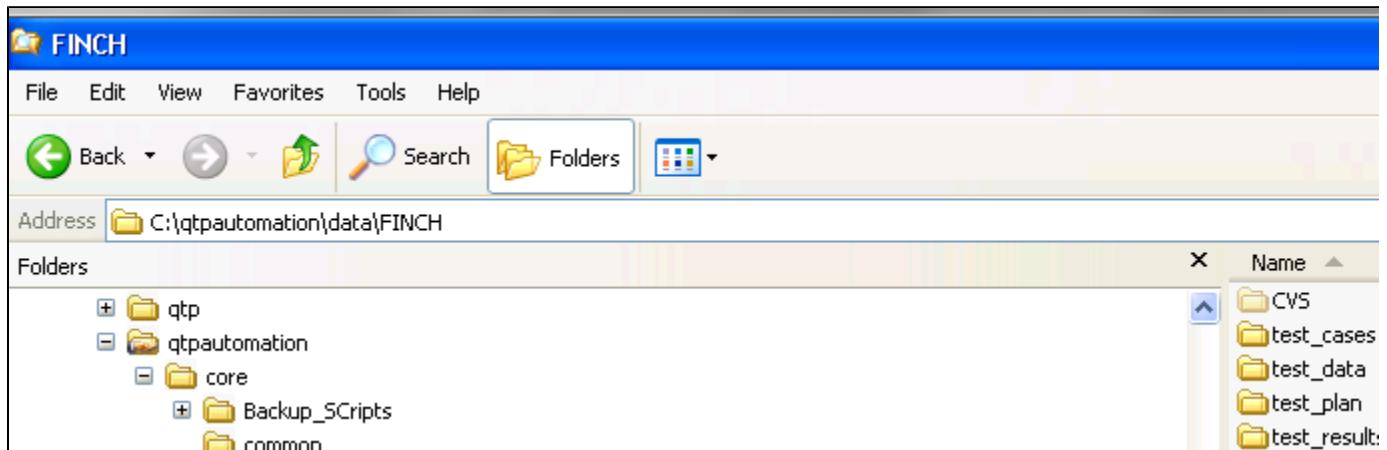
```
<?xml version="1.0" encoding="UTF-8"?>
<Projects>
<Project projectname="FINCH" url="http://10.1.6.75:8090/finch/" imageCapture ="Yes" Baseline="Yes"
Validate="Yes" BaselineOverwrite="No" SendEmail="Yes" Emailid="abc@gmail.com">Yes</Project>
</Projects>
```

Attributes of the Applauncher.xml file

- projectname attribute has the value of project names
- url attributes value should be url of application
- imageCapture attributes value should be Yes or No(If it is marked as yes then imagecapture report will be generated in the test result folder, if it is marked as No then no imagecapture report will be generated)
- Baseline attributes have the value of Yes or No (If the value is yes then a baseline.xls file will be generated in runtime)
- Validate attributes have the value of Yes or No (If the value is yes then a validate.xls file will be generated in runtime, if baseline file exists)
- BaselineOverwrite attributes have the value of Yes or No(If the value is yes then a baseline.xls file will be generated in runtime, it will overwrite the existing excel file)
- SendEmail have the value of Yes or No(If the value is Yes then an email is generated after test case execution complete)
- Emailid have the value of listed email id(Email will be sent to the desired user mentioned in attribute value
If the node value of the node name Project is Yes then test case will be executed for that project)

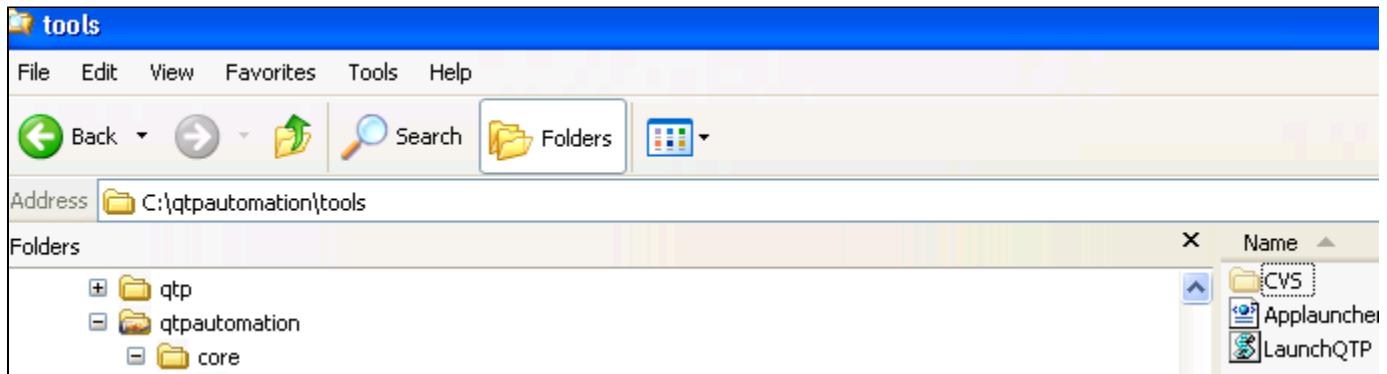
Create Test Result Folder

Create a folder test_results in Project name(Finch) folder

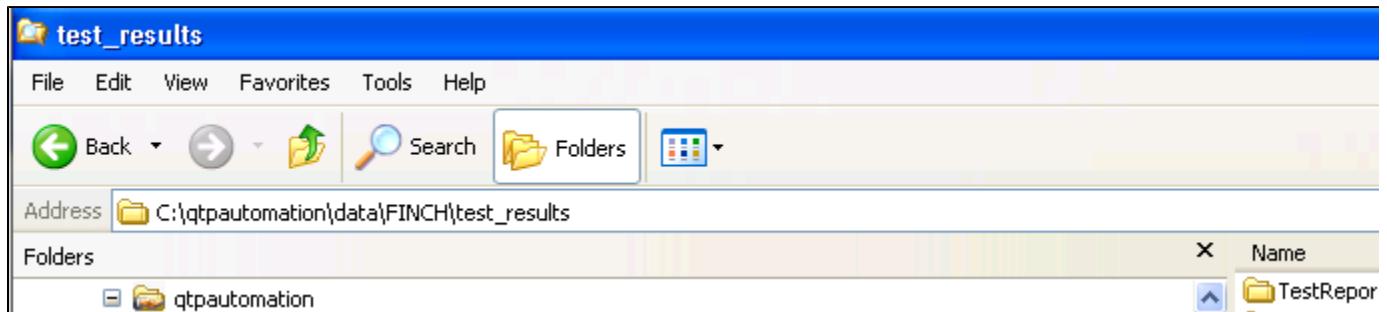


Execute the Test Case

1. Go to Tools folder
2. Double click on LaunchQTP.vbs file



Test case will be executed after double clicking the executable file, test results will be generated in TestReport .



The actions which are required for preparing test cases is attached in [List_of_Actions_in_QTP.xls](#) file

3.5.5 Analysis Report From Sonar

Introduction

This page will explain about creation of Analysis reports from [Sonar Cube](#).

References:

[User Guide](#) and [Administration](#).

Description:

To Create Analysis Report from Sonar Cube following required:

- Set the Analysis Properties
- Run the Sonar Analysis
- Check the Report in Dashboard

Analysis Properties

Parameters to configure project analysis set in multiple places. Here is the hierarchy of parameters:

- Global analysis parameters, defined in the UI, apply to all the projects (From the top bar, go to Settings > General Settings)
- Project analysis parameters, defined in the UI, override global parameters (At a project level, go to Configuration > Settings)
- Project analysis parameters, defined in a project analysis configuration file or an analyzer configuration file, override the ones defined in the UI
- Analysis / Command line parameters, defined when launching an analysis, override project analysis parameters

Note that only parameters set through the UI are stored in the database.

For example, if we override the sonar.profile parameter via command line for a specific project, it will not be stored in the database. Local analyses in Eclipse, for example, would still be run against the default quality profile.

Note that the list of parameters below is not exhaustive. The property keys shown in the interface, at both global and project levels, can also be set as analysis parameters.

Mandatory Parameters

Server

Key	Description	Default Value
sonar.host.url	Server URL	http://localhost:9000

Database

Key	Description	Default Value
sonar.jdbc.url	JDBC Connection URL	jdbc url default is <code>jdbc:h2:tcp://localhost:9092/sonar</code>
sonar.jdbc.username	User for the JDBC Connection	jdbc user name default is sonar

sonar.jdbc.password	password for the JDBC Connection	jdbc password default is sonar
---------------------	----------------------------------	-----------------------------------

Project Configurations

Key	Description	Default Value
sonar.projectKey	The project key that is unique for each project. Set through <groupId>:<artifactId> when using Maven.	
sonar.projectName	Name of the project that will be displayed on the web interface. Set through <name> using Maven.	
sonar.projectVersion	The project version. Set through <version> when using Maven.	
sonar.language	Set the language of the source code to analyze. If not set, a multi-language analysis will be triggered.	
sonar.sources	Comma-separated paths to directories containing source files. If not set, the source code is retrieved from the default Maven source code location.	

Optional Parameters

Project Configuration

Key	Description	Default Value
sonar.projectDescription .	The project description. Not compatible with Maven, which uses the <description> attribute	NA for maven
sonar.binaries	Comma-separated paths to directories containing the binary files (directories with class files, in the case of Java). Not compatible with Maven, which retrieves binaries from the default location for Java Maven projects.	
sonar.tests	Comma-separated paths to directories containing tests. Not compatible with Maven, which retrieves test from the default location for Java Maven projects.	
sonar.libraries	Comma-separated paths to files with third-party libraries (JAR files in the case of Java). Patterns can be used. Example: sonar.libraries=path/to/specific/library/myLibrary.jar,path/to/library/*.jar Note that the * wildcard character is not supported for directories, only for files. This property is used by rule engines during issues detection (mainly the SonarQube and FindBugs engines, which both rely on bytecode). Having the bytecode of these libraries allows the rules engines to get more information on coupling, possible null parameters when calling external APIs, etc., thus getting more accuracy during issue detection.	

sonar.analysis.mode	<p>Set the analysis mode. See Concepts.</p> <p>Possible values:</p> <ul style="list-style-type: none"> analysis : Standard way to analyze the source code. The source code is analyzed and measures and issues are pushed to the SonarQube database. The results of the analysis can be browsed through the web interface preview : The source code is analyzed but the measures and issues are not pushed to the SonarQube database. Therefore, they cannot be browsed through the web interface. This mode can be used with the Issues Report plugin, which generates an HTML issues report to local file. incremental :Same as Preview mode but only new or modified files (compared to the latest version available on the remote server) are analyzed 	Analysis
sonar.preview.readTimeout	<p>This property is only relevant in the context of preview analysis. In a preview analysis certain information about the project is downloaded from the server into a local database. This property is the timeout value in milliseconds for the reading of that data. Typically the default value is fine, but it may need adjusting in very large or busy environments.</p>	60000
sonar.sourceEncoding	<p>Set the source file encoding. Encoding of source files. Example of values: UTF-8, MacRoman, Shift_JIS. This property can be replaced by the standard property project.build.sourceEncoding in Maven projects.</p>	System encoding
sonar.importSources	<p>Allow or suppress the import of the text of source files into SonarQube. For security or other reasons there are times when project sources must not be stored and displayed.</p> <p>Set this value to false to prevent the text of a project's source files from being available via the SonarQube interface to anyone at all.</p>	true

sonar.projectDate	<p>Assign a date to the analysis.</p> <p>Note: This parameter is applicable to a few, special use cases, rather than being an "every day" parameter:</p> <ul style="list-style-type: none"> When analyzing a new project, we may want to retroactively create some history for the project in order to get some information on quality trends over the last few versions. When moving from one database engine to another, it is highly recommended (even mandatory) to start from a fresh new database schema. In doing so, we will lose the entire history for all our projects. Which is why we may want to feed the new SonarQube database with some historical data. <p>To answer those use cases, we can use the sonar.projectDate property. The format is yyyy-MM-dd, for example: 2014-11-30.</p> <p>The process is the following:</p> <ul style="list-style-type: none"> Retrieve a the oldest version of our application's source that we wish to populate into the history (from a specific tag, whatever). Run a SonarQube analysis on this project by setting the sonar.projectDate property. Example: sonar-runner -Dsonar.projectDate=2014-11-30 Retrieve the next version of the source code of your application, update the sonar.projectDate property, and run another analysis. <p>And so on for all the versions of our application we're interested in</p> <p>Note: We must analyze our versions in chronological order, oldest first.</p>	Current date
sonar.branch	Manage SCM branches. Two branches of the same project are considered to be different projects in SonarQube.	
sonar.profile	Override the profile that would normally be used to analyze a project. Through the web interface, we define as many quality profiles as we want, and we associate one of these quality profiles to a given project though the web interface.	Default profile for the given language
sonar.skipDesign	Skip the computation of design metrics and dependencies.	false
sonar.working.directory	Set the working directory for an analysis triggered with the SonarQube Runner Path must be relative and unique for each project. Beware: the specified folder is deleted before each analysis.	.sonar

Exclusions / Inclusions

See Narrowing the Focus to:

- Exclude files from analysis
- Prevent some files from being checked for duplications
- Prevent some files from being taken into account for code coverage by unit tests and integration tests
- Ignore issues on certain components and against certain coding rules

Analyzer's Log

Key	Description	Default value
sonar.showProfiling	Display logs to see where the analyzer spends time.	false
sonar.showSql	Display all the SQL queries executed by the analyzer.	false
sonar.showSqlResults	Display the results of all the SQL queries executed by the analyzer.	false
sonar.verbose	Activate DEBUG mode for the analyzer.	false

Reports

All reports are accessible using following [link](#)

1. Dashboard : <http://finch-release:7000/sonarqube/dashboard/index/1>
2. Hotspots : <http://finch-release:7000/sonarqube/dashboard/index/1?did=2>
3. Issues: <http://finch-release:7000/sonarqube/dashboard/index/1?did=3>
4. Time Machine: <http://finch-release:7000/sonarqube/dashboard/index/1?did=4>

3.5.5.1 Narrowing the Focus

Introduction

This page will explain how to narrow the focus to analyse the project (i.e. precisely configuring what to analyze for the project)

Description:

If SonarQube's results aren't relevant, developers will push back on using it. That's why precisely configuring what to analyze for each project is a very important step.

Doing so allows us to remove noise, like the issues and duplications marked on generated code, or the issues from rules that aren't relevant for certain types of objects.

SonarQube gives us several options for configuring exactly what will be analyzed.

We can

- completely ignore some files or directories
- exclude files/directories from Issues detection (specific rules or all of them) but analyze all other aspects
- exclude files/directories from Duplications detection but analyze all other aspects
- exclude files/directories from Coverage calculations but analyze all other aspects

Ignoring Files

There are four different ways to narrow our analysis to the source code i.e. exclusion generated code, source code from libraries, etc combination of all together tune the analysis scope.

Source Directories

sonar.sources property limits the scope of the analysis to certain directories

Skipping Modules

Some project modules should not be analyzed and consolidated with global project measures.

For instance: sample modules, integration tests modules, etc.

To exclude those modules, at the project level, go to Configuration > Settings > Exclusions > Files and set the Module Exclusions property.

The format is a comma-separated list of modules: finch-test,finch-test-infra. If a module's artifactId differs from its module name (the directory name), use the artifactId instead of the module name

We can also work the other way around with inclusions by setting the Module Inclusions property to a list of only those modules we want analyzed.

Be careful: the root project must be added to the list.

File Suffixes

Most language plugins offer a way to restrict the scope of analysis to files matching a set of extensions. Go to Settings > General Settings > LanguagePlugin and set the File suffixes property

Excluding Files

It is possible to exclude specific files from analysis. At the project level, go to Configuration > Settings > Exclusions > Files and set the:

- sonar.exclusions property to exclude source code files
- sonar.test.exclusions property to exclude unit test files

Ignore Issues

To ignore any issues on certain components and against certain coding rules. Go to Configuration > Settings > Exclusions > Issues.

Ignore Issues on files

Set Ignore all issues on files that contain a block of code matching a given regular expression.

Ignore Issues in Blocks

Ignore all issues on specific blocks of code, while continuing to scan and mark issues on the remainder of the file.

Ignore Issues on Multiple Criteria

We can ignore issues on certain components and for certain coding rules.

e.g.

- We want to ignore all issues on all files => *;**/*
- we want to ignore all issues on java script program scripts/some.js => *;script/some.js
- we want to ignore all issues on jspx program pages/page.jspx => *;pages/page.jspx

Restrict Scope of Coding Rules

The restriction the application of a rule to only certain components, ignoring all others.

Ignore Duplications

To prevent some files from being checked for duplications.

To do so, go to Settings > General Settings > Exclusions > Duplications and set the Duplication Exclusions property. See the Patterns section for more details on the syntax.

Ignore Code Coverage

To prevent some files from being taken into account for code coverage by unit tests.

go to Settings > General Settings > Exclusions > Code Coverage and set the Coverage Exclusions property. See the Patterns section for more details on the syntax.

Patterns

Paths are relative to the project base directory.

The following wildcards can be used:

Wildcards	Matches
*	zero or more characters

**	zero or more directories
?	a single character

3.5.5.2 Creating Coding Rules

Introduction

The Quality Profiles service is the heart of SonarQube, since it is where we define our requirements by defining sets of coding rules (ex: Methods must not have a complexity greater than 10).

This page will explain how to create coding rules for the project

Requisite:

Quality Profile will be get created by user belongs to administrative group of sonar for [detail](#) and [Quality Profile](#).

Description

How to Configure rules available in [Configuring Rules](#) page.

3.5.5.3 Sonar Report

Introduction

This document will explain Sonar Reports;

Description:

Open Sonar Dashboard page : <http://finch-release:7000/sonarqube/dashboard/index/1>

Will display Dashboard reports

Dashboard

[Hotspots](#)

[Issues](#)

[Time Machine](#)

TOOLS

[Components](#)

[Issues Drilldown](#)

[Design](#)

[Libraries](#)

[Compare](#)



Version 2.1.12.1 - Nov 12 2014 08:54 Time changes...

Lines of code
55,348 ↗
103,994 lines ↗
20,350 statements ↗

Files
707 ↗
186 directories
751 classes ↗
4,185 functions ↗
2,438 accessors ↗

Duplications
8.3%
8,680 lines ↗
272 blocks ↗
98 files ↗

Complexity
2.1 /function
11.5 /class
12.3 /file
Total: 8,665 ↗

Issues
4,069 ↗

Technical Debt
167d ↗

Blocker 4
Critical 38 ↗
Major 2,480 ↗
Minor 1,460 ↗
Info 79 ↗

Package tangle index
12.4% ↗
> 61 cycles

Dependencies to cut
26 between packages
67 between files

Unit Tests Coverage
73.2% ↗
76.4% line coverage ↗
60.4% branch coverage ↗

Unit test success
100.0%
0 failures
0 errors
2,915 tests ↗
22 skipped ↗
14:56 min ↗

Events All

Date	Version
Nov 12 2014	2.1.12.1
Oct 20 2014	2.1.11.0
Sep 30 2014	2.1.10.0
Sep 22 2014	2.1.9.0
Sep 05 2014	2.1.8.0
Aug 26 2014	2.1.7.0
Aug 14 2014	2.1.6.0
Jul 28 2014	2.1.5.0

Show more

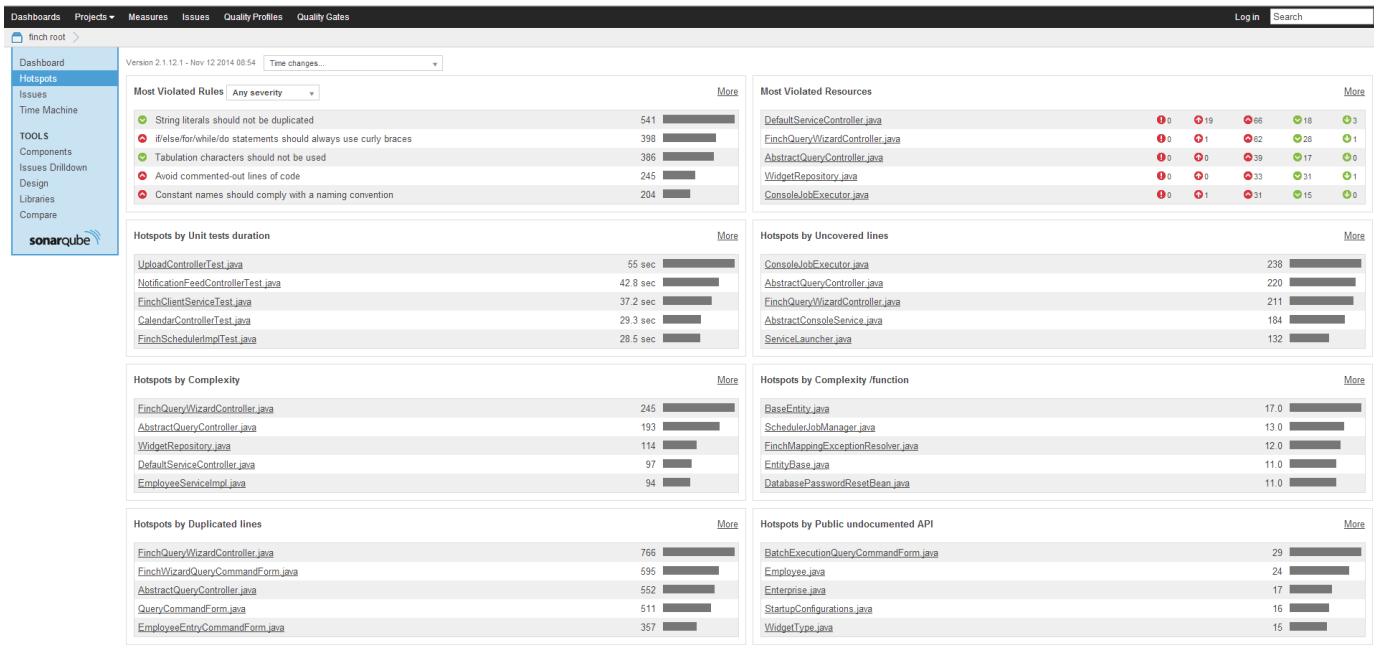
finch Root

Key:
Links:

- com.mifl.finch.finch-root
- Developer connection
- Issues
- Sources

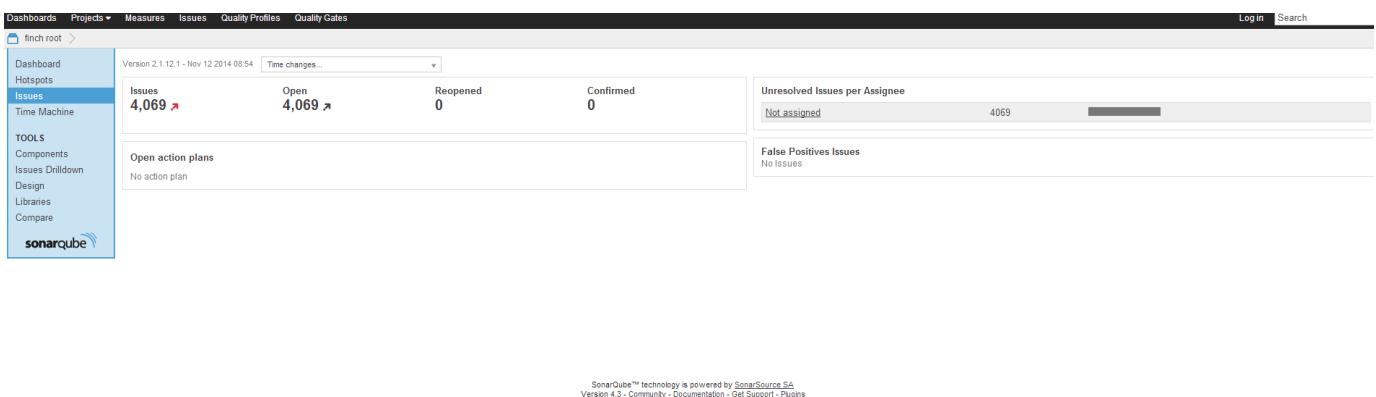
Click Hotspots in Left : <http://finch-release:7000/sonarqube/dashboard/index/1?did=2>

Will display Hotspots reports



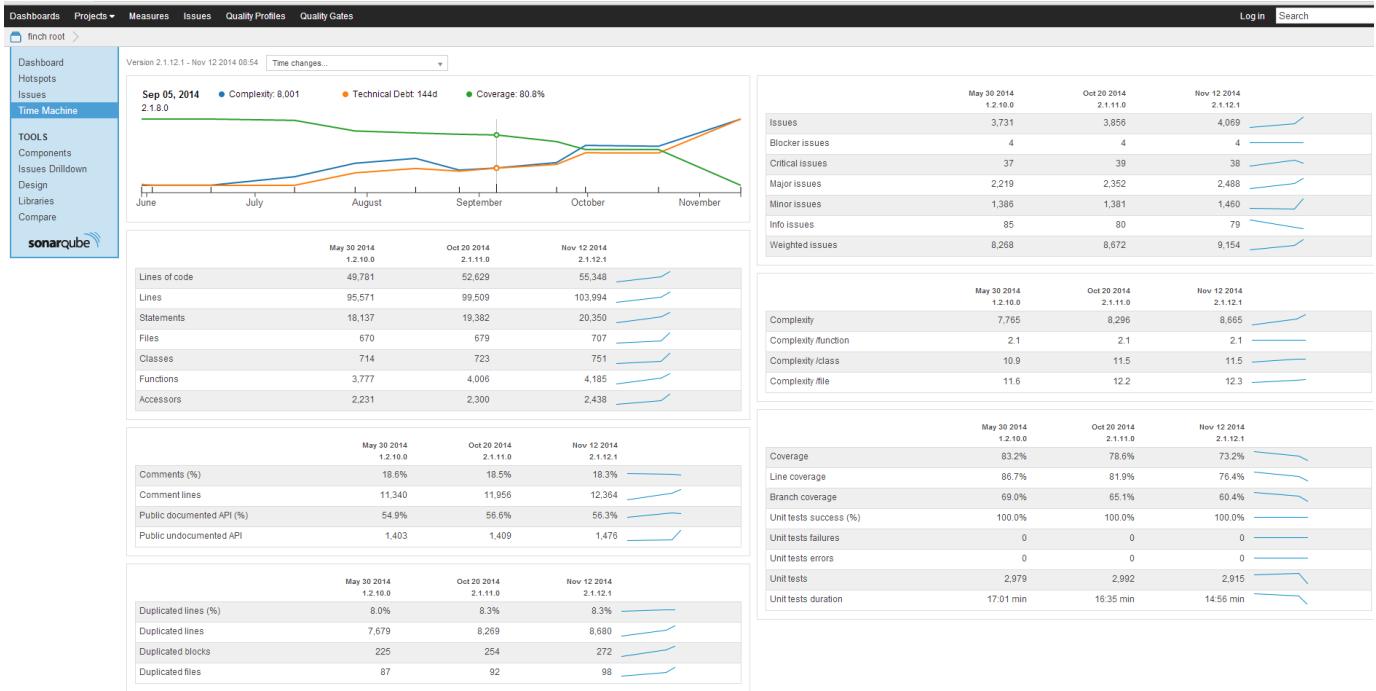
Click Issues in Left : <http://finch-release:7000/sonarqube/dashboard/index/1?did=3>

Will display Issues reports



Click Time Machine in Left : <http://finch-release:7000/sonarqube/dashboard/index/1?did=3>

Will display Time Machine reports



Any point required can be drilled down to source code level via clicking the link of the same:

Tools

Components:

click Components under Tools section of Left Pane:

finch root

Dashboard Hotspots Issues Time Machine **Components** Issues Drilldown Design Libraries Compare sonarqube

Name	Coverage	Build time	Links
finch_root	73.2% ↘	Nov 12 2014	🔗
Name	Coverage	Build time	Links
finch_Batch	39.9% ↘	Nov 12 2014	🔗
finch_Client_API	80.6%	Nov 12 2014	🔗
finch_Console_Infrastructure	56.4%	Nov 12 2014	🔗
finch_Core	81.3% ↘	Nov 12 2014	🔗
finch_DB	81.4% ↘	Nov 12 2014	🔗
finch_Notification	70.0% ↘	Nov 12 2014	🔗
finch_Scheduler	67.3% ↘	Nov 12 2014	🔗
finch_Upload	85.5%	Nov 12 2014	🔗
finch_Web_Infrastructure	72.4% ↘	Nov 12 2014	🔗

Clicked Finch Batch

Name	Coverage	Build time	Links
finch Batch	39.9%	Nov 12 2014	
Name			
src/main/java/com/nrift/finch/inf/batch	78.7%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/batch/metadata	0.0%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/batch/scanner	85.6%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/batch/service	30.7%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/batch/visitors	95.5%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/batch/chui	100.0%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/console	3.8%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/console/batch	40.5%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/console/batch/server	41.9%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/domain/entity	53.3%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/domain/repository	26.7%	Nov 12 2014	
src/main/java/com/nrift/finch/inf/domain/service	9.1%	Nov 12 2014	
src/test/java/com/nrift/finch/inf/batch		Nov 12 2014	
src/test/java/com/nrift/finch/inf/batch/scanner		Nov 12 2014	
src/test/java/com/nrift/finch/inf/batch/server		Nov 12 2014	
src/test/java/com/nrift/finch/inf/batch/service		Nov 12 2014	
src/test/java/com/nrift/finch/inf/console		Nov 12 2014	
src/test/java/com/nrift/finch/inf/console/batch/server		Nov 12 2014	
src/test/java/com/nrift/finch/inf/domain/entity		Nov 12 2014	
src/test/java/com/nrift/finch/inf/domain/repository		Nov 12 2014	
src/test/java/com/nrift/finch/inf/domain/service		Nov 12 2014	

Clicked src/main/java/com/nrift/finch/inf/batch package:

Name	Coverage	Build time	Links
src/main/java/com/nrift/finch/inf/batch	78.7%	Nov 12 2014	
Name			
BatchExecutionError.java	100.0%	Nov 12 2014	
BatchExecutionResult.java	100.0%	Nov 12 2014	
BatchMetaDatBuilderProcessRunner.java	100.0%	Nov 12 2014	
BatchMetaData.java	80.5%	Nov 12 2014	
BatchOption.java	96.3%	Nov 12 2014	
BatchParams.java	94.9%	Nov 12 2014	
DefaultExecutionStrategy.java	50.0%	Nov 12 2014	
DropDownDataGenerator.java		Nov 12 2014	
ExecutionStrategy.java		Nov 12 2014	
ProcessShutdownListener.java	12.9%	Nov 12 2014	

Clicked BatchExecutionError.java to check the code:

```

finch root / finch Batch
src/main/java/com/nrift/finch/inf/batch/BatchExecutionError.java

Coverage Duplications Issues Source Raw
Lines: 100 Classes: 1 Comments (%): 27.1%
Lines of code: 35 Statements: 9 Comment lines: 13
Functions: 3 Complexity: 3
Accessors: 6 Complexity /function: 1.0
Public API: 3

1 /**
2 * +=====
3 * Copyright (C) 2013-2018
4 * Nomura Research Institute Financial Technologies India Pvt. Ltd.
5 * All Rights Reserved
6 *
7 *
8 * This document is the sole property of Nomura Research Institute Financial Technologies
9 * India Pvt. Ltd. No part of this document may be reproduced in any form or by any
10 * means - electronic, mechanical, photocopying, recording or otherwise - without the prior
11 * written permission of Nomura Research Institute Financial Technologies India Pvt. Ltd.
12 *
13 * Unless required by applicable law or agreed to in writing, software distributed under
14 * the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF
15 * ANY KIND, either express or implied.
16 *
17 * +=====
18 */
19
package com.nrift.finch.inf.batch;
20
import java.io.Serializable;
21
22 import com.nrift.finch.inf.console.batch.server.BatchMetaDataResourceProvider;
23 /**
24 * Used by the #link BatchMetaDataResourceProvider to return error code to the calling application,
25 * in case it is not able to start the batch or some error occurs.
26 * @author amipa
27 */
28
29 public class BatchExecutionError implements Serializable {
30
31     /** Default serial version UID */
32     private static final long serialVersionUID = 1L;
33
34     private String errorKey;
35     private String componentName;
36     private String errorMessage;
37
38     public BatchExecutionError() {}
39
40     /**
41      * @param errorKey
42      * @param componentName
43      */
44 }
45

```

Issue Drilldown

click Issue Drilldown under Tools section of Left Pane:

Screenshots illustrating the SonarQube interface for the 'finch root' project.

Issues Drilldown:

- Dashboard
- Hotspots
- Issues
- Time Machine
- TOOLS**
 - Components
 - Issues Drilldown**
 - Design
 - Libraries
 - Compare
- sonarqube

The Issues Drilldown section shows a summary of issues by severity (Blocker, Critical, Major, Minor, Info) and a detailed list of rules violated across various code files. A 'Time changes...' dropdown is also present.

Severity	Count	Rule	Count
Blocker	4	Throwable and Error classes should not be caught	4
Critical	38	Throwable.printStackTrace() should never be called	21
Major	2,488	System.exit() and Runtime.getRuntime().exit() should not be called	15
Minor	1,460	Switch cases should end with an unconditional break statement	1
Info	79	Strings should be compared using equals()	1
		else/for/while/do statements should always use curly braces	398

Code Structure:

```

finch root >
  finch Web Infrastructure
  finch Core
  finch DBD
  finch Console Infrastructure
  finch Notification
  finch Batch
  src/main/java/com/mnf/finch/inf/web/controller
  src/main/java/com/mnf/finch/dbd/domain/repository
  ...
  DefaultServiceController.java
  FinchQueryWizardController.java
  WidgetRepository.java
  AbstractQueryController.java
  AccessLogRepository.java
  ConsoleJobExecutor.java
  ...
  
```

Design

click Design under Tools section of Left Pane:

Screenshot of the SonarQube interface showing the 'Design' tool.

Left Panel:

- Dashboard
- Hotspots
- Issues
- Time Machine
- TOOLS**
 - Components
 - Issues Drilldown
 - Design**
 - Libraries
 - Compare
- sonarqube

Right Panel:

The 'Design' tool displays a dependency matrix between various modules. The legend indicates:

- Dependency (grey square)
- Suspect dependency (cycle) (red square)
- uses > (green square)
- uses > (blue square)
- (yellow square)

	finch Scheduler	finch Upload	finch DBD	finch Client API	finch Notification	finch Web Infrastructure	finch Batch	finch Console Infrastructure	finch Core
finch Scheduler	-	-	-	1	1	1	1	1	1
finch Upload	-	-	-	-	-	-	-	-	-
finch DBD	-	-	-	-	-	-	-	-	-
finch Client API	1	-	-	-	-	-	-	-	-
finch Notification	1	1	-	-	-	-	-	-	-
finch Web Infrastructure	1	1	1	-	-	-	-	-	-
finch Batch	1	-	-	-	-	-	1	-	-
finch Console Infrastructure	1	-	-	-	1	1	1	-	-
finch Core	1	1	1	1	1	1	1	1	-

Libraries

click Libraries under Tools section of Left Pane:

The screenshot shows the SonarQube interface for the 'finch root' project. The left sidebar includes links for Dashboard, Hotspots, Issues, Time Machine, Tools (Components, Issues Drilldown, Design), Libraries, Compare, and a SonarQube logo. The main content area displays a tree view of dependencies under 'finch root 2.1.12.1'. The tree starts with 'finch root 2.1.12.1' and branches into various libraries such as com.google.guava.guava 13.0.1 (compile), joda-time.joda-time 2.1 (compile), com.atlassian.fugue.fugue 1.1 (compile), org.slf4j.slf4j-over-slf4j 1.7.2 (compile), org.slf4j.slf4j-over-slf4j 1.7.2 (compile), org.slf4j.slf4j-api 1.7.2 (compile), ch.qos.logback.logback-core 1.0.9 (compile), ch.qos.logback.logback-classic 1.0.9 (compile), org.apache.commons.commonslang3 3.1 (compile), commons-beanutils.commonsbauutils 1.8.3 (compile), commons-collections.commonscollections 3.2.1 (compile), commons-digester.commonsdigester 2.1 (compile), commons-discovery.commonsdiscov 0.2 (compile), org.apache.camel.camel-core 2.12.1 (compile), org.apache.camel.camel-jms 2.12.1 (compile), com.fasterxml.jackson.core.jackson-databind 2.1.4 (compile), com.fasterxml.jackson.core.jackson-annotations 2.1.4 (compile), com.fasterxml.jackson.core.jackson-core 2.1.4 (compile), com.thoughtworks.xstream.xstream 1.3.1 (compile), com.sun.jersey.jersey-json 1.17.1 (compile), jdjson.b10 (compile), javax.jms.jms 1.1 (compile), oracle.jdbc16.11.2.0.1.0 (compile), org.apache.xmlbeans.xmlbeans-spring 3.7 (compile), org.hibernate.hibernate-core 4.1.12.Final (runtime), org.hibernate.hibernate-entitymanager 4.1.12.Final (runtime), org.hibernate.java.persistence.hibernate-jpa-2.0-api 1.0.1.Final (compile), org.springframework.spring-aspects 3.2.2.RELEASE (compile), org.springframework.spring-beans 3.2.2.RELEASE (compile), org.springframework.spring-jms 3.2.2.RELEASE (compile), org.springframework.spring-context 3.2.2.RELEASE (compile), org.springframework.spring-core 3.2.2.RELEASE (compile), org.springframework.spring-orm 3.2.2.RELEASE (compile), org.springframework.spring-exm 3.2.2.RELEASE (compile).

Compare

click Compare under Tools section of Left Pane to see the Metric of of different measurements:

Home	Add metric	Add project								
TOOLS										
Dependencies										
Compare										
										
	finch root 2.1.7.0 Aug 26 2014	finch root 2.1.8.0 Sep 05 2014	finch root 2.1.9.0 Sep 22 2014	finch root 2.1.10.0 Sep 30 2014	finch root 2.1.11.0 Oct 20 2014	finch root 2.1.12.1 Nov 12 2014				
Lines of code	50,936	51,200	51,588	52,617	52,629	55,348				
Complexity	7,972	8,001	8,074	8,307	8,296	8,665				
Comments (%)	18.5%	18.5%	18.4%	18.5%	18.5%	18.3%				
Duplicated lines (%)	8.4%	8.5%	8.5%	8.3%	8.3%	8.3%				
Issues	3,728	3,748	3,784	3,862	3,856	4,069				
Coverage	80.9%	80.8%	79.8%	78.6%	78.6%	73.2%				
Complexity /class	11.3	11.3	11.3	11.5	11.5	11.5				
Complexity /function	2.1	2.1	2.1	2.1	2.1	2.1				
Complexity /file	12.0	12.0	12.0	12.2	12.2	12.3				
File dependencies to cut	63	63	63	67	67	67				
Package cycles	51	51	52	61	61	61				
Package dependencies to cut	23	23	23	26	26	26				
Package tangle index	13.0%	12.9%	12.4%	13.1%	13.1%	12.4%				
Public documented API (%)	56.0%	56.1%	56.0%	56.5%	56.6%	56.3%				
Public undocumented API	1,383	1,385	1,395	1,413	1,409	1,476				
Comment lines	11,577	11,632	11,669	11,941	11,956	12,364				
Duplicated blocks	240	248	250	254	254	272				
Duplicated files	90	92	90	92	92	98				
Duplicated lines	8,156	8,270	8,269	8,304	8,269	8,680				
Blocker issues	4	4	4	4	4	4				
Critical issues	39	39	39	39	39	38				
Major issues	2,271	2,278	2,304	2,353	2,352	2,488				
Weighted issues	8,379	8,413	8,502	8,678	8,672	9,154				

SonarQube™ technology is powered by SonarSource SA
Version 4.3 - Community - Documentation - Get Support - Plugins

Also Metric can be added using Add Metric Drop down

3.6 Frequently Asked Questions (FAQ)

- 3.6.1 General Features
- 3.6.2 UI - General
- 3.6.3 Console
- 3.6.4 UI - Dashboard
- 3.6.5 Transaction & Persistence
- 3.6.6 Data Model

- 3.6.7 Unit Testing
- 3.6.8 WAS
- 3.6.9 JMS
- 3.6.10 Logging

3.6.1 General Features

Q 1: Is infrastructure supports access from smart device?

Ans: Details given in [about finch](#).

Q 2. How to configure access control from UI?

Ans: Details about Access Control available in this [page](#) and related pages.

3.6.2 UI - General

Q 1. What configuration need to done to show application date in header after sign-in in UI?

A 1: Please refer to [4.3.1 Application Date](#)

Q 2. In order to Create entry page what steps required?

Ans: Please refer to [Screen Design](#) and related page.

Q 3. In order to Create Query page with Result,Sub Action, reports, what steps required?

Ans: Please refer to [Screen Design](#) and related page.

Q 4. For Bulk Action in UI for set of different like items, what are the steps required?

Ans: One example with steps available in this [page](#).

Q 5. For Upload what are the steps required?

Ans: Please refer to [file Upload](#) page.

Q 6. For closing one Action (can be UI/Console) for a particular user what are steps required.

3.6.3 Console

Q1. In order to Execute Batch Process from UI , what are the step needs to perform?

Ans: Please refer to [Batch Process from UI and Console](#).

3.6.4 UI - Dashboard

Q1. In order add notification widget in Dashboard what are the steps needs to perform?

Ans: For Design please refer to [Widget Design](#) and example of [chart](#).

3.6.5 Transaction & Persistence

3.6.6 Data Model

3.6.7 Unit Testing

3.6.8 WAS

Q1. What are steps need to be perform to deploy and run the application in IBM WAS?

Ans: Problem faced with solution available during deployment in WAS is available in [this](#) page.

3.6.9 JMS

Q1. For JMS apart Active MQ installation and execution what are configuration required

Ans: Available in [this](#) page.

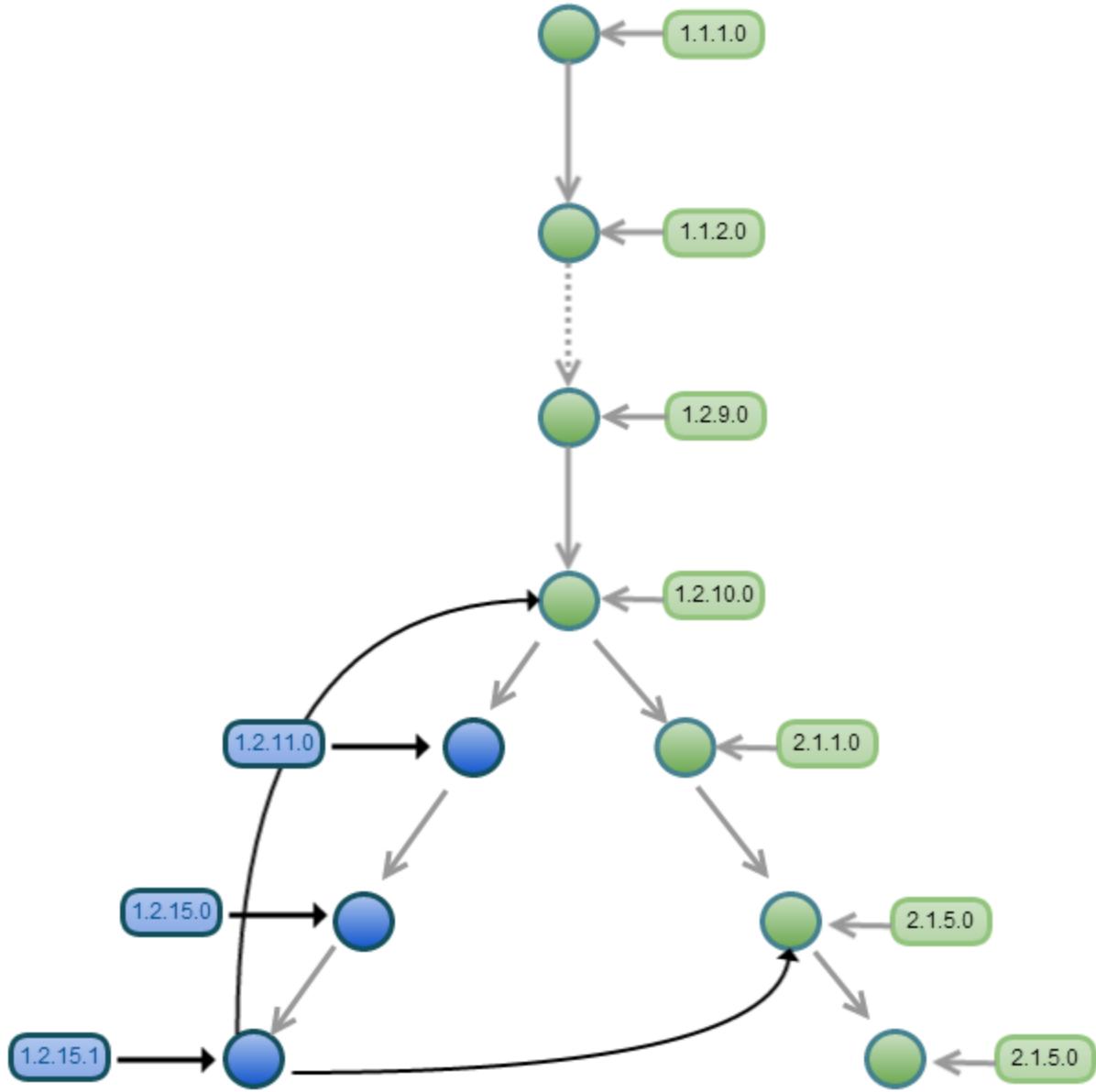
3.6.10 Logging

Q1. In order to create log message for action performed in application, what are the steps required?

Ans: Please refer to [this](#) page.

3.7 Version Upgrade

How to Upgrade



Please note that you cannot directly upgrade from one version to another. There are two possible upgrade scenarios.

- Linear upgrade (within same version, say finch1 to another finch1 or finch2 to another finch2 version)
- Non-linear upgrade (within different version, like finch1 to finch2 version)

Linear Upgrade

For example, to upgrade from version 1.2.15.0 to version 2.1.6.0, it is necessary to first upgrade to the next version of 1.2.5.0 (like 1.2.5.1), then to version 2.1.1.0, and so on; assuming 1.2.5.1 is the last version of finch1. There are mainly two things need to be considered: database changes and configuration changes. In every release note finch provide Change Notes section that is particularly important for application developer for every release. This section contains the changes (database, configuration, API, jar etc.) that need to be carried out by application developers for upgrading to that particular version from previous version.

Non-linear Upgrade

Database Change

In every release note finch provide database scripts for Oracle and MySQL as zip file. If you open a particular zip then you will find three folders COMMON, CONSOLE and WEB. COMMON is applicable for both console and web features and contains scripts that need to be executed for both. Inside COMMON and CONSOLE you will find two folders DELTA_PART and TAR_FILE; first one contains the delta script (database change from previous version) and second one contains the whole database script; this is the script for creating database stuffs (like tables, sequences etc.) for that version newly. So for upgrading from previous version you need the DELTA_PART. In WEB folder same structure is maintained but in two components: INF and DBD. Run scripts for INF followed by DBD.

Configuration Change

Go through the changes like New API/features, Modified API/feature/library and do the changes in your application as described.

Release Notes

1.4.3 Release Note