Apache OFBiz Developer Manual

The Apache OFBiz Project

Version Trunk

Table of Contents

1. Introduction

Welcome to the Apache OFBiz developer manual. This manual provides information to help with customizing and developing OFBiz. If you are new to OFBiz and interested in learning how to use it, you may want to start with the "Apache OFBiz User Manual".

OFBiz is a large system composed of multiple subsystems. This manual attempts to introduce the overall architecture and high level concepts, followed by a detailed description of each subsystem. In addition, the manual will cover topics necessary for developers including the development environment, APIs, deployment, security, and so on.

1.1. Main systems

OFBiz at its core is a collection of systems:

- A web server (Apache Tomcat)
- A web MVC framework for routing and handling requests.
- An entity engine to define, load and manipulate data.
- A service engine to define and control business logic.
- A widget system to draw and interact with a user interface.

On top of the above mentioned core systems, OFBiz provides:

- A data model shared across most businesses defining things like orders, invoices, general ledgers, customers and so on.
- A library of services that operate on the above mentioned data model such as "createBillingAccount" or "updateInvoice" and so on.
- A collection of applications that provide a user interface to allow users to interact with the system. These applications usually operate on the existing data model and service library. Examples include the "Accounting Manager" and "Order Manager".
- A collection of optional applications called "plugins" that extend basic functionality and is the main way to add custom logic to OFBiz.

1.2. Components

The basic unit in OFBiz is called "component". A component is at a minimum a folder with a file inside of it called "ofbiz-component.xml"

Every application in OFBiz is a component. For example, the order manager is a component, the accounting manager is also a component, and so on.

By convention, OFBiz components have the following main directory structure:

```
component-name-here/
 ├─── config/              - Properties and translation labels (i18n)
 ├─── data/                - XML data to load into the database
 ├─── entitydef/           - Defined database entities
 ├─── groovyScripts/       - A collection of scripts written in Groovy
 ├─── minilang/            - A collection of scripts written in minilang (deprecated)
 ├─── ofbiz-component.xml  - The OFBiz main component configuration file
 ├─── servicedef           - Defined services.
 ├─── src/
     ├─── docs/            - component documentation source
     └─── main/java/       - java source code
     └─── test/java/       - java unit-tests
 ├─── testdef              - Defined integration-tests
 ├─── webapp               - One or more Java webapps including the control servlet
 └─── widget               - Screens, forms, menus and other widgets
```

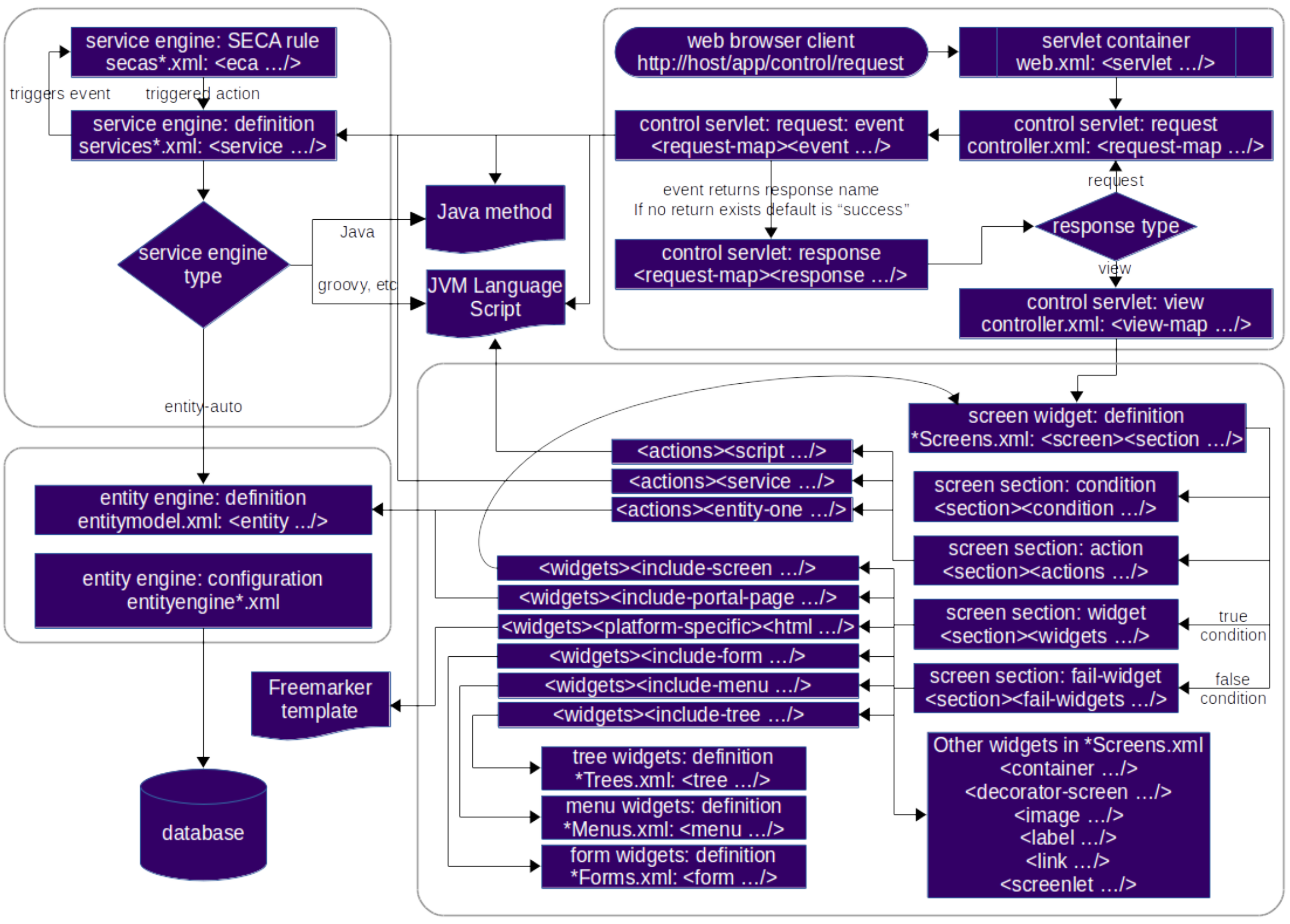
It is apparent from the above directory structure that each OFBiz component is in fact a full application as it contains entities, data, services, user interface, routing, tests, and business logic.

Both core OFBiz applications as well as plugins are nothing more than components. The only difference is that core applications reside in the "applications" folder whereas plugins reside in the "plugins" folder; also OFBiz does not ship with plugins by default.

1.3. Example workflow

Many basic concepts were explained so far. An example would help in putting all of these concepts together to understand the bigger picture. Let us take an example where a user opens a web browser and enters a certain URL and hits the enter key. What happens? It turns out answering this question is not quite simple because lots of things occur the moment the user hits "enter".

To try to explain what happens, take a look at the below diagram. Do not worry if it is not fully understandable, we will go through most of it in our example.



service engine: SECA rule secas*.xml: <eca .../>
triggers event
triggered action
service engine: definition services*.xml: <service .../>
service engine type
Java
groovy, etc
Java method
JVM Language Script
entity-auto
web browser client http://host/app/control/request
servlet container web.xml: <servlet .../>
control servlet: request: event <request-map><event .../>
control servlet: request controller.xml: <request-map .../>
event returns response name
If no return exists default is "success"
request
response type
control servlet: response <request-map><response .../>
view
control servlet: view controller.xml: <view-map .../>
screen widget: definition *Screens.xml: <screen><section .../>
<actions><script .../>
<actions><service .../>
<actions><entity-one .../>
entity engine: definition entitymodel.xml: <entity .../>
entity engine: configuration entityengine*.xml
screen section: condition <section><condition .../>
screen section: action <section><actions .../>
screen section: widget <section><widgets .../>
true condition
screen section: fail-widget <section><fail-widgets .../>
false condition
<widgets><include-screen .../>
<widgets><include-portal-page .../>
<widgets><platform-specific><html .../>
<widgets><include-form .../>
<widgets><include-menu .../>
<widgets><include-tree .../>
Freemarker template
database
tree widgets: definition *Trees.xml: <tree .../>
menu widgets: definition *Menus.xml: <menu .../>
form widgets: definition *Forms.xml: <form .../>
Other widgets in *Screens.xml <container .../> <decorator-screen .../> <image .../> <label .../> <link .../> <screenlet .../>

1.3.1. User enters URL

In the first step in our example, the user enters the following URL:

https://localhost:8443/accounting/control/findInvoices

If we break down this URL, we identify the following parts:

- localhost: Name of the server in which OFBiz is running
- 8443: Default https port for OFBiz
- accounting: web application name. A web application is something which is defined *inside* a component
- control: Tells OFBiz to transfer routing to the control servlet
- findInvoices: request name inside the control servlet

1.3.2. Control servlet takes over

The Java Servlet Container (tomcat) re-routes incoming requests through web.xml to a special OFBiz servlet called the control servlet. The control servlet for each OFBiz component is defined in controller.xml under the webapp folder.

The main configuration for routing happens in controller.xml. The purpose of this file is to map requests to responses.

Request Map

A request in the control servlet might contain the following information:

- Define communication protocol (http or https) as well as whether authentication is required.
- Fire up an event which could be either a piece of code (like a script) or a service.
- Define a response to the request. A response could either be another request or a view map.

So in this example, the findInvoices request is mapped to a findInvoices view.

View Map

A view map maps a view name to a certain view-type and a certain location.

View types can be one of:

- screen: A screen widget which translates to normal HTML.
- screenfop: A PDF screen designed with Apache FOP based constructs.
- screencsv: A comma separated value output report.
- screenxml: An XML document.
- simple-content; A special MIME content type (like binary files).
- ftl: An HTML document generated directly from a FreeMarker template.
- screenxls: An Excel spreadsheet.

In the findInvoices example, the view-map type is a normal screen which is mapped to the screen: component://accounting/widget/InvoiceScreens.xml#FindInvoices

1.3.3. Widget rendered

Once the screen location is identified and retrieved from the previous step, the OFBiz widget system starts to translate the XML definition of the screen to actual HTML output.

A screen is a collection of many different things and can include:

- Other screens
- Decorator screens
- Conditional logic for hiding / showing parts of the screen
- data preparation directives in the <action> tag
- Forms
- Menus
- Trees
- Platform specific code (like FreeMarker for HTML output)
- Others (portals, images labels etc …)

Continuing the example, the FindInvoices screen contains many details including two forms. One form is for entering invoice search fields and the other form displays search results.

1.4. The Apache OFBiz documentation system

1.4.1. Overview

there are two supports for OFbiz documentation, the wiki and some mains documents (in pdf and html format)

- user-manual
- developer-manual
- documentation_guidelines
- README

Asciidoc

The OFBiz documents are generated from a number of Asciidoc files. In general the files are stored on each component in the 'src/docs/asciidoc' directories.
The general main documents include all files from component.

The manuals and guidelines documents are located in docs/asciidoc directories, and REAME.adoc is in root directory.

Help link in the OFBiz user interface, are link to the user-manual generated by buildbot process from Apache OFBiz community. It’s possible to change a property in OFBiz to have link to your own generation.

For details about asciidoc rules used in Apache OFBiz have a look to Documentation Guidelines

Application components

All main files of each component are included in user-manual.adoc

Framework components

All main files of each component are included in developer-manual.adoc except for webtools which is included in user-manual

Plugins

For the main files of the plugin components, there are two ways to read them.

On the one hand, the plugin documentation generation process generates one document per plugin, so that you can see the list of documents in the pluginsdoc directory and thus read each of them;

On the other hand, each plugin master file is included in the plugin chapter of the user manual or developer manual, depending on whether the plugin is "technical" or "functional".

Wiki

Wiki is the second way to contribute to the documentation. Detail on how to Help for providing help content is on the wiki ☺

Most of wiki page has been or will be migrated to the asciidoc pages, but, as wiki is more easier to be update (update existing page or create new one) the two system will continue to exist and live.

1.4.2. Contributing a new help screen, a few steps.

Documentation Guidelines is the first doc to read to be able to contribute to documentation and/or help.

If you are looking for asciidoc files format examples, please look at the following files:

1. An example for a chapter of a component at: applications/humanres/src/docs/asccidoc/_include/hr-intro.adoc
2. An example of a help screen: applications/humanres/src/docs/asccidoc/_include/HELP-hr-main.adoc

Screens

If you would like to create a new help for a certain screen, you need to do the following:

1. Write documentation in a functional point of view and in a process perspective.
2. Each title (in all level) generate in html an anchor, so starting point of the help should be a title.
3. Take the anchor generated (something like _the_title , with only lowercase), for example by looking in the html file generated.
4. In the screen add a `<set field` for `helpAnchor` with anchor generated as value.

1.4.3. Internationalization of the help system

Currently documentation is only in English (except for 3 or 4 files, not included).

In near future, there will be a solution to be able to have documentation/help in multiple languages, a jira (OFBIZ-12030) is open of that.

The switching between locale will be completely automatic (depending on OFBiz user local)

2. Web Framework

3. Web Applications

The OFBiz webapp is one of the core framework components. It is tightly integrated with other framework components.

3.1. Cross-domains Single Sign On (SSO)

In some cases you need to split the OFBiz applications on different servers, and possibly in production on different domains. This can happen for different reasons, most often for performance reason.

As it’s annoying to give each time a credential when changing from an OFBiz application to another on the same server, the same applies when changing from an OFBiz application to another on another domain.

To prevent that on the same server, the ExternalLoginKey mechanism is used. The cross-domains SSO feature allows to navigate from a domain to another with automated SSO.

It based on 3 technologies:

JWT
: JWT Official site - Wikipedia for JWT

CORS
: CORS (Mozilla doc) - Wikipedia for CORS

Ajax
: Ajax, now well known I guess, in OFBiz we use jQuery for that.

The mechanism is simple.

On the source side:

1. When an user log in in an application (webApp) a webappName.securedLoginId cookie is created. This cookie will be used by the mechanism to know the current logged in user. *Note that all webappName.securedLoginId cookies are deleted when the user session is closed or time out. Hence (apart also using an intrinsically secured cookie) the mechanim is secured, even on shared machines. Of course if people are sharing a machine during their sessions, things could get complicated. This unlikely later case is not taken in account.*
2. The user is given a JavaScript link which passes the URL to reach and the calling webapp name to the sendJWT() Ajax function.
3. The sendJWT() Ajax function calls the loadJWT() Ajax function which in turn calls the CommonEvents::loadJWT method through the common controller.
4. The CommonEvents::loadJWT method uses the calling webapp name to retrieve the userLoginId from the secured webappName.securedLoginId cookie, creates a JWT containing the userLoginId, and returns it to the loadJWT() Ajax function.
5. Then the sendJWT() Ajax function sends an Authorization header containing the JWT to the URL

to reach. At this stage, if all things are correct, the flow leaves the source side.

On the server side:

1. A CORS policy is needed. *Without it, the Authorization token containing the JWT will be rejected. It’s a simple policy but you need to strictly define the authorized domains. Never use the lazy "*" for domains (ie all domains), else the preflight request will not work.* Here is an example for Apache HTTPD (domain value is "https://localhost:8443" for official OFBiz demo):

```
Header set Access-Control-Allow-Origin domain
Header set Access-Control-Allow-Headers "Authorization"
Header set Access-Control-Allow-Credentials "true"
```

1. The checkJWTLogin preprocessor, similar to the checkExternalLoginKey, intercepts the JWT, checks it and if all is OK signs the user on. That’s it !

In the example component, the FormWidgetExamples screen contains 2 new fields in the LinksExampleForm which demonstrate the use from a local instance to the trunk demo instance.

If you are interested in more details you may refer to https://issues.apache.org/jira/browse/OFBIZ-10307

3.2. Control Servlet

3.2.1. Requests

3.2.2. Views

4. The Entity Engine.

4.1. Entity Engine guide.

Entity engine Guide

4.2. Entity Engine Configuration Guide.

entity engine configuration guide

4.3. Entity overview.

Entity overview

4.4. Entities

4.4.1. Standard Entities

4.4.2. View Entities

4.4.3. Extended Entities

4.4.4. Dynamic View Entities

4.5. XML Data

4.6. The OFBiz Datafiles

4.6.1. Intoduction

There is a data import tool in OFBiz called the DataFile tool.
It uses XML files that describe flat file formats (including character delimited, fixed width, etc) and parses the flat files based on those definitions. So, by design it is somewhat like the Entity Engine.

It uses a generic object to represent a row in the flat file.
It includes features like a line type code for each line and can support hierarchical flat files (ie where parent/child relationships are implied by sub-records).

The code is in the ofbiz/framework/datafile directory, and there is an XSD there to describe how the data file definition XML file should look.

The DataFile has a web page in WebTools to parse data files and generate entity xml files (that can be imported in OFBiz) from them, and to do in/outtesting.

both the import of fixed width flat files and character delimited files are implemented.

4.6.2. Definitions

- File : this is an xml file that contains one or more "Data File Definition"s;
  - the grammar of the file is defined in framework/datafile/dtd/datafile.xsd
- Data file Definition : the definition of a data file structure (e.g. names/types of the fields/columns in a "Data File";
  - the same "Data File" could have more than one "Data File Definition"s (e.g. the definitions could consider different subsets of all the fields available in the data file).
- Data file : the text file that contains all the data (a fixed width or a character delimited text file)

4.6.3. How to use the "Work With Data Files" screen in the Webtools application

Prerequisites: a definition file (containing the fields' definition of the data file) and a data file (containing the data you want to parse/import) should be available in the OFBiz server.Steps:

1. connect to the Webtools application
2. go to the "Work With Data Files" screen
3. enter the path to the your definition file in the "Definition Filename or URL" input field
4. click on the submit button>
5. the "Data File Definition Name" input field will be changed into a drop down box containing all the definitions available in the definition file
6. select the definition you want to use from the drop down box>
7. enter the path to the your data file in the "Data Filename or URL" input field
8. if you want to generate an entity xml file that can be later imported in OFBiz, enter a path for it in the "Save to entity xml file:
9. input field and click on the submit button; the file will be created
   (see the paragraph "Tips to create Entity Xml Files" for more details)

```
<field name="productId" type="String">
            </field>
```

4.6.4. Examples

Sample fixed width CSV file posreport.csv to be imported:

An example of fixed width flat file import.

```
021196033702    ,5031BB GLITTER GLUE PENS BRIGH  ,1            ,5031BB      ,       1,
299,
021196043121    ,BB4312 WONDERFOAM ASSORTED      ,1            ,BB4312      ,       1,
280,
021196055025    ,9905BB  PLUMAGE MULTICOLOURED   ,1            ,9905BB      ,       4,
396,
```

Sample xml definition file for importing select columns

Sample xml definition file for importing select columns posschema.xml:

```
    <data-files
xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/datafiles.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <data-file name="posreport" separator-style="fixed-length" type-code="text">
            <record name="tillentry" limit="many">
                <field name="tillCode" type="String" length="16" position="0"></field>
                <field name="name" type="String" length="32" position="17"></field>
                <field name="prodCode" type="String" length="12" position=
"63"></field>
                <field name="quantity" type="String" length="8" position="76"></field>
                <field name="totalPrice" type="String" length="8" position=
"85"></field>
            </record>
        </data-file>
    </data-files>
```

Another example reading fixed record little endian binary files

```
    <data-files
xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/datafiles.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <data-file name="stockdata" separator-style="fixed-record" type-code="text"
record-length="768">
            <record name="stockdataitem" limit="many">
                <field name="barcode" type="NullTerminatedString" length="12"
position="0"></field>
                <field name="prodCode" type="NullTerminatedString" length="12"
position="68"></field>
                <field name="price" type="LEInteger" length="4" position="80"></field>
                <field name="name" type="NullTerminatedString" length="30"
position="16"></field>
            </record>
        </data-file>
    </data-files>
```

Procedure:

In the interface enter something like:

1. Definition Filename or URL: posschema.xml
2. Data File Definition Name: posreport
3. Data Filename or URL: posreport.csv

4.7. Entity engine configuration

4.8. Supported databases

4.9. Data Model Changes

The Apache OFBiz® Project Release trunk

Apache OFBiz follows **The Universal Data Model** by **Len Silverston**, with a grain of salt.

The following file contains information about the data model changes in the Apache OFBiz. The detailed description of migration scripts specified here can be found at Revisions Requiring Data Migration - upgrade ofbiz page.

4.9.1. Changes with OFBiz Trunk (Upcoming Branch)

Entity Changes

Added new entities

1. ProdPromoCodeContactMech
2. OrderDenylist
3. OrderDenylistType
4. AgreementStatus

Removed/Deprecate 3 entities

1. ProductPromoCodeEmail
2. OrderBlacklist
3. OrderBlacklistType

Field Changes

Entity	Field	Action	IsPK	Revision
GlXbrlClass	parentGlXbrlClassId	Added	No	36a123a
Agreement	statusId	Added	No	da8817d

If in the list above fields are mentioned where 'IsPK' = yes, we advice to follow steps below for a successful upgrade:

1. Stop your OFBiz implementation
2. Go to the entity definition in the appropriate entitymodel.xml file
3. Disable temporarily the prim-key reference of the added field, and save the file
4. Restart the OFBiz implementation
5. Check with your RDBMS client that the new field is present in the table
6. Add values for the field in the table (if appropriate)
7. Stop the OFBiz implementation
8. Go back to the definition in the appropriate entitymodel.xml file
9. Enable the temporary disabled prim-key reference of the added field, and save the file
10. Check with your RDBMS client that the new field is part of the primary key defintion of the table

Migration Scripts

1. Migration service migrateProductPromoCodeEmail is implemented to migrate the ProductPromoCodeEmail entity to ProductPromoCodeContactMech.
   (More detail at OFBIZ-5426)

4.9.2. Changes with OFBiz 17

Field types `id-ne`, `id-long-ne` & `id-vlong-ne` has been removed. Use `id`, `id-long` and `id-vlong` instead (detailed description at OFBIZ-9351).

Entity Changes

No changes

Field Changes

Entity	Field	Action	IsPK	Revision
MarketingCampaignPrice	fromDate	Added	Yes	R1805961
MarketingCampaignPrice	thruDate	Added	No	R1805961
MarketingCampaignPromo	fromDate	Added	Yes	R1805961
MarketingCampaignPromo	thruDate	Added	No	R1805961
MarketingCampaignRole	fromDate	Added	Yes	R1805961

Entity	Field	Action	IsPK	Revision
MarketingCampaignRole	thruDate	Added	No	R1805961
Product	manufacturerPartyId	Removed	No	R1804408
SecurityGroupPermission	fromDate	Added	Yes	R1812383
SecurityGroupPermission	thruDate	Added	No	R1812383

Migration Scripts

1. Updated sql-type for date-time and time field in fieldtypemysql.xml file at commit R1793300
   Update msyql sql-type for datetime field-type to support Fractional Seconds in Time Values Please upgrade mysql to at least 5.6.4 or higher.
   After upgrade run `generateMySqlFileWithAlterTableForTimestamps` service, groupName is required field for this service.
   It will generate sql file with alter query statement for date-time and time field at location `$\{ofbiz.home}/runtime/tempfiles/.sql`
   You can use execute sql statement from any of the mysql batch command.

4.9.3. Changes between OFBiz 9 to OFBiz 16

Entity Changes

Added 77 new entities

1. JobRequisition
2. ProductAverageCostType
3. WorkEffortSurveyAppl
4. WorkEffortIcalData
5. WebSiteContactList
6. WebAnalyticsType
7. WebAnalyticsConfig
8. UserLoginSecurityQuestion
9. UomGroup
10. TrainingRequest
11. ThirdPartyLogin
12. TestFieldType
13. TestingSubtype
14. TestingStatus

15. TestingRemoveAll
16. TestingItem
17. TestingCrypto
18. SystemProperty
19. ShipmentGatewayUsps
20. ShipmentGatewayUps
21. ShipmentGatewayFedex
22. ShipmentGatewayDhl
23. ShipmentGatewayConfig
24. ShipmentGatewayConfigType
25. ReturnContactMech
26. QuoteNote
27. ProductPromoContent
28. ProductPromoContentType
29. ProductGroupOrder
30. ProductCostComponentCalc
31. CostComponentCalc
32. PayPalPaymentMethod
33. PaymentGroupType
34. PaymentGroup
35. PaymentGroupMember
36. PaymentGatewayConfig
37. PaymentGatewayConfigType
38. PaymentGatewayWorldPay
39. PaymentGatewaySecurePay
40. PaymentGatewaySagePay
41. PaymentGatewayOrbital
42. PaymentGatewayEway
43. PaymentGatewayCyberSource
44. PaymentGatewayAuthorizeNet
45. PaymentGatewayIDEAL
46. PaymentContentType
47. PaymentContent
48. OAuth2LinkedIn
49. OAuth2GitHub

50. JobManagerLock
51. JobInterviewType
52. JobInterview
53. JavaResource
54. InvoiceNote
55. InvoiceItemAssocType
56. InvoiceItemAssoc
57. InvoiceContentType
58. InvoiceContent
59. GlAccountCategoryType
60. GlAccountCategoryMember
61. GlAccountCategory
62. GitHubUser
63. FixedAssetTypeGlAccount
64. FacilityContent
65. ExcelImportHistory
66. EmplLeaveReasonType
67. EbayShippingMethod
68. EbayConfig
69. CountryAddressFormat
70. ContentSearchResult
71. ContentSearchConstraint
72. ContentKeyword
73. CheckAccount
74. AgreementFacilityAppl
75. AgreementContentType
76. AgreementContent

Removed 8 entities

1. DepreciationMethod
2. FixedAssetMaintMeter
3. OagisMessageErrorInfo
4. OagisMessageInfo
5. SalesOpportunityTrackingCode
6. SimpleSalesTaxLookup

7. TestBlob
8. WorkEffortAssignmentRate

Field Changes

Entity	Field	Action	IsPK	Revision
AcctgTransAttribute	attrDescription	Added	No	NA
AcctgTransEntry	inventoryItemId	Added	No	NA
AcctgTransTypeAttr	description	Added	No	NA
BenefitType	parentTypeId	Added	No	NA
BenefitType	hasTable	Added	No	NA
BudgetAttribute	attrDescription	Added	No	NA
BudgetItemAttribute	attrDescription	Added	No	NA
BudgetItemTypeAttr	description	Added	No	NA
BudgetStatus	changeByUserLoginId	Added	No	NA
BudgetTypeAttr	description	Added	No	NA
CommunicationEventRole	statusId	Added	No	NA
CommunicationEventType	contactMechTypeId	Added	No	NA
ContactListCommStatus	partyId	Added	No	NA
ContactListCommStatus	messageId	Added	No	NA
ContactListCommStatus	changeByUserLoginId	Added	No	NA
ContactMechAttribute	attrDescription	Added	No	NA
ContactMechTypeAttr	description	Added	No	NA
DeductionType	parentTypeId	Added	No	NA
DeductionType	hasTable	Added	No	NA
DocumentAttribute	attrDescription	Added	No	NA

Entity	Field	Action	IsPK	Revision
DocumentTypeAttr	description	Added	No	NA
EmploymentApp	approverPartyId	Added	No	NA
EmploymentApp	jobRequisitionId	Added	No	NA
EmploymentAppSourceType	parentTypeId	Added	No	NA
EmploymentAppSourceType	hasTable	Added	No	NA
EmplPositionClassType	parentTypeId	Added	No	NA
EmplPositionClassType	hasTable	Added	No	NA
EmplPositionType	parentTypeId	Added	No	NA
EmplPositionType	hasTable	Added	No	NA
EmplPositionType	partyId	Removed	No	NA
EmplPositionType	roleTypeId	Removed	No	NA
FinAccountAttribute	attrDescription	Added	No	NA
FinAccountTransAttribute	attrDescription	Added	No	NA
FinAccountTrans	glReconciliationId	Added	No	NA
FinAccountTrans	statusId	Added	No	NA
FinAccountTransTypeAttr	description	Added	No	NA
FinAccountTypeAttr	description	Added	No	NA
FinAccountStatus	changeByUserLoginId	Added	No	NA
FixedAsset	acquireOrderId	Added	No	NA
FixedAsset	acquireOrderItemSeqId	Added	No	NA
FixedAssetAttribute	attrDescription	Added	No	NA
FixedAssetTypeAttr	description	Added	No	NA
GlAccount	externalId	Added	No	NA
GlAccount	openingBalance	Added	No	NA

Entity	Field	Action	IsPK	Revision
GlReconciliation	createdDate	Added	No	NA
GlReconciliation	lastModifiedDate	Added	No	NA
GlReconciliation	statusId	Added	No	NA
GlReconciliation	openingBalance	Added	No	NA
InventoryItemAttribute	attrDescription	Added	No	NA
InventoryItemStatus	changeByUserLoginId	Added	No	NA
InventoryItemTypeAttr	description	Added	No	NA
InvoiceAttribute	attrDescription	Added	No	NA
InvoiceItemAttribute	attrDescription	Added	No	NA
InvoiceItemTypeAttr	description	Added	No	NA
InvoiceStatus	changeByUserLoginId	Added	No	NA
InvoiceTypeAttr	description	Added	No	NA
InvoiceTermAttribute	attrDescription	Added	No	NA
JobSandbox	currentRetryCount	Added	No	NA
JobSandbox	tempExprId	Added	No	NA
JobSandbox	currentRecurrenceCount	Added	No	NA
JobSandbox	maxRecurrenceCount	Added	No	NA
JobSandbox	jobResult	Added	No	NA
OrderAdjustment	amountAlreadyIncluded	Added	No	NA
OrderAdjustment	isManual	Added	No	NA
OrderAdjustment	oldPercentage	Added	No	NA
OrderAdjustment	oldAmountPerQuantity	Added	No	NA
OrderAdjustment	lastModifiedDate	Added	No	NA
OrderAdjustment	lastModifiedByUserLogin	Added	No	NA

Entity	Field	Action	IsPK	Revision
OrderAdjustmentAttribute	attrDescription	Added	No	NA
OrderAdjustmentTypeAttr	description	Added	No	NA
OrderAttribute	attrDescription	Added	No	NA
OrderItem	supplierProductId	Added	No	NA
OrderItem	cancelBackOrderDate	Added	No	NA
OrderItem	changeByUserLoginId	Added	No	NA
OrderItemAttribute	attrDescription	Added	No	NA
OrderItemShipGroup	facilityId	Added	No	NA
OrderItemShipGroup	estimatedShipDate	Added	No	NA
OrderItemShipGroup	estimatedDeliveryDate	Added	No	NA
OrderItemShipGrpInvRes	priority	Added	No	NA
OrderItemShipGrpInvRes	oldPickStartDate	Added	No	NA
OrderItemTypeAttr	description	Added	No	NA
OrderTermAttribute	attrDescription	Added	No	NA
OrderPaymentPreference	track2	Added	No	NA
OrderPaymentPreference	swipedFlag	Added	No	NA
OrderPaymentPreference	lastModifiedDate	Added	No	NA
OrderPaymentPreference	lastModifiedByUserLogin	Added	No	NA
OrderShipment	shipGroupSeqId	Added	No	NA
OrderTypeAttr	description	Added	No	NA
PartyAcctgPreference	orderSequenceEnumId	Removed	No	NA

Entity	Field	Action	IsPK	Revision
PartyAcctgPreference	quoteSequenceEnumId	Removed	No	NA
PartyAcctgPreference	invoiceSequenceEnumId	Removed	No	NA
PartyAcctgPreference	oldOrderSequenceEnumId	Added	No	NA
PartyAcctgPreference	oldQuoteSequenceEnumId	Added	No	NA
PartyAcctgPreference	oldInvoiceSequenceEnumId	Added	No	NA
PartyAcctgPreference	orderSeqCustMethId	Added	No	NA
PartyQual	infoString	Removed	No	NA
PartyQual	institutionInternalId	Removed	No	NA
PartyQual	institutionPartyId	Removed	No	NA
PartyQual	partyQualId	Removed	No	NA
PartyRate	percentageUsed	Added	No	NA
PartyRate	rate	Removed	No	NA
PartyResume	contentId	Added	No	NA
PaymentAttribute	attrDescription	Added	No	NA
PaymentGatewayResponse	gatewayCvResult	Added	No	NA
PaymentMethod	finAccountId	Added	No	NA
PaymentTypeAttr	description	Added	No	NA
PerfRatingType	parentTypeId	Added	No	NA
PerfRatingType	hasTable	Added	No	NA
PerfReview	payHistoryRoleTypeIdTo	Removed	No	NA
PerfReview	payHistoryRoleTypeIdFrom	Removed	No	NA
PerfReview	payHistoryPartyIdTo	Removed	No	NA
PerfReview	payHistoryPartyIdFrom	Removed	No	NA
PerfReview	payHistoryFromDate	Removed	No	NA

Entity	Field	Action	IsPK	Revision
PerfReviewItemType	parentTypeId	Added	No	NA
PerfReviewItemType	hasTable	Added	No	NA
PersonTraining	trainingRequestId	Added	No	NA
PersonTraining	workEffortId	Added	No	NA
PersonTraining	approverId	Added	No	NA
PersonTraining	approvalStatus	Added	No	NA
PersonTraining	reason	Added	No	NA
PostalAddress	houseNumber	Added	No	NA
PostalAddress	houseNumberExt	Added	No	NA
PostalAddress	cityGeoId	Added	No	NA
PostalAddress	municipalityGeoId	Added	No	NA
PostalAddress	geoPointId	Added	No	NA
PosTerminal	terminalName	Added	No	NA
PosTerminalInternTx	reasonEnumId	Added	No	NA
Product	releaseDate	Added	No	NA
Product	originalImageUrl	Added	No	NA
Product	inventoryItemTypeId	Added	No	NA
Product	shippingWeight	Added	No	NA
Product	productWeight	Added	No	NA
Product	diameterUomId	Added	No	NA
Product	productDiameter	Added	No	NA
Product	virtualVariantMethodEnum	Added	No	NA
Product	defaultShipmentBoxTypeId	Added	No	NA
Product	lotIdFilledIn	Added	No	NA
Product	orderDecimalQuantity	Added	No	NA
Product	weight	Removed	No	NA
Product	taxCategory	Removed	No	NA
Product	taxVatCode	Removed	No	NA

Entity	Field	Action	IsPK	Revision
Product	taxDutyCode	Removed	No	NA
ProductAttribute	attrDescription	Added	No	NA
ProductAverageCost	productAverageCostTypeId	Added	No	NA
ProductAverageCost	facilityId	Added	No	NA
ProductContent	sequenceNum	Added	No	NA
ProductKeyword	keywordTypeId	Added	No	NA
ProductKeyword	statusId	Added	No	NA
ProductRole	sequenceNum	Added	No	NA
ProductStore	balanceResOnOrderCreation	Added	No	NA
ProductStore	defaultTimeZoneString	Added	No	NA
ProductStore	oldStyleSheet	Added	No	NA
ProductStore	oldHeaderLogo	Added	No	NA
ProductStore	oldHeaderRightBackground	Added	No	NA
ProductStore	oldHeaderMiddleBackground	Added	No	NA
ProductStore	styleSheet	Removed	No	NA
ProductStore	headerLogo	Removed	No	NA
ProductStore	headerRightBackground	Removed	No	NA
ProductStore	headerMiddleBackground	Removed	No	NA
ProductStorePaymentSetting	paymentCustomMethodId	Added	No	NA
ProductStorePaymentSetting	paymentGatewayConfigId	Added	No	NA
ProductStoreShipmentMeth	shipmentCustomMethodId	Added	No	NA
ProductStoreShipmentMeth	shipmentGatewayConfigId	Added	No	NA
ProductStoreShipmentMeth	allowancePercent	Added	No	NA

Entity	Field	Action	IsPK	Revision
ProductStoreShipmentMeth	minimumPrice	Added	No	NA
ProductTypeAttribute	attrDescription	Added	No	NA
QuoteAdjustment	lastModifiedDate	Added	No	NA
QuoteAdjustment	lastModifiedByUserLogin	Added	No	NA
QuoteAttribute	attrDescription	Added	No	NA
QuoteItem	leadTimeDays	Added	No	NA
QuoteRole	fromDate	Added	Yes	NA
QuoteRole	thruDate	Added	No	NA
QuoteTerm	termDays	Added	No	NA
QuoteTerm	textValue	Added	No	NA
QuoteTerm	description	Added	No	NA
QuoteTermAttribute	attrDescription	Added	No	NA
QuoteTypeAttr	description	Added	No	NA
RequirementAttribute	changeByUserLoginId	Added	No	NA
RequirementStatus	changeByUserLoginId	Added	No	NA
ResponsibilityType	parentTypeId	Added	No	NA
ResponsibilityType	hasTable	Added	No	NA
ReturnAdjustment	createdByUserLoginId	Added	No	NA
ReturnAdjustment	lastModifiedDate	Added	No	NA
ReturnAdjustment	lastModifiedByUserLogin	Added	No	NA
ReturnHeader	supplierRmaId	Added	No	NA
ReturnItemResponse	finAccountTransId	Added	No	NA
ReturnStatus	changeByUserLoginId	Added	No	NA
SalaryStep	fromDate	Added	Yes	NA
SalaryStep	thruDate	Added	No	NA

Entity	Field	Action	IsPK	Revision
SalaryStep	createdByUserLoginId	Added	No	NA
SalaryStep	lastModifiedByUserLogin	Added	No	NA
SalesOpportunity	nextStepDate	Added	No	NA
ServiceSemaphore	lockedByInstanceId	Added	No	NA
ShoppingListItem	modifiedPrice	Added	No	NA
SkillType	parentTypeId	Added	No	NA
SkillType	hasTable	Added	No	NA
SupplierProduct	shippingPrice	Added	No	NA
SupplierProduct	supplierCommissionPerc	Removed	No	NA
TaxAuthorityRateProduct	isTaxInShippingPrice	Added	No	NA
TerminationType	parentTypeId	Added	No	NA
TerminationType	hasTable	Added	No	NA
TestingNodeMember	extendFromDate	Added	No	NA
TestingNodeMember	extendThruDate	Added	No	NA
TimeEntry	planHour	Added	No	NA
Timesheet	approvedByUserLoginId	Added	No	NA
TrainingClassType	parentTypeId	Added	No	NA
TrainingClassType	hasTable	Added	No	NA
UnemploymentClaim	thruDate	Added	No	NA
UserLogin	externalAuthId	Added	No	NA
UserLogin	userLdapDn	Added	No	NA
UserLogin	disabledBy	Added	No	NA
ValueLinkKey	createdByUserLogin	Added	No	NA
WebSite	visualThemeSetId	Added	No	NA
WebSite	hostedPathAlias	Added	No	NA
WebSite	isDefault	Added	No	NA

Entity	Field	Action	IsPK	Revision
WebSite	displayMaintenancePage	Added	No	NA
WebSitePathAlias	fromDate	Added	Yes	R1738588
WebSitePathAlias	thruDate	Added	No	R1738588
WorkEffort	tempExprId	Added	No	NA
WorkEffort	sequenceNum	Added	No	NA
WorkEffortAttribute	attrDescription	Added	No	NA
WorkEffortAssocAttribute	attrDescription	Added	No	NA
WorkEffortAssocTypeAttr	description	Added	No	NA
WorkEffortContactMech	fromDate	Added	Yes	NA
WorkEffortContactMech	thruDate	Added	No	NA
WorkEffortFixedAssetAssign	availabilityStatusId	Added	No	NA
WorkEffortPartyAssignment	assignedByUserLoginId	Added	No	NA
WorkEffortPurposeType	parentTypeId	Added	No	NA
WorkEffortStatus	reason	Added	No	NA
WorkEffortTypeAttr	description	Added	No	NA
WorkOrderItemFulfillment	shipGroupSeqId	Added	No	NA

5. Service Engine

A description of the service engine can be found at the OFBiz wiki Service Engine Guide

5.1. Declaration and Implementation

5.2. Supported languages

5.3. Transaction management

5.4. Web services

6. Widget System

6.1. Screen Widget

6.1.1. Decoration Pattern

6.2. Form Widget

6.3. Menu Widget

6.4. Tree Widget

6.5. Portal Widget

6.6. Platform Specific Code

6.7. Apache OFBiz Themes

The Apache OFBiz Project Release trunk

6.7.1. What is a theme

A Theme is an ofbiz component that defines all elements necessary to render all information generated by the screen engine through an embedded technology. Currently themes presents in Apache OFBiz use html5/jquery/css to do that.

To realize that, a theme can define some properties, among them some can be necessary. It can define its own ftl macro to render the different modelScreen elements and can define its own screen decorator to prepare the final user screen structure for the technology used by the theme.

6.7.2. How to define it

A theme is a standard component, present in the directory themes or plugins with a file definition present on widget/Theme.xml and support the widget-theme.xsd

To offer the possibility for end users to select the theme through the "Select Theme Screen", the theme need to load the entity VisualTheme and one or more entries related to this theme.

6.7.3. Structure of Theme.xml

The theme definition file help OFBiz to know what specific rendering the Theme want to use.
It's composed by two definition elements and four optional blocks

Main definition

The First mandatory element is one or several visualThemeId related to the theme
Defined like that

```
    <visual-themes>
        <visual-theme id="MY_THEME"/>
    </visual-themes>
```

You need to define in the database an entry in VisualTheme Entity for each visual-theme id defined.

```
<entity-engine-xml>
    <VisualTheme visualThemeId="MY_THEME" visualThemeSetId="BACKOFFICE"
description="My theme - Example (based on flatgrey)"/>
</entity-engine-xml>
```

a theme component can load one or more visual theme id. Usually only one is present.

The second important (but not mandatory) element is **implements**

```
  <extends location="component://common-theme/widget/Theme.xml"/>
```

This element indicates that your theme copies from the extend theme all information not present in its file definition.
If this element isn't present in your theme, you will need to define all information present in common-theme to be sure that OFBiz misses nothing for a correct run. Otherwise some functionnalities can be broken…

It's highly recommended to extend the common-theme to be sure that your theme works correctly and to surcharge only what you need.

The four following blocks are optionnal if you define an extends theme

General properties

This block contains all properties that the screen engine can use to prepare the rendering and that the theme can implement

```
    <widget-properties><!--Transversal properties relative to ofbiz widget component--
>
        <default-view-size value="20"/>
        <autocompleter
                default-view-size="10"
                default-min-lenght="2"
                default-delay="300"
                display-return-field="true"/>
        <lookup
                position="topleft"
                width="640"
                height="500"/>
        <layered-modal
                width="800"
                height="600"/>
    </widget-properties>
```

Theme's specific properties

This block contains all properties specific to this theme. In general these are some properties present on ftl template, that are initialized by the theme and can be surchaged by another theme through the extends elements.

```
        <property name="jgrowlPosition" value="center" type="String"/><!--possible
value: top-left, top-right, bottom-left, bottom-right, center-->
        <property name="jgrowlWidth" value="800" type="Integer"/>
        <property name="jgrowlHeight" value="" type="Integer"/>
        <property name="jgrowlSpeed" value="100" type="Integer"/>
```

Ftl macro library

This block defines for each technology implemented by the screen engine where it can find the macro library for each model (Screen, Form, Tree, Menu)

```
    <templates><!-- Freemarker template use by this theme to render widget model-->
        <template name="screen" type="html" content-type="UTF-8" encoding="none"
encoder="html" compress="false">
            <template-file widget="screen" location="component://common-
theme/template/macro/HtmlScreenMacroLibrary.ftl"/>
            <template-file widget="form" location="component://common-
theme/template/macro/HtmlFormMacroLibrary.ftl"/>
            <template-file widget="tree" location="component://common-
theme/template/macro/HtmlTreeMacroLibrary.ftl"/>
            <template-file widget="menu" location="component://common-
theme/template/macro/HtmlMenuMacroLibrary.ftl"/>
        </template>
        ...
    </templates>
```

If you want surcharge some macros, you can just create the desired macros and import the others from common-theme (at the top of file) like that : `<#include "component://common-theme/template/macro/HtmlFormMacroLibrary.ftl"/>`

Screen library

This block defines where OFBiz can find all official screens definitions in framework/common

Normally, you don't need to change this file except if you need to define a default screen style that doesn't exist OOTB. If you need to extend an existing one, you have to do it in the theme directory.

To define a new default screen style, you have to add it in this file, and point to the screen decorator in common-theme to define your default screen style as the default one.

```
      <common-screens><!--list all common screen and decorator global to each
application that each theme can be surcharge or not and use the screen present on
common theme-->
        <structural-decorator default-location="component://common-
theme/widget/CommonScreens.xml">
            <screen name="GlobalDecorator"/>
            ...
        </structural-decorator>
        <embed-decorator default-location="component://common-
theme/widget/CommonScreens.xml">
            <screen name="FindScreenDecorator"/>
        </embed-decorator>
        <general-screen default-location="component://common-
theme/widget/CommonScreens.xml">
            <screen name="geoChart"/>
            ...
        </general-screen>
     </common-screens>
```

Screens are separated in three types :

- structural-decorator : contains all decorators that organise the screens structures
- embed-decorator : decorator used only on sub screens
- general-screen : list all generic inter applications screens

6.7.4. The common-theme

This is the root theme that contains all information to ensure a good basic theme for OFBiz. Currently it keeps all old themes system for backward compatibility with ftl template managed by the entity **VisualThemeResource**

6.7.5. Create your own theme

As a theme is a component, you can create a new theme like a plugin.

After creating a component, you can add the two minimal information :

- Theme.xml file in **plugins/my-theme/widget/** with minimal information :

```
<theme name="my-theme"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/widget-theme.xsd">
    <visual-themes>
        <visual-theme id="MY_THEME" display-name="My Theme"/>
    </visual-themes>
</theme>
```

- your data file to add your visual theme in **plugins/my-theme/data/**

```
<entity-engine-xml>
    <VisualTheme visualThemeId="MY_THEME" visualThemeSetId="BACKOFFICE"/>
</entity-engine-xml>
```

The presence of VisualTheme entity helps to indicate which theme is available in your instance, specially helpful for tenant installations.

To display your theme in OFBiz theme library, you can complete the information on each visual theme like

```
<theme name="my-theme"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/widget-theme.xsd">
    <visual-themes>
        <visual-theme id="MY_THEME" display-name="My Theme">
            <description>My new funny theme under nice tecnno</description>
            <screenshot location="/mytheme/screenshot1.png"/>
            <screenshot location="/mytheme/screenshot2.png"/>
        </visual-theme>
    </visual-themes>
</theme>
```

display-name and **description** support the flexibleStringExpander syntax

extends common-theme

This is a first step to understand how the theme system works. With your new theme, you can try to surchage different elements.
To start, extends the common-theme :

```
<theme name="my-theme"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/widget-theme.xsd">
    <visual-themes>
        <visual-theme id="MY_THEME" display-name="My Theme">
            <description>My new funny theme under nice tecnno</description>
            <screenshot location="/mytheme/screenshot1.png"/>
        </visual-theme>
    </visual-themes>
    <extends location="component://common-theme/widget/Theme.xml"/>
</theme>
```

Now your theme should be operational, but without particularity.

You can surcharge a ftl macro, to do this create your own ftl macro file in **plugins/my-theme/templates/macro/HtmlFormMacroLibrary.ftl** with

```
<#include "component://common-theme/template/macro/HtmlFormMacroLibrary.ftl"/>

<#macro renderDisplayField type imageLocation idName description title class alert
inPlaceEditorUrl="" inPlaceEditorParams="">
    <#if description?has_content>
    *###*${description?replace("\n", "<br />")}**<#t/>
    <#else>
        *#&nbsp;#*<#t/>
    </#if>
</#macro>
```

Now indicate to your theme that you want use this library

```
<theme name="my-theme"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/widget-theme.xsd">
    <visual-themes>
        <visual-theme id="MY_THEME" display-name="My Theme">
            <description>My new funny theme under nice tecnno</description>
            <screenshot location="/mytheme/screenshot1.png"/>
        </visual-theme>
    </visual-themes>
    <extends location="component://common-theme/widget/Theme.xml"/>
    <templates>
        <template name="screen" type="html" content-type="UTF-8" encoding="none"
encoder="html" compress="false">
            <template-file widget="form" location="component://my-
theme/template/macro/HtmlFormMacroLibrary.ftl"/>
        </template>
    </templates>
</theme>
```

and check the result when you select your theme. The result isn't really interesting but it's to understand how it works.

create from scratch

TODO…

6.7.6. Backware compatibility with OFBiz 16.11 and above

How themes worked before

Before the theme management by model definition, all configurations have been present in the database through entity **VisualTheme** and **VisualThemeRessource**.
These ressources were loaded in a **layoutProperties** variable and used directly by decorator screens and ftl templates.

Now with the common-theme

All this logic is still present in the common-theme template to keep backward compatibility, but the VisualThemeRessource is now useless and properties have been migrated to the Theme definition in the part **theme-properties**

Example with BlueLight

The blue light theme has been these properties in VisualThemeRessource :

```
    <VisualTheme visualThemeId="BLUELIGHT" visualThemeSetId="BACKOFFICE"
description="BlueLight Theme: breadcrumbs, drop-down menus and rounded corners"/>
    <VisualThemeResource visualThemeId="BLUELIGHT" resourceTypeEnumId="VT_NAME"
resourceValue="BLUELIGHT" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_IMAGE_URL" resourceValue="/images/ofbiz_logo.png"
sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_SHORTCUT_ICON" resourceValue="/images/ofbiz.ico"
sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT" resourceTypeEnumId="VT_SCREENSHOT"
resourceValue="/bluelight/screenshot.jpg" sequenceId="01"/>

    <!-- CSS references -->
    <VisualThemeResource visualThemeId="BLUELIGHT" resourceTypeEnumId="VT_STYLESHEET"
resourceValue="/bluelight/style.css" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HELPSTYLESHEET" resourceValue="/bluelight/help.css"
sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_DOCBOOKSTYLESHEET"
resourceValue="/bluelight/webapp/bluelight/docbook.css" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT" resourceTypeEnumId="VT_STYLESHEET"
resourceValue="/common/js/jquery/plugins/asmselect/jquery.asmselect-1.0.4a-beta.css"
sequenceId="02"/>

    <!-- Javascript references -->
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/jquery/jquery-
1.11.0.min.js" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/jquery/jquery-
migrate-1.2.1.js" sequenceId="02"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/jquery/ui/js/jquery-
ui-1.10.3.min.js" sequenceId="03"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/jquery/plugins/asmselect/jquery.asmselect-1.0.4a-beta.js"
sequenceId="05"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
```

```
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/jquery/plugins/datetimepicker/jquery-ui-timepicker-
addon.min-1.4.3.js" sequenceId="07"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/jquery/plugins/mask/jquery.mask-1.14.13.min.js"
sequenceId="10"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/jquery/plugins/validate/jquery.validate.min.js"
sequenceId="12"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/util/OfbizUtil.js"
sequenceId="15"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/util/fieldlookup.js"
sequenceId="16"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/plugins/date/date.timezone-min.js" sequenceId="18"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/util/miscAjaxFunctions.js" sequenceId="19"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/util/selectMultipleRelatedValues.js" sequenceId="20"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/common/js/util/util.js"
sequenceId="21"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT"
resourceValue="/common/js/plugins/date/FromThruDateCheck.js" sequenceId="22"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_JAVASCRIPT" resourceValue="/bluelight/dropdown.js"
sequenceId="30"/>

    <!-- ftl references -->
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_HDR_TMPLT_LOC"
resourceValue="component://bluelight/template/Header.ftl" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_FTR_TMPLT_LOC"
resourceValue="component://bluelight/template/Footer.ftl" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_NAV_OPEN_TMPLT"
resourceValue="component://bluelight/template/AppBarOpen.ftl" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_NAV_CLOSE_TMPLT"
resourceValue="component://bluelight/template/AppBarClose.ftl" sequenceId="01"/>
    <VisualThemeResource visualThemeId="BLUELIGHT"
resourceTypeEnumId="VT_MSG_TMPLT_LOC"
```

```
resourceValue="component://bluelight/template/Messages.ftl" sequenceId="01"/>
```

Now it’s just

```
    <VisualTheme visualThemeId="BLUELIGHT" visualThemeSetId="BACKOFFICE"/>
```

And on theme definition

```
    <theme-properties>
        <!--javascript lib-->
        <property name="VT_HDR_JAVASCRIPT['add']" value="/bluelight/dropdown.js"
sequenceId="30"/>
        <!--Css style-->
        <property name="VT_STYLESHEET['add']" value="/bluelight/style.css"/>
        <property name="VT_HELPSTYLESHEET['add']" value="/bluelight/help.css"/>
        <property name="VT_DOCBOOKSTYLESHEET['add']"
value="/bluelight/webapp/bluelight/docbook.css"/>
        <!--template location-->
        <property name="VT_HDR_TMPLT_LOC"
value="component://bluelight/template/Header.ftl"/>
        <property name="VT_FTR_TMPLT_LOC"
value="component://bluelight/template/Footer.ftl"/>
        <property name="VT_NAV_OPEN_TMPLT"
value="component://bluelight/template/AppBarOpen.ftl"/>
        <property name="VT_NAV_CLOSE_TMPLT"
value="component://bluelight/template/AppBarClose.ftl"/>
        <property name="VT_MSG_TMPLT_LOC"
value="component://bluelight/template/Messages.ftl"/>
    </theme-properties>
```

Values with **/images/…** have been moved to the common-theme that bluelight extends, the theme definition keeps only what the theme adds to the extended theme.

property name supports the FlexibleMapAccessor syntax, so you can continue to populate a list (VT_STYLESHEET['add']), reset a list (VT_STYLESHEET[]) or add an element on the top list (VT_STYLESHEET[+0]) because some time the order libraries loading is important

Migrate you own theme

Easily, create you Theme.xml and move your VisualThemeResource in **theme-properties** like in the BlueLight example above.
Maybe you will need to update your template because the modelTheme return ressources not always as list. So :
`<property name="VT_HDR_TMPLT_LOC" value="component://bluelight/template/Header.ftl"/>` → return a String with `component://bluelight/template/Header.ftl`
`<property name="VT_STYLESHEET['add'] value="…"` → return a List<String>

6.8. SPA / FrontJs with Vue.Js

SinglePageApplication or GUI done with a front done with a Javascript framework

done with **Vue.Js framework**, and **Vuetify Material-Design library**

vuejs component included a **FrontJs-Renderer** (similar as Macro-Renderer) usable to handle user interface with modern javascript framework (Vue.Js, React, Angular, ect…) to be able to build a Single Page Application for one or more OFBiz component.

It use the standard OFBiz Widget System defined by XML and a vuejs application.

This component is currently at Work In Progress state, it aims to concretely build applications/components in order to discuss the development process and the use of the application.
It is produced in an "Agility" spirit which mean that it focus to result in despite of some technical debt. Currently there are still some TODO in the code and choice has been made to implement only xml tags and attributes which are used, at least, in one application/component.

XML tag or attribute not yet implemented generate warning or error.

This documentation aim to,

- Explain the current state of the component, how it work and which points had been choose or took away.
- Document how to use it to develop OFBiz application by explaining each specific item by example.

The two main goal of using a javascript framework for user interface is

1. to have a "modern" look and feel
   - using recent GUI library
   - usable in multiple context (PC, tablet, smartphone)
2. to increase interactive elements in screen, even if theses screens are based on standard modules :
   - update some part of the screen by action or by data update;
   - modify forms field as function of other field values;
   - simpler screen configuration without having to worry about screenlet interaction.

6.8.1. Vue.Js Renderer

Actually the first javascript application enabling to use portlet system - portal Apache OFBiz is written with the Vue.js framework.

Vue.js has been choose for its easy learning curve and its community driven development instead of a corporation.
In Vue.js, library vuetify has been choosen, because it’s base on material design and have e very

active community, so seem to be the most used vue.js GUI libraries.

POC Vuejs Renderer installation

1. From a standard trunk ofbiz-framework with 205914a1beb commit or superior (that can be downloaded at documentation standard SourceRepository using git).
2. Create the 'plugins' repository at ofbiz-framework root folder.
3. In the Apache OFBiz plugins repository, download the next repository using git :
    a. **example plugin** at http://svn.apache.org/repos/asf/ofbiz/ofbiz-plugins/trunk/example
4. In the ofbizextra plugins repository, download the 3 next repositories using git :
    a. **vuejsportal plugin** that can be downloaded at https://gitlab.ofbizextra.org/ofbizextra/ofbizplugins/vuejsPortal all vuejs components, a new ScreenViewHandler and 4 new renderers (screen, form, menu, tree) and all common files to all fjs component.
    b. **examplefjs plugin** that can be downloaded at https://gitlab.ofbizextra.org/ofbizextra/ofbizplugins/examplefjs.git specifics files for using vue.js with ofbiz example component
    c. **flatgreyfjs plugin** that can be downloaded at https://gitlab.ofbizextra.org/ofbizextra/ofbizplugins/flatgreyfjs.git dedicated theme for vue.js with vuetify library
5. In the ofbizextra git repository download ofbizJiraPatchAvailable project
If there are some files, some patch may be necessary for this POC, for each patch you can goto JIRA to have more explanation for each.
To apply them it is recommended to create a branch on your ofbiz, and apply the patchs with git command `git am patchName`. Currently these patchs are necessary:
    a. OFBIZ-11645_0001-simple-methods-optional-in-compound-file.patch ⇐ compound-widget files are used in POC
    b. OFBIZ-11676_0001-Fixed-drop-down-field-not-work-when-in-a-compound-wi.patch ⇐ compound-widget files are used in POC
    c. OFBIZ-11708_0001-fixed-sort-order-field-not-works-in-CompoundWidgets.patch ⇐ compound-widget files are used in POC
6. If there are some files in the ofbizCommit2add, it's some patch waiting jira creation or dedicated for this POC, so it's needed to apply them too with `git am fileName` at the ofbiz root directory (replace fileName by the patch file name and path)
7. After that you have to modify some files of ofbiz-framework (ex: model.java because some tag are added).
For that, a (bash) script exist, it's located in the 'plugins/vuejs' :
`./tools/applyOfbizFilesPatchs.sh`
And then copy some files
`rsync -a --exclude-from=./ofbizFiles/rsyncExcludes ./ofbizFiles/ ../../`
8. To be able to build the vue.js application you must have Node.js installed on your system.
    a. To install Node.js on a debian based system, execute the following commands :
    ```
    curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
    sudo apt-get install -y nodejs
    ```
    In case of another system, you can consult https://nodejs.org/en/download/package-

manager/#debian-and-ubuntu-based-linux-distributions

   b. Meanwhile an integrated gradle command

      i. Being located at 'plugins/vuejs/webapp/vuejs'

         - `npm i` ⇐ to install the project dependencies
         - `npm run build-prod` ⇐ to build the project (it's possible to use build-dev to be able to use browser dev-tools)

9. You can now launch OFBiz as regular :
   `./gradlew cleanAll loadAll ofbiz`
   (being located at the **root of ofbiz-framework**)
10. You can now connect to the application portal at https://localhost:8443/examplefjs/control/main

POC choices

Apache OFBiz **trunk branch** usage.

Principe and architecture of Apache OFBiz **Portal Page and Porlet** usage.

The whole use cases are applied **to example component** and during the POC some new files used which should to be located in common have been moved to examplefjs aiming to simplify the installation and update process.

With examplefjs, there is an **additional webapp (examplefjs)**, it centralize allowed URIs for vuejs component, currently this applicationName is listed in constantes.js file.
vuejs component is not bound to a component, it just need a base URI for send its requests. This uri is temporary set in constantes.js but it will be more flexible later aiming to use vuejs with multiple components

Portal component (https://localhost:8443/examplefjs/control/login) use **ofbiz standard login by cookies** mechanism, it will later use ofbiz login by token (cf. https://issues.apache.org/jira/browse/OFBIZ-9833).

screens, forms, menus used for portlets are defined **with xml**, and dedicated files for better readability.

There is **a component (vuejs) for each ofbiz screen element** (SingleItemRow ⇒ vue.single-item-row), which are defined by renderer for screen, form and menu (and for html renderer that correspond globally to macro.ftl). Actually, for the POC, to list them, you can search in vuejs existing file for `Vue*.vue` in webapp/vuejs/src/component/ .

XML tags used

During the POC, few new properties or XML tags had been added but some of existing tags and properties had been distorted for an exclusive Vue.Js use.

- screen
  - in `<container` tag
    container is the main piece of dynamics screen (update a part of screen on event)

    - `id` is used to identify the container and enable to inject content into it with `set-area` later. When set-area is done for this id, its content will be replace by the content received from area-target.
      So, do not resend the container !
    - `watcher-name` (new attribute) is a name in the local store (in web browser) which can contain some data (parameters). It's possible to update watcher content with a `set-watcher`. Each time the watcher content changes, the `auto-update-target` is called using watcher content as parameters to update the container.
      If `refresh-area` is call, the refresh is done by using auto-update-target and watcher content (if exist).
    - `auto-update-target` is used to bind a `watcher` to the container.
      To use it with set-watcher, a new attribute `watcher-name` has been added, and when the content of this this watcher changes a set-area is done with auto-update-target as target and watcher content as parameters.
      `auto-update-target` support the REST uri format with some parameters in it include with {}, value is populate with content of watcher content. If uri contain, at least one {xxx} or have no parameters a GET method will be used otherwise a POST will be used.
  - screenlet
    - collapsible, if collapsible = "true" then it's necessary to have a "unique" id to know what collapse when button is clicked. It's possible to have multiple screenlet with same id or each with a different one.
      So, it's important to give a `id` if collapsible ="true" otherwise collapse will collapse all screenlet without id !
    - id : is manatory if the screenlet should be collapse
  - in `platform-specific`-`html` tag a new tag
    - `vuejs` is to call a specifics vuejs component named `component-name`.
      it's possible to give to it some parameters with sub tag `parameters`
- form
  - property `entity-name` is mandatory for grid or single form, vuejs use it to build entity store. When entity-name, exist data print to screen are associated to the web-browser local store, so when a data change in one screenlet, same data in an other screenlet will change too.
  - in `<on-event-update-area` tag (there can be many of them and they will be executed in xml apparition order. It will wait current task is completed before starting the next one. If any of these events is set, the submit button will prevent the form default behaviour, so if you need to post in addition to vuejs feature you must add an event-type `submit`)
    - `event-type`
      Define the action to be executed during the form submitting (click on submit button). It can take one of next values :
      - `submit` will submit the form to `area-target` with provided `parameter` elements. Like `set-area` if no `parameter` are defined it will use the whole form. Error and success messages will be displayed.
        Form `target` support the REST uri format with some parameters in it include with {}, value is populate with content of form - fields when submit.

The http method used is POST, but you can add a field _method to choose the method received by the ofbiz controler (ex: PUT for an update)

    - `set-area` update area with id `areaId` with the post result at URI `area-target` with included `parameter` elements.
      If uri contain, at least one {xxx} or have no parameters a GET method will be used otherwise a POST will be used.
    - `refresh-area` similar with set-area but target used will be the `auto-update-target` define at the container level and, if a watcher-name is define for this container, it's content will be used as parameters (for solving {} in target or as parameters if no {} in target.
    - `set-watcher` update watcher `areaId` with included `parameter`. In case no parameter was passed to `set-watcher` it will use all form fields to update watcher content.
    - `refresh-watcher` all container which watch a watcher-name are update when watcher content change, with refresh-watcher update is done without change watcher content.
    - `collapse` do an action on collapse/expand screenlet.
      `area-target` is the action to do : **collapse** or **expand** or **toggle**;
      `area-id` is the `screenlet name` to change.
    - `close-modal` close the current modal, currently `area-id` is necessary, when modal is open for a lookup areaId is {formName}_{fieldName}; area-target is not used, so can be left empty.

- in `<on-field-event-update-area` tag (there can be many of them and they will be executed in xml apparition order. It will wait current task is completed before starting the next one.)
  - `event-type`
    Define the action to be executed during the click on the associated hyperlink (<a link or button). It can take one of next values :
    - `post` will post at URI `area-target` with the `parameter` elements included in the tag. If there are no parameters, it will use the whole form. It will display error and success messages.
    - `put` will post at URI `area-target` with the http methof PUT if target contain {xxxx} xxxx will be replace by value if xxxx is define in parameters included in the tag. If there are no parameters, it will use the whole form. It will display error and success messages.
    - `delete` will post at URI `area-target` with the http methof DELETE if target contain {xxxx} xxxx will be replace by value if xxxx is define in parameters included in the tag. If there are no parameters, it will use the whole form. It will display error and success messages.
    - `set-area` update area with id `areaId` with the post result at URI `area-target` with included `parameter` elements.
      If uri contain, at least one {xxx} or have no parameters a GET method will be used otherwise a POST will be used.
    - `refresh-area` similar with set-area but target used will be the `auto-update-target` define at the container level and, if a watcher-name is define for this container, it's

content will be used as parameters (for solving {} in target or as parameters if no {} in target.

        - `set-watcher` update watcher `areaId` with included `parameter`. In case no parameter was passed to `set-watcher` it will use all form fields to update watcher content.
        - `refresh-watcher` all container which watch a watcher-name are update when watcher content change, with refresh-watcher update is done without change watcher content.
        - `set-field-in-form` will set the value of property `area-target` from form `area-id` from the actual field value.
        - `collapse` do an action on collapse/expand screenlet.
          `area-target` is the action to do : **collapse** or **expand** or **toggle**;
          `area-id` is the `screenlet name` to change.
        - `close-modal` close the current modal, `area-id` is to know which modal close, when modal is open for a lookup areaId is {formName}_{fieldName};
          if area-id is empty all modal are closed;
          area-target is not used, so can be left empty.
    - in `parameters` tag
      - `auto-parameters-portlet` allow to automatically fill necessary portlet parameters :
        - `portalPageId`
        - `portalPortletId`
        - `portletSeqId`
        - `currentAreaId`
    - in `<hyperlink` tag
      - `link-type="anchor"` is used to make an internal call to FrontJs via `set-area`, `target-window` is used as id to update and `target` as area-target, parameters as parameters.
      - `target-window` is used to design the watcherName that have to be updated.
      - `url-mode` three new modes is added to be able to give the http method to use, POST is the default
        - intra-post : do a http POST request for the set-area, and so sending multiple parameters
        - intra-put : do the http request with a PUT method
        - intra-delete : do the http request with a DELETE method
- menu
    - in `<link` tag
      - `link-type="set-area"` is used to make an internal call to FrontJs via `set-area`, `target-window` is used as id to update and `target` as area-target, it's a http get so parameters should be in target and if there area some {xxx} in target they will be replace by correct value with fields list in parameters.
      - `link-type="auto"` (default value) and `url-mode="intra-app"` call the target-url to print a

new page (using internal routing).

- `link-type="layered-modal"` is used to open a modal windows and do a set-area in it
- `url-mode` three new modes is added to be able to give the http method to use, POST is the default (except for set-area)
  - intra-post : do a http POST request for the set-area, and so sending multiple parameters
  - intra-put : do the http request with a PUT method
  - intra-delete : do the http request with a DELETE method

decorator, lookup

Currently vuejs app is call in a ftl included in the classic mainDecorator, without application menu (which are directly manage by vuejsApp cf Specifics URI).

Currently a specifics theme has created, to be able to remove most of css to avoid "conflict" with vuetify lib. This theme is forced in the main decorator of the application.

Lookup

1. `LookupDecorator` is a specifics one for VueJs, but its usage is same as the standard one. It contain 2 main specificity
   - for autocomplete return, all data are in viewData
   - for search - result, when lookup receive parameters.lookupResult = "Y" only result form is sent to be injected in area lookup-results
2. `LookupForm - find` :
   - it's necessary to add a hidden field

     ```
         <field name="lookupResult"><hidden value="Y"/></field>
     ```

   - it's necessary to add 2 on-event-update-are,
     first one to send result in the area lookup-result second one to collapse Find screenlet (if you want collapse it

     ```
             <on-event-update-area event-type="set-area" area-id="lookup-results"
     area-target="LookupExample"/>
             <on-event-update-area event-type="collapse" area-id="LookupFindScreenlet"
     area-target="collapse"/>
     ```
3. `LookupForm - list` :
   - for the link return field, 2 on-field-event-update-area must be added
     first one to set Field which call the lookup with the correct value
     second one to close the modal

```
            <on-field-event-update-area event-type="set-field-in-form" area-id="SelectExample" area-target="exampleId" />
            <on-field-event-update-area event-type="close-modal" area-id="SelectExample_exampleId" area-target="" />
```

4. `FindScreenDecorator`, is a specifics one for VueJs, but its usage is same as the standard one. Its specificity is about ListResult update. Some modifications should be done in the find form
   - add a hidden field onlyList

     ```
                 <field name="onlyList"><hidden value="Y"/><!-- used by FindScreenDecorator to only send list result --></field>
     ```

   - add a on-event-update-area to update List result area

     ```
              <on-event-update-area area-target="findExample" event-type="set-area" area-id="search-results"/><!-- When parameters-map is empty all form field are sent  -->
     ```

   - add a on-event-update-area to collapse find screenlet (optional)

     ```
            <on-event-update-area event-type="collapse" area-id="findScreenlet" area-target="collapse"/>
     ```

FrontJs Renderer

A **new viewHandler** and a new set of **renderer** had been created.
A new package had been created in org.apache.ofbiz.widget.renderer named frontJs which contain all new renderer.

In this package, there is a new FrontJsOutput class which allow to construct necessary elements whit the wished format. An object of this class is instantiate at the begin of viewHandler process, then is completed by rendered object call.

Uri **showPortletFj** of component examplefjs use the new viewHandler to return screen render result, showPortlet.

This URI is used by Vue.Js application for display information gathering.
New viewHandle return a json which contain 2 maps (viewScreen and viewEntities).
viewScreen contain all information about display. viewEntities contain all information about data set.

In future revision, it will probably be feasible to only receive the data map.

Specifics URI

there is a **showPortal** URI in vuejs (CommonScreen), which is the Start point of the VueJs application.

showPortlet screen is redefine in this POC for simplification purpose, to only do what we want.

applicationMenu is a new uri which return with the FrontJsScreenViewHandler the data (ViewScreen & ViewData) from application xml menu.
In each fjs component, it’s necessary to define a view-map entry applicationMenu pointing to a screen.xml with only the correct menu.

REST URI

For OFBiz applications URI structure is : {resourceName}/{cover}/{Pkvalue: .*}

So, for Example resource, it translate into :

URI	method	Goal
Example/find	get	find form to select options for list
Example/list	get (via post)	Example list, ⇒ with a get (so without parameters) all the examples ⇒ with a post (parameters and _method="GET") examples selected by options
Example/create	get	create form
Example/edit/{exampleId}	get	edit form and data for the id sent
Example/show/{exampleId}	get	show form and data for the id sent
Example/summary/{exampleId}	get	summary form and data for the id sent
Example/data/{exampleId}	get	just data for the id sent
Example/change	post	create a example
Example/change/{exampleId}	put (via post)	update example with exampleId
Example/change/{exampleId}	delete	delete example with exampleId

All these actions([cover]) is about screenlet (part of a page), if cover should be for a page, its name is suffix by Pg.

HTTP request other than POST method cannot have parameters (other than in URI) so to be able to do a GET or a PUT with parameters a workarround is to used a POST with added `<field name="_method"><hidden value="GET">` (or "PUT") in the form (or as one of the parameters).

TOMCAT httpServlet.java not manage PATCH method, so it's not usable

Added field

A new field had been added to PortalPagePortlet entity : **watcherName** which is the name of the watcher that fire portlet update.

ftl

ftl file can be mixed with Vue.js app, so it's necessary to create a dedicated component by ftl migrated. Look at XML tags used on point `platform-specific-html` - `vuejs`.

POC current situation

Currently VueJs POC is only able to manage PortalPage, but it will be simple to manage all other screen types.

The Objective here is to make different use cases portal page to work.
At the menu, there are different pages, actually the name of portal pages name are MgmtPageFrontJs and RecapPageFrontJs.
The first page aim to find and edit examples. The second page aim to show all properties of a given example.

For each portlet present in theses pages, a portlet component is created in Vue.js with the portlet name which initialise itself with showPortletFj.

ShowPortletFj give the component all information about components it have to render (viewScreen) and data set it have to work with (viewEntities). Actually ShowPortletFj respond with a JSON.

[Imbrication des container] | *PortalPageDetail-Example.svg*

ViewScreen and ViewEntities

Actually FrontJsViewHandler send all information to component who need update. In a client side logic, ViewScreen would be send only the first time while the portlet call information then would only care about ViewEntity which update the data set it work on.

Actually "use-when" are handled by ofbiz in model model management whatever level they belong to (screen, form, field, menu), so we can't get this information in the renderer. Later, we had to transmit to renderer all screen elements and if there was "use-when" for manage them in the frontend. This way next call could sent "use-when" correct value to properly update the final screen.

POC’s remaining todos

This list contain only points which are already meet in use case process.

Most of time, each new use case (or portlet) give new task (ex: new field type not yet meet, or field properties not yet process)

Minors

- review and change field-find to include title in the field to remove the "title" coloumn in find form
- Manage the ConfMod in the showPortalPage, to show (and manage) the portlet parameters page.
- do a uri to send label for login screen, to be able to have translation in login screen
- Include header and footer in the VueJs application to have a full page manage by VueJs, and so have a complete Vue.Js look.
- Include all Common and Commonext part, like systemInfoNote

Majeurs

- use VuejsRouter to be able to manage all screenType
- showPortletFj* must verify **security** about field SecurityServiceName and SecurityMainAction.

PortletWidget migration

Start of documentation for all portletwidget migration

1. create/copy the screen in {component}PortletScreen and remove <portalPortlet properties
    - portletTypeId
    - component, subComponent, useScreen
        i. add screenName and screenLocation
    - useScript, useMenu
2. copy the forms (and update screen if necessary)
3. update screen with portletUiLabelDecorator
4. update screen with screenlet and container if menu need it (id for setArea)
5. update screen with script, if necessary
6. update screen with menu, if necessary
7. copy from controller for uri edit and action
8. in Menu, replace

```
          <show-portlet portlet-id="${portalPortletId}" target="EditPersonPt" area-
id="${subAreaDivId}"
                         image-location="${iconsPurpose.Edit}" image-
title="${uiLabelMap.IconsTooltips_Edit}">
              <parameter param-name="partyId" from-field="parameters.partyId"/>
          </show-portlet>
```

```
with
```

```
            <link  text="Edit" target="editperson" link-type="anchor" target-
window="viewPartyInfo"> <!-- usage du target-window pourdonner l'id de destination
-->
                <parameter param-name="partyId" from-field="parameters.partyId"/>
                <parameter param-name="DONE_PAGE" from-field="DONE_PAGE"/>
                <auto-parameters-portlet/>
<!--                 TODO is it necessary to have 3 parameters not auto-parameters
-->
<!--                 <parameter param-name="portalPageId"    from-
field="parameters.portalPageId"/> -->
<!--                 <parameter param-name="portalPortletId" from-
field="parameters.portalPortletId"/> -->
<!--                 <parameter param-name="portletSeqId"    from-
field="parameters.portletSeqId"/> -->
                <image src="${iconsPurpose.Edit}"
title="${uiLabelMap.IconsTooltips_Edit}"/>
            </link>
```

POC’s use cases

Operational

All example management by the two portal page ExampleMgmt et ExampleRecap.
On the find, the sub list are not yet manage.

Next step

Party component will the next step whith most of its functions

Vue.js

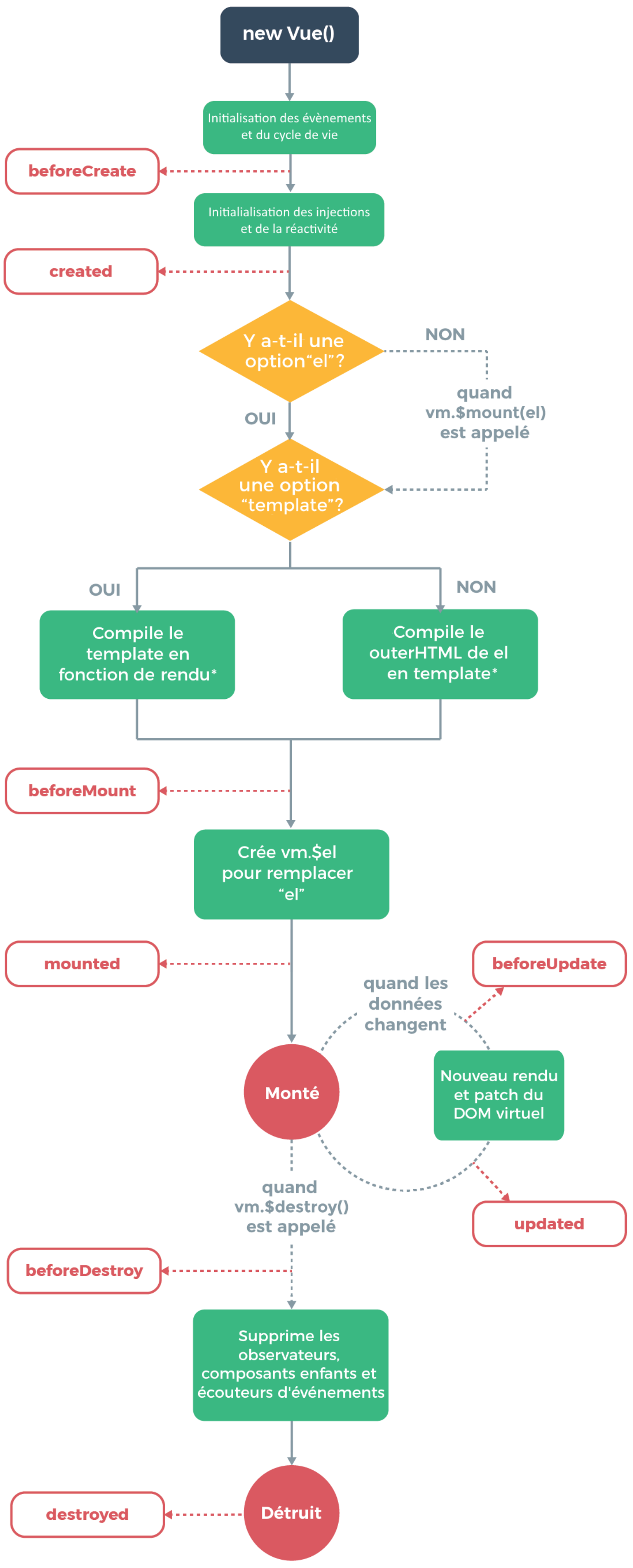
Vue.js general

todo

Components

A Vue.Js component is defined by 3 distinct blocks (template, script, style) which are regrouped in a single '.vue' file.

1. Template :
    - Template must be contained in a single root element (div, span, table, ect…)
    - In template we can access computed properties and function with double curly bracket **{{}}**
    - Classic html attributes can be preceded by ':' to use script properties and js code instead of plain text
    - Vue.Js give us some directives (v-for, v-if v-on:click, ect…) that are bind to script context and work like a classic attribute preceded by ':' and so can bind to script part and interpret js.
2. Script
    - Script section is wrapped into an `export default {}` that contain the whole script elements.
      TIP: import have to be made before this export
    - This export is a map that can contain predefined keys used by Vue.Js (data, computed, methods, props, created, ect…)
        - data() is a function that contain variable used is the component context.
          All variable contained in data() are reactive, which mean that Vue.Js will track any change on them and will reverberate theses changes everywhere there are used to reevaluate render.
        - computed is a map that contain evaluation based on reactive data.
          They can be used like a reactive variable in the component and will reevaluate itself when one of its entry had changed.
        - methods is a map containing helper function (handleClick, doPost, ect…).
          WARNING: these functions don’t aim to be reactive.
        - props is an array of string who give attribute to the component for passing data from its parent to it.
        - This section can also contain **hooks** of the component life cycle (created(), mount(), beforeUpdate(), ect…).
          Theses block of code will be executed when the hook is fired.
          See below :


new Vue()
Initialisation des évènements et du cycle de vie
beforeCreate
Initialialisation des injections et de la réactivité
created
Y a-t-il une option"el"?
NON
quand vm.$mount(el) est appelé
OUI
Y a-t-il une option "template"?
OUI
NON
Compile le template en fonction de rendu*
Compile le outerHTML de el en template*
beforeMount
Crée vm.$el pour remplacer "el"
mounted
beforeUpdate
quand les données changent
Monté
Nouveau rendu et patch du DOM virtuel
updated
quand vm.$destroy() est appelé
beforeDestroy
Supprime les observateurs, composants enfants et écouteurs d'événements
destroyed
Détruit

* la compilation est faite à l'avance si vous utilisez
une étape de build, par ex. des composants monofichiers

3. Style
   This part is dedicated to internal css of the component.
   Style of this part is applied before other style of the project.

Vuex

Vuex is the centralised state system of the application. It allow us to create 'store' which contain data that can be accessed/modified by any component of the application.

A store is composed of 4 elements :

1. `State`
   State is a map that contain all stored informations.
   State can only be modify by `mutation`. Default state’s data will be reactive.
   If Data had to be added in the state after its creation, they must be added with the method `Vue.set()` for them to be reactive.
2. `Mutations`
   Mutations is a map that contain function allowed to alter state. By convention, mutation’s key have to be upper-case. Mutations can’t be accessed through code, they must be called by actions.
   Mutations must be synchronous.
3. `Actions`
   Mutation first aim to fire mutation.
   Actions can be asynchronous, in which case there return a promise.
4. `Getters`
   Getters allow to access state data.
   Getters are reactive.
   Getters can use parameters, if they are they return a function instead of a reactive variable.

Store can be split into modules who can be described in an index.js file, stores can so be stored into a modules repository.

Reactivity

todo

6.8.2. modular and generic UI

Aiming to have an open and modular ERP, it’s important that the user interface configuration allow to manage multiple screens from a set of screen "module" for manage **same business object** type in **different business context**.

Next chapters describe a user interface management system’s POC aiming to respond to theses modularity needs.

Use cases for POC-UI

Theses use cases are to be used for new UI POC, documentation associated and selenium unit task test.

All these use cases should be done with existing entities and services, if it’s necessary to develop one, simplify the use case, the goal is UI, not service or entity.

These use case description are done in a agile philosophy, only main point is present and during realization details choices will be discuss and done.

Preliminary remarks :

1. In this document, the term "application" corresponding to "plugin component" in the ofbiz terminology which is not same as a "applications/trunk component" in ofbiz terminology. An application is dedicated for a business purpose for a user type, it’s build by assembling piece of ofbiz components, sometimes without any specifics entities, services and screen (ex: CRM-B2C, CRM-B2B, SFA are 3 applications uses by sales men)
2. Each use case is on an "application" and is part of one of the menu of this application.

   Of course, this document describe only menu-option needed by the use-case. As it’s difficult to do a clear definition of "web-page" because it’s depending of theme/template, use case is for me at two level :

   a. screen, which can be well define
   b. page, which depend on default theme

   Of course, some of use case (screen or page) will be done by a previous one (ex : sometime edit is done by the same screen as add). It’s even, one of the re-usable important point (UI, Doc and Selenium)
3. Each time a line (or point) start by a "?" the question is "is this point is needed ?"

Release and goal

Goal is the POC realization, not doing multiple super applications. POC realization should generate discussion and decision, each time be careful to mainly discuss on UI, Doc or Selenium and not on use case business justification. Use case are to be realistic but mainly as a support for a specifics UI, Doc or Selenium needed.

V1

Main Goal is to proof the technical points. Of course UI re-organization is the driver for this phase, documentation start to work in parallel but will wait first realization to test integration for help. For selenium unit task, it will be a simple usage of the new UI with the default theme, the sole purpose being to check that test with default theme is simple.

So never mind if all cases is not done, the list exist to give different classic case what we will need to know how to process it. Some case which seem to be duplicate from other can be use for beginner who want to help and check if the concept is understanding.

During V1 realization, UseCase list will be updated

V2

Main Goal is to check if the solution is **useful for the different type of contributors**

- experiment developer
- beginner developer
- functional consultant for parameters
- functional consultant for personalization
- ? end user - for parameters ? (if it’s still possible with the new architecture)

Use Case list will be use as a deployment plan. The list contains similar case to these which are realize in V1, so those on V2 can be achieve by all types of contributors.

Documentation Goal

- Apache OFBiz User documentation (asscidoc)
- Apache OFBiz web site wiki
- OFBiz Help

Selenium test

1. demo data for each use case and scenario
2. selenium scenario test for each page (or page group)
3. selenium unit test for each screen

Simple Party Management

Which Sub-Application

1. in HR application, simple employee management
2. in CRM B2C application, simple customer (person) management
3. in eCommerce, simple profile page
4. in HR application, simple organization/place (group) management
5. in CRM B2B application, simple customer (company) management
6. in Facility application, simple worker management

Which Party sub-component

1. Party - Person - PartyGroup
2. Contact Mech,
    - with postal address, phone and mail;
    - one or two fixes purpose (ex: phone fix number and mobile phone number)
3. Role
4. Party Identification

5. Party association
6. Not ~~**UserLogin**~~ because all Security entities should be use and it will generate a too large domain for this POC

Which Screen

1. Party
   - find, list
   - A person
     - add / edit, show
   - A group
     - add / edit, show
   - A company
     - add / edit, show
   - show a synthesis view (party/person/company, contact informations, roles, Identification)
     - Person
     - Company
     - PartyGroup
2. Contact information
   - all contact informations (for one party / facility)
     - with and without purpose
     - with and without history
     - deactivated
   - add / edit postal address
   - add / edit mail
   - add / edit phone
3. Role
   - list for a party
   - add a role (for a parent RoleType)
   - add a role in two step :
4. select parent RoleType
5. select the role
   - remove a role
6. Party Identifications
   - list, add, remove

HR Employee management

In HR Component, starting person management with the more complete form about person.

- Menu option to manage employee
  - find, list, show, add, edit and manage his
    - contact information
    - identification (3 idTypes, one mandatory, two optionals)
- template page with a header (or sidebar or …) to show on which employee we are

Use Case Screen :

1. find Person
   - simple form (only on party or person)
   - with an add button (which can be show or not depending on parameter or authorization)
2. Person list with an add button (which can be show or not depending on parameter or authorization)
3. add a Person
4. show a Person
5. show a Person with sub-menu with two options : contact informations and Identifications
6. edit a Person
7. List of all contact informations for a person, with an add button (which can be show or not depending on parameter or authorization)
8. add a postal address
9. add a mail
10. add a phone number (to go step by step, without purpose management, will be done in next Use Case group)
11. edit a postal address
12. edit a mail
13. edit a phone number
14. List of all identification number for a person, with an add button (which can be show or not depending on parameter or authorization)
15. add a identification number with choice of identification type
16. edit a identification number with choice of identification type
17. add a identification number with a fix identification type
18. edit a identification number with a fix identification type

Use Case Page :

1. create a person

2. search a person
3. visualize a person
4. manage informations about a person
5. template page with a header (or sidebar or …) to show on which employee we are, (for example to show all his knowledges, or his skills, or his positions, or his …)
6. manage informations about a person on one page, and with access at this page directly by a field (auto-completion on id, first, last name)

CRM B2C, customer mgnt

In a CRM B2C application, the customer (so, in this context, a person) management.
The difference from previous use case group is :

1. person form is more simple than in HR
2. role will be used to characterize customer position (suspect, prospect, with_Quote, customer)

Menu option to manage employee

- find (with role field), list, show, add, edit and manage his
  - contact informations
  - identification (3idTypes, one mandatory, two optionals)
- template page with a header (or sidebar or …) to show on which customer we are

Use Case Screen :

1. find Person with an add button (which can be show or not depending on parameter or authorization)
   - search field same as in HR find person
   - role field which can appear or not, when not appear a fix value has been put as parameters.
   - contact information field, phone, mail, town. These fields can be show or not by the user with a "deploy" button
2. Person list with an add button (which can be show or not depending on parameter or authorization)
   - role field appear or not, when not appear a fix value has been put as parameters, so only person with this role appear
3. add a Person, all main informations in the form
   - role
   - less field about person than in HR form
   - 1 postal address
   - 2 phone number
   - 1 identification number

4. show a Person, all main informations in the screen with indicator for contact information and identification when there are more data that what it’s show.
5. show a Person with sub-menu with options :
   - contact informations
   - Identifications
   - role history
   - change role : a direct action button
6. edit a Person, only "Person" field
7. a button bar to change role (ex: for a suspect, there are the 3 options), this use case is for having a action bar, in this business process case it’s maybe not a need, but for more complex object like order or task, it’s a classical need.
8. List of all contact informations for a person, with one or multiple add buttons (which can be show or not depending on parameter or authorization) and purpose are show, it’s the second step, with purpose management.
9. add a postal address (or just a purpose)
10. add a mail
11. add a phone number
12. edit a postal address
13. edit a mail
14. edit a phone number
15. List of all identification number for a person, with an add button (which can be show or not depending on parameter or authorization)
16. add a identification number with choice of identification type
17. edit a identification number with choice of identification type

Use Case Page

1. create a new entry in CRM (role is choose during creation)
2. search a "customer" (or suspect, prospect, …)
3. visualize a "customer"
4. manage informations about a "customer"
5. template page with a header (or sidebar or …) to show on which "customer" we are, (for example to show all his quotes, or his orders, or …)
6. manage informations about a person on one page, and with access at this page directly by a field (auto-completion on id, first, last name)

eCommerce, profile page

A simple profile page.
The difference from previous use case will be mainly on Use Case Page because eCommerce theme

could be more original and public user interface should be, most of the time, more simple.

Use Case Screen :

1. show the person, all main informations in the screen with indicator for contact information and identification when there are more data that what it’s show.
2. show the Person with sub-menu with options :
    - contact informations
    - Identifications
3. edit a Person, only "Person" field
4. List of all contact informations for a person, with an add button and purpose are show, purpose is need for invoice or shipping.
5. add a postal address (or just a purpose)
6. add a mail
7. add a phone number
8. edit a postal address
9. edit a mail
10. edit a phone number

Use Case Page :

1. visualize the profile (the person) with edit button
2. manage his contact informations
3. manage his identifications
4. All in one page, which can be look as a long page.

HR organization mgnt

In HR component, a simple organization/place (group) management.
Now PartyGroup management (very simple), but with complex screen to manage hierarchy. In this use case group we will use the word "group" for service or department, or subsiadiry.

- Menu option to manage the Company organization
    - manage group
    - associated employee in a group
    - manage a hierarchy of group

Use Case Screen :

1. find group (with a specific partyType)
    - simple form (only on party or partyGroup)
    - with an add button (which can be show or not depending on parameter orauthorization)

2. PartyGroup list with an add button (which can be show or not dependingon parameter or authorization)
3. add a group
4. show a Person, all informations in screen with sub-menu with two options : contact informations and Identifications
5. edit a Group
6. List all contact informations for a group, with an add button (which can be show or not depending on parameter or authorization)
7. add a postal address
8. add a phone number
9. edit a postal address
10. edit a phone number
11. List all identification number for a group, with an add button (which can be show or not depending on parameter or authorization)
12. add a identification number with choice of identification type
13. edit a identification number with choice of identification type
14. add a identification number with a fix identification type
15. edit a identification number with a fix identification type
16. List all person associated to the group with two add buttons (which can be, individually, show or not depending on parameter or authorization)
    - add a manager
    - add a member
17. List all group associated to the group (the child) with two add buttons (which can be, individually, show or not depending on parameter or authorization)
    - add an existing group as a child
    - create a new group and add it as a child
    - in the list, each group is a link to this screen, to be able to navigate top-down
    - a third button to go to the parent level, to be able to navigate bottom-up
    - the name of the group manager appear above the list
18. ? List all parent group for a group or for a person ?
19. show group hierarchy as a tree with action or detail at each level, top-down
20. show group hierarchy as a tree with action or detail at each level, bottom-up

Use Case Page :

1. search a group
2. manage a group
3. manage its contact informations

4. manage hierarchy step by step (parent to child or child to parent)
5. manage hierarchy with a tree view
6. in HR employee, show the tree, top-down or bottom-up with the template "for an employee"

CRM B2B customer mgnt

In a CRM B2B application, the customer (so, in this context, a company) management.
For clarification, in these Use Cases, B2B is an other application than B2C.
The "CRM B2C & B2B" will be a third, but not in this list because it contains no specificity on screen-page definition

The main difference between B2C is :

1. company versus person,
2. contact management with PartyAssociation
3. ? customer organization management ?

Use Case Screen :

1. find customer (a company (specific partyType)) with an add button (which can be show or not depending on parameter or authorization)
    - search field are on multiple entities with some part deploy or not
    - role field which can appear or not, when not appear a fix value has been put as parameters.
    - contact information field, phone, mail, town. These fields can be show or not by the user with a "deploy" button
2. Company list with an add button (which can be show or not depending on parameter or authorization)
    - role field appear or not, when not appear a fix value has been put as parameters, so only company with this role appear
3. add a Company, all main informations in the form
    - role
    - field from PartyGroup
    - 1 postal address
    - 2 phone number
    - 2 identification number
4. show a Company, all main informations in the screen with indicator for contact informations and identification when there are more data that what it’s show.
5. show a Company with sub-menu with options :
    - contact informations
    - Identifications
    - role history

    - change role : a direct action button

6. edit a Company, only "Company" field
7. a button bar to change role (ex: for a suspect, there are the 3 options), this use case is for having a action bar.
   In this business process case it’s maybe not a need, but for more complex object like order or task, it’s a classical need.
8. List of all contact informations for a company, with an add button (which can be show or not depending on parameter or authorization) and purpose are show, (so, with purpose management).
9. add a postal address (or just a purpose)
10. add a mail
11. add a phone number with purpose
12. edit a postal address
13. edit a mail
14. edit a phone number
15. List of all identification number for a company, with an add button (which can be show or not depending on parameter or authorization)
16. add a identification number with choice of identification type
17. edit a identification number with choice of identification type
18. list of contact (person) associated to this company with an add button (which can be show or not depending on parameter or authorization)
    - a contact is a person with contact information
    - list with only one line per contact
    - list of block with contact details for each
19. edit a contact or his contact information

Use Case Page :

Exactly the same as the CRMB2C

1. create a new entry in CRM (role is choose during creation)
2. search a "customer" (or suspect, prospect, …)
3. visualize a "customer"
4. manage informations about a "customer"
5. template page with a header (or sidebar or …) to show on which "customer" we are, (for example to show all his quotes, or his orders, or …)
6. manage informations about a company on one page, and with access at this page directly by a field (auto-completion on id, first, last name).

Facility worker mgnt

In Facility application, simple facility’s worker management.
For this last use case group, it’s a simplification of the previous one.
Only a very simple and short process for adding people.

It’s the last one, because the goal is to check if it’s easy and rapid to create (or parametrize) a new small application from existing one.

In the Warehouse Management application (simple version OOTB)

- in the administration menu
    - the user menu to manage internal user per facility In the standard business process, it will be used mainly for login and authorization, in our case we will only manage person, his phone number and his facility (where he’s authorized)
    - the facility menu to manage contact informations and person authorized

Use Case Screen :

Already existing screen used

1. find Person
    - simple form (only on party or person)
    - with an add button
2. Person list with an add button
3. add a Person, simple form 3-6 fields
4. show a Person
5. show a Person with sub-menu with option to manage contact informations
6. edit a Person
7. List of all contact informations for a person, with one or multiple add button
8. add a mail
9. add a phone number
10. edit a mail
11. edit a phone number

New Screen

1. add a facility, simple form, if service exist, including some contact informations
2. List of all existing facility
3. List of all contact informations for a facility, with one or multiple add button
4. List of all persons associated to the facility, with two add button
    - add an existing person

    - create a new person and add it to the facility
5. List of all facility associated to a person, with one add button
    - add an existing facility

Use Case Page :

1. manage facilities
2. manage persons
3. visualize a facility details (info, contact informations, persons associated)

6.8.3. Screenlet autonomous

in a modular approach, screenlet is an autonomous part of the screen, which mean that there can be action that only alter this part of the screen, mainly fired by itself.

Sreenlet allow a huge modularity gain in user interface, that way an user action (click on a link, type in a field, ect…) must not precise the Sreenlet itself but a logical name which can be subscribed by one or more screenlet.

This logical name which is subscribed by screenlet is called `watcherName`, this field is a new attribute for the `container` tag.

6.8.4. FrontJs Glossary

watchers

They are objects stored in js store that can be altered remotely. watcher's subscribers update themselves with the new value when it change.

watcherName

It is the key of a watcher

7. Core APIs

8. Development environment

8.1. Setup your environment

8.1.1. Java SE

8.1.2. IDE

Eclipse

Intellij Idea

8.1.3. Database

8.2. Web tools

The Webtools application is the UI-gateway to all the framework functions.

8.2.1. Help for Webtools Main page.

This is the default screen for the Webtools application.
Several links are present on this page to access specific tool screens directly.
Using the application menu you can select the tool you need.

- The Logging section is used to view and configure the OFBiz system logs.
- The Cache & Debug section is used to monitor the OFBiz cache system status. You can even set or clear some cache content or force Garbage collection with this tool.
- The Artifact Info section is used to navigate through all OFBiz artifact files. When accessing this section the complete OFBiz code base is scanned and a list of all artifacts is offered to the user to be navigated. Please note that the initial scan can take a while to be completed.
- The Entity Engine section is used to interact with the entities defined in the system. You can view the entity structures, search for entity content, navigate though related entities, etc.
- The Service Engine section is used to interact with the services defined in the system. You can view all services details, monitor the jobs that are running, the active threads. You can even manually run a service or schedule a periodic/delaied job execution.
- The Import/Export section is used to transfer entity content from the OFBiz system to external systems and viceversa. Various import/export systems and formats are available.
- The Configuration section is used to set parameters for the OFBiz system.

9. From Mini Language to Groovy

This is a small guide for everybody involved in converting the Mini Language into Groovy.

Why is this important?
This tutorial is directly linked to the efforts of converting all scripts in Mini Language to newer Groovy Scripts. All of this is done, because Groovy is much more readable and easier to review, more up to date and many other reasons, which can be found here: Proposal for deprecating Mini Language

To contribute, or just be up to date with the current process, you can look at the existing JIRA issue OFBIZ-9350 - Deprecate Mini Lang

For memory, a description of the Mini Language guide can be found at the OFBiz wiki - Mini Language Reference

9.1. Groovy DSL (dynamic scripting library)

9.1.1. How to get Groovy support in your IDE

The following paragraph is for Eclipse users.

It is possible to get Groovy support in Eclipse by converting the loaded project to a Groovy Project. The project itself will work as before.

To do this just follow these few steps:

1. Right-click on the project that has to be converted
2. Click on "Configure"
3. Click on "Convert to Groovy Project"

Eclipse will automatically load the file OfbizDslDescriptorForEclipse.dsld , in which the known fields and methods used in Groovy Scripts are defined.

9.1.2. Known Fields

`property name: 'parameters'`
`type : 'java.util.Map'`
These are the parameters given to the Groovy Script, when it is called as a service. It is equivalent to `Map<String, Object>` context in the Java-Service-Definition.

`property name: 'context'`
`type: 'java.util.Map'`
More parameters, which are, for example, given through a screen or another Groovy Script. This is important when the script is called through an action segment of a screen.

`property name: 'delegator'`

`type: 'org.apache.ofbiz.entity.Delegator'`
Normal instance of the Delegator, which is used for special database access.

`property name: 'dispatcher'`
`type: 'org.apache.ofbiz.service.LocalDispatcher'`
Normal instance of the LocalDispatcher, which is used to call services and other service-like operations.

`property name: 'security'`
`type: 'org.apache.ofbiz.security.Security'`
Normal instance of the Security-Interface with which permission checks are done.

9.1.3. Known Methods

`method name: 'runService'`
`type: 'java.util.Map'`
`params: [serviceName: 'String', inputMap: 'java.util.Map']`
Helping method to call services instead of dispatcher.runSync(serviceName, inputMap). Also possible: run service: serviceName, with: inputMap

`method name: 'makeValue'`
`type: 'java.util.Map'`
`params: [entityName: 'String']`
Helping method to make a GenericValue instead of delegator.makeValue(entityName). Creates an empty GenericValue of the specific entity.

`method name: 'findOne'`
`type: 'java.util.Map'`
`params: [entityName: 'String', inputMap: 'java.util.Map']`
Helping method to find one GenericValue in the database. Used instead of delegator.findOne(entityName, inputMap)

`method name: 'findList'`
`type: 'java.util.List'`
`params: [entityName: 'String', inputMap: 'java.util.Map']`
Helping method to find many GenericValue in the database. Used instead of delegator.findList(entityName, inputMap, null, null, null, false)

`method name: 'select'`
`type: 'org.apache.ofbiz.entity.util.EntityQuery'`
`params: [entity: 'java.util.Set']`
Helping method used instead of EntityQuery.use(delegator).select(...)

`method name: 'select', type: 'org.apache.ofbiz.entity.util.EntityQuery', params: [entity: 'String…']`
As above.

`method name: 'from'`
`type: 'org.apache.ofbiz.entity.util.EntityQuery'`

`params: [entity: 'java.lang.Object']`
Helping method used instead of EntityQuery.use(delegator).from(…)

`method name: 'success'`
`type: 'def'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnSuccess(message)

`method name: 'failure'`
`type: 'java.util.Map'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnFailure(message)

`method name: 'error'`
`type: 'def'`
`params: [message: 'String']`
Helping method used instead of ServiceUtil.returnError(message)

`method name: 'logInfo'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logInfo(message, fileName)

`method name: 'logWarning'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logWarning(message, fileName)

`method name: 'logError'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logError(message, fileName)

`method name: 'logVerbose'`
`type: 'void'`
`params: [message: 'String']`
Helping method used instead of Debug.logVerbose(message, fileName)

The actual definition of the methods can be found in `` `/framework/service/src/main/java/org/apache/ofbiz/service/engine/GroovyBaseScript.groovy ``, the variables `dctx`, `dispatcher` and `delegator` are set in the file `GroovyEngine.java` which can be found in the same location.

9.2. Services

9.2.1. From MiniLang to Groovy

To see additional examples and finished conversions, which may help with occurring questions, click: OFBIZ-9350 - Deprecate Mini Lang There is a chance that a similar case has already been converted.

When a simple-method ends, it will automatically at least return a success-map.

All the Groovy Services have to return success at least, too.

```
return success()
```

9.2.2. Getting started

MiniLang files consist of services, which, in most cases, implement services.

The get converted to Groovy like the following:

```
<!-- This is MiniLang -->
<simple-method method-name="createProductCategory" short-description="Create an
ProductCategory">
   <!-- Code -->
</simple-method>
```

```
// This is the converted Groovy equivalent
/**
 * Create an ProductCategory
 */
def createProductCategory() {
    // Code
}
```

It will be useful for future developers, and everybody who has to check something in the code, to put at least the short-description as the new Groovydoc. This will hopefully more or less explain, what the method should or shouldn't do. If the short-description isn't helpful enough, feel free complete it.

The structure of if and else in MiniLang is a little different than the one from Groovy or Java and can be a bit confusing when first seen, so here is an example:

```
<if-empty field="parameters.productCategoryId">
    <sequenced-id sequence-name="ProductCategory" field=
"newEntity.productCategoryId"/>
<else>
    <set field="newEntity.productCategoryId" from-field=
"parameters.productCategoryId"/>
    <check-id field="newEntity.productCategoryId"/>
    <check-errors/>
</else>
</if-empty>
```

Notice, that the else always starts before the if-tag is closed, but sometimes isn't indented as one would expect it.

When navigating through bigger `if`-phrases, the navigation itself will be much easier through just clicking in the opening or closing `if`-tag; Eclipse will automatically mark the matching opening or closing `if`-tag for you.

There are two possibilities to initialize a field/variable in Groovy.

1. To define a field/variable with its correct typing
   `String fieldName = "value"``
2. To just "define" a field/variable. The IDE you are working with may not recognize the typing, but OFBiz can work with it:
   `def fieldName = "value"`

9.3. Checking Fields

Minilang	Groovy

`<if-empty field="fieldName"></if-empty>`	`//checks if fieldName is existent and/or empty`
	`if (!fieldName) {}`

Minilang	Groovy
`<if-empty field="fieldName.property"></if-empty>`	`// fieldName has to be existent, property doesn't need to` `// if known, that property does exist, the ? can be left out` `if (!fieldName?.property) {}` `// CAUTION: every query like this in Groovy evaluates to a Boolean type` `// everything that is empty or false will turn into false:` `// null, [], [:], "", false -> false` `if (UtilValidate.isEmpty(fieldName)) {}`
`<if>` `<condition>` `<or>` `<if-empty field="field1"/>` `<if-empty field="field2"/>` `</or>` `</condition>` `<then>` `<!-- code in if -->` `</then>` `<else>` `<!-- code in else -->` `</else>` `</if>`	`if (!field1 || !field2) {` `// code in if` `} else {` `// code in else` `}`
`<if-compare-field field="product.primaryProductCategoryId" to-field="parameters.productCategoryId" operator="equals">` `<!-- code -->` `</if-compare-field>`	`// this will even work, if product is not existent or null` `if (UtilValidate.areEqual(product?.primaryProductCategoryId, parameters.productCategoryId)) {` `// code` `}`
`<if-instance-of field="parameters.categories" class="java.util.List"></if-instance-of>`	`if (parameters.categories instanceof java.util.List) {}`

9.4. Setting Fields

Minilang	Groovy

`<set field="fieldName" value="value"/>`	` // if fieldName is not initialized`
	`String fieldName = "value"`
	` // if fieldName is initialized`
	`fieldName = "value"`

`<set field="otherFieldName.property"`
`value="value"/>`
`<set`
`field="otherFieldName.otherProperty"`
`value="true" type="Boolean"/>`
`<set`
`field="otherFieldName.otherProperty"`
`from-field="parameters.property/>`
	` // if otherFieldName is not yet`
	`initialized, you have to do it first`
	` // MiniLang does that automatically`
	`Map otherFieldName = [:] // empty Map`
	` // now put the values in`
	`otherFieldName = [`
	`    property: "value",`
	`    otherProperty: true`
	`]`
	` // or the less efficient way`
	`otherFieldName.property = "value"`
	`otherFieldName.otherProperty = true`
	
	` // it is possible to put different`
	`values in later:`
	`otherFieldName.property = parameters`
	`.property`

`<set field="thisFieldName"`
`value="${groovy: []}" type="List"/>`
	` // this is easier in Groovy`
	`List thisFieldName = []`

`<property-to-field`
`resource="CommonUiLabels"`
`property="CommonGenericPermissionError"`
`field="failMessage"/>`
`<!-- there are different cases of this,`
`which are not distinguished in MiniLang`
`-->`
`<property-to-field`
`resource="general.properties"`
`property="currency.uom.id.default"`
`field="parameters.rateCurrencyUomId"/>`
	`String failMessage = UtilProperties`
	`.getMessage("CommonUiLabels",`
	`"CommonGenericPermissionError",`
	`parameters.locale)`
	` // in Groovy there can be a difference`
	`for the second case`
	`parameters.rateCurrencyUomId =`
	`UtilProperties.getPropertyValue('general`
	`.properties', 'currency.uom.id.default')`

Minilang	Groovy
`<clear-field field="product.primaryProductCategoryId" />`	`product.primaryProductCategoryId = null`

9.5. Starting Services

Minilang	Groovy
`<set field="relatedCategoryContext.parentProductCategoryId"  from-field="defaultTopCategoryId"/>` `<call-service service-name="getRelatedCategories" in-map-name="relatedCategoryContext">` `    <result-to-field result-name="categories" field="resCategories"/>` `</call-service>`	`def relatedCategoryContext = [parentProductCategoryId: defaultTopCategoryId]` `def serviceResult = run service: "getRelatedCategoryies", with: relatedCategoryContext` `def resCategories = serviceResult .categories` ` // if it is not too confusing to read you can leave out the extra variable` `run service: "getRelatedCategoryies", with: [parentProductCategoryId: defaultTopCategoryId]`
`<set-service-fields service-name="productCategoryGenericPermission" map="parameters" to-map="productCategoryGenericPermissionMap"/>` `<call-service service-name="productCategoryGenericPermission" in-map-name="productCategoryGenericPermissionMap">` `    <results-to-map map-name="genericResult"/>` `</call-service>`	` // instead of setting the service fields from parameters, it is possible to run the service with the parameters map` `Map genericResult = run service: "productCategoryGenericPermission", with: parameters`

9.6. Preparing Service Results

Minilang	Groovy

`<field-to-result field="fieldBudgetId" result-name="budgetId"/>`	` // MiniLang knows this implicitly`
	`def result = success()`
	`result.budgetId = fieldBudgetId`
	`return result`

9.7. Database Communication

Minilang	Groovy

`<make-value entity-name="FinAccountTrans" value-field="newEntity"/>`
`<set-nonpk-fields map="parameters" value-field="newEntity"/>`
`<set-pk-fields map="parameters" value-field="newEntity"/>`
	` // this is the easy way`
	`GenericValue newEntity = makeValue("FinAccountTrans", parameters)`
	` // this is also possible`
	`GenericValue newEntity = makeValue("FinAccountTrans")`
	`newEntity.setPKFields(parameters)`
	`newEntity.setNonPKFields(parameters)`

`<entity-and entity-name="BudgetStatus" list="budgetStatuses">`
`    <field-map field-name="budgetId" from-field="parameters.budgetId"/>`
`    <order-by field-name="-statusDate"/>`
`</entity-and>`
	` // this can also be done in one line, but it can easily become unreadable`
	`def budgetStatuses = from("BudgetStatus")`
	`    .where("budgetId", paramters.budgetId)`
	`    .orderBy("-statusDate")`
	`    .queryList()`

Minilang	Groovy

`<entity-one entity-name="StatusValidChange" value-field="statusValidChange">`
`    <field-map field-name="statusId" from-field="budgetStatus.statusId"/>`
`    <field-map field-name="statusIdTo" from-field="parameters.statusId"/>`
`</entity-one>`
`<!-- entity-one can be called without child elements, too -->`
`<entity-one entity-name="Product" value-field="product" auto-field-map="true"/>`
	` // MiniLang has false set for useCache as the default value`
	`statusValidChange = findOne("StatusValidChange", [statusId: budgetStatus.statusId, statusIdTo: parameters.statusId], false)`
	` // this is also possible`
	`statusValidChange = from("StatusValidChange")`
	`    .where("statusId", budgetStatus.statusId, "statusIdTo", parameters.statusId)`
	`    .queryOne()`
	` // if there are no child elements, this can be used`
	`GenericValue product = from("Product").where(parameters).queryOne()`

`<find-by-primary-key entity-name="ProductCategoryMember" map="lookupPKMap" value-field="lookedUpValue"/>`	`GenericValue lookedUpValue = findOne("ProductCategoryMember", lookupPKMap, false)`
	` // this is also possible`
	`lookedUpValue = from("ProductCategoryRole")`
	`    .where(lookupPKMap)`
	`    .queryOne()`

Minilang

```xml
<entity-condition entity-name="ProductCategoryContentAndInfo" list="productCategoryContentAndInfoList" filter-by-date="true" use-cache="true">
    <condition-list combine="and">
        <condition-expr field-name="productCategoryId" from-field="productCategoryList.productCategoryId"/>
        <condition-expr field-name="prodCatContentTypeId" value="ALTERNATIVE_URL"/>
    </condition-list>
    <order-by field-name="-fromDate"/>
</entity-condition>
<!-- entity-condition can also be used with the "or" operator -->
<entity-condition entity-name="ProdCatalogCategory" list="prodCatalogCategoryList" filter-by-date="true">
    <condition-list combine="and">
        <condition-expr field-name="productCategoryId" from-field="parameters.productCategoryId"/>
        <condition-list combine="or">
            <condition-expr field-name="prodCatalogCategoryTypeId" value="PCCT_VIEW_ALLW"/>
            <condition-expr field-name="prodCatalogCategoryTypeId" value="PCCT_PURCH_ALLW"/>
        </condition-list>
    </condition-list>
</entity-condition>
```

Groovy

```groovy
 // the Groovy methods use the "and" and "equals" operator as default values
List productCategoryContentAndInfoList = from("ProductCategoryContentAndInfo")
    .where("productCategoryId", productCategoryList.productCategoryId, "prodCatContentTypeId", "ALTERNATIVE_URL")
    .cache().orderBy("-fromDate")
    .filterByDate()
    .queryList()
 // with the use of the "or" operator you have to build your condition like this
EntityCondition condition = EntityCondition.makeCondition([
    EntityCondition.makeCondition([
        EntityCondition.makeCondition("prodCatalogCategoryTypeId", "PCCT_VIEW_ALLW"),
        EntityCondition.makeCondition("prodCatalogCategoryTypeId", "PCCT_PURCH_ALLW")
    ], EntityOperator.OR),
    EntityCondition.makeCondition("productCategoryId", parameters.productCategoryId)
])
List prodCatalogCategoryList = from("ProdCatalogCategory").where(condition).filterByDate().queryList()
```

Minilang	Groovy
`<make-value entity-name="FinAccountTrans" value-field="newEntity"/>` `<set-nonpk-fields map="parameters" value-field="newEntity"/>` `<!-- In this case multiple fields of the GenericValue are set -->` `<make-value entity-name="ProductCategoryRollup" value-field="newLimitRollup"/>` `<set field="newLimitRollup.productCategoryId" from-field= "newEntity.productCategoryId"/>` `<set field="newLimitRollup.parentProductCategoryId" from-field="productCategoryRole.productCategoryId"/>` `<set field="newLimitRollup.fromDate" from-field="nowTimestamp"/>`	`def newEntity = makeValue("FinAccountTrans", parameters)` ` // you can set multiple fields of a GenericValue like this` `def newLimitRollup = makeValue("ProductCategoryRollup", [` `    productCategoryId: newEntity.productCategoryId,` `    parentProductCategoryId: productCategoryRole.productCategoryId,` `    fromDate: nowTimestamp` `])`
`<set field="statusValidChange.prop" value="value"/>`	`statusValidChange.prop = "value"`
`<create-value value-field="newEntity"/>`	`newEntity.create()`
`<store-value value-field="newEntity"/>` `<store-list list="listToStore"/>`	`newEntity.store()` `delegator.storeAll(listToStore)`
`<clone-value value-field="productCategoryMember" new-value-field="newProductCategoryMember"/>`	`def newProductCategoryMember = productCategoryMember.clone()`
`<remove-value value-field="lookedUpValue"/>`	`lookedUpValue.remove()`

Minilang	Groovy
`<sequenced-id sequence-name="ProductCategory" field="newEntity.productCategoryId"/>`	`newEntity.productCategoryId = delegator .getNextSeqId("ProductCategory")`
`<check-id field="newEntity.productCategoryId"/>`	`UtilValidate.checkValidDatabaseId(newEntity.productCategoryId)`
`<make-next-seq-id value-field="newEntity" seq-field-name="linkSeqId"/>`	`delegator.setNextSubSeqId(newEntity, "linkSeqId", 5, 1)  // the numbers 5 and 1 are used in the Java implementation of the MiniLang method  // and can also be found as the default values in the MiniLang documentation`

9.8. Permissions

To also check for admin-permissions, this method has to be used: `hasEntityPermission(permission, action, userLogin)`

If the method is used with wildcards, it is important to not forget the underscore, which comes before the parameter action!

Minilang	Groovy
`<check-permission permission="CATALOG" action="_CREATE">     <alt-permission permission="CATALOG_ROLE" action="_CREATE"/>     <fail-property resource="ProductUiLabels" property="ProductCatalogCreatePermissionError"/> </check-permission> <check-errors/>`	`if (!(security.hasEntityPermission("CATALOG", "_CREATE", parameters.userLogin)     || security.hasEntityPermission("CATALOG_ROLE", "_CREATE", parameters.userLogin))) {     return error(UtilProperties.getMessage("ProductUiLabels", "ProductCatalogCreatePermissionError", parameters.locale)) }`

Minilang	Groovy

`<set field="hasCreatePermission" value="false" type="Boolean"/>`
`<if-has-permission permission="${primaryPermission}" action="${mainAction}">`
`    <set field="hasCreatePermission" value="true" type="Boolean"/>`
`</if-has-permission>`
	`// this will automatically be set to false if the user doesn't have the permission`
	`def hasCreatePermission = security.hasEntityPermission(primaryPermission, "_${mainAction}", parameters.userLogin)`

9.9. Timestamp And System Time

The first two simple-method are deprecated; the third method should have been used instead.

Minilang	Groovy
`<now-timestamp field="nowTimestamp"/>`	`Timestamp nowTimestamp = UtilDateTime.nowTimestamp()`
`<now-date-to-env field="nowDate"/>`	`Timestamp nowDate = UtilDateTime.nowTimestamp()`

`<!-- this method also has the parameter "type", which is set to 'java.sql.timestamp' as default -->`
`<now field="fooNow"/>`
	`Timestamp fooNow = UtilDateTime.nowTimestamp()`

`<if-compare-field field="productCategoryMember.thruDate" to-field="expireTimestamp" operator="less" type="Timestamp">`
`    <!-- code -->`
`</if-compare-field>`
	`Timestamp thruDate = productCategoryMember.thruDate`
	`if (thruDate && thruDate.before(expireTimestamp)) {`
	`    // code`
	`}`

9.10. Logging

Since all of the log methods are know to the Groovy Language, it is possible to just nearly use them as they are in MiniLang.
For further explanation, here are some examples:

Minilang	Groovy
`<log level="verbose" message="Permission check failed, user does not have permission"/>`	`logVerbose("Permission check failed, user does not have the correct permission.")`
`<log level="info" message="Applying feature [${productFeatureId}] of type [${productFeatureTypeId}] to product [${productId}]"/>`	`logInfo("Applying feature [${productFeatureId}] of type [${productFeatureTypeId}] to product [${productId}]")`

9.11. General

Minilang	Groovy

`<call-simple-method method-name="checkCategoryRelatedPermission"/>`
`<check-errors/>`
	`// simple-methods inside of classes, as long as they are not services, will be called like normal methods`
	`Map res = checkCategoryRelatedPermission("updateProductCategory", "UPDATE", null, null)`
	`if (!ServiceUtil.isSuccess(res)) {`
	`    return res`
	`}`

`<iterate list="subCategories" entry="subCategory">`
`    <!-- code -->`
`</iterate>`
	`for (def subCategory : subCategories) {`
	`    // code`
	`}`
	`subCategories.each { subCategory ->`
	`    // code`
	`}`

Minilang	Groovy

`<iterate-map`
`map="parameters.productFeatureIdByType"`
`key="productFeatureTypeId"`
`value="productFeatureId">`
`    <!-- in here something should happen`
`with value and key -->`
`</iterate-map>`
	`for (Map entry : parameters`
	`.productFeatureIdByType.entrySet()) {`
	`    def productFeatureTypeId = entry`
	`.getKey()`
	`    def productFeatureId = entry`
	`.getValue()`
	`    // in here something should happen`
	`with value and key`
	`}`

`<if>`
`    <condition>`
`        <not>`
`            <or>`
`                <if-has-permission`
`permission="CATALOG"`
`action="_${checkAction}"/>`
`                <and>`
`                    <if-has-permission`
`permission="CATALOG_ROLE"`
`action="_${checkAction}"/>`
`                    <not><if-empty`
`field="roleCategories"/></not>`
`                </and>`
`            </or>`
`        </not>`
`    </condition>`
`    <then>`
`        <!-- code -->`
`    </then>`
`</if>`
	`if (!security.hasEntityPermission`
	`("CATALOG", "_${checkAction}",`
	`parameters.userLogin)`
	`    && !(security.hasEntityPermission`
	`("CATALOG_ROLE", "_${checkAction}",`
	`parameters.userLogin)`
	`    && roleCategories)) {`
	`    // code`
	`}`

`<set field="validDate" from-`
`field="parameters.validDate"/>`
`<if-not-empty field="validDate">`
`    <filter-list-by-date`
`list="productCategoryMembers" valid-`
`date="validDate"/>`
`</if-not-empty>`
	`def query = from(`
	`"ProductCategoryMember").where("productC`
	`ategoryId", parameters.`
	`productCategoryId)`
	`if (parameters.validDate) {`
	`    query.filterByDate()`
	`}`
	`List productCategoryMembers = query`
	`.queryList()`

Minilang	Groovy

`<order-map-list list="productsList">`
`    <order-by field-name="sequenceNum"/>`
`</order-map-list>`
	`productsList = EntityUtil.orderBy`
	`(productsList, ["sequenceNum"])`

9.12. Where to find MiniLang implementation

If you find yourself in a position, where you don't know how to convert a certain tag from MiniLang to Groovy, you can always check the Java implementation of the MiniLang method.
All of the methods have an existing Java implementation and you can find all of them in this folder: `/ofbiz/trunk/framework/minilang/src/main/java/org/apache/ofbiz/minilang/method`

The interesting part of this implementation is the method `exec()`, which actually runs the MiniLang tag.
The tag `<remove-by-and>` for example is realized using this part of code here:

```
@Override

public boolean exec(MethodContext methodContext) throws MiniLangException {
    @Deprecated
    String entityName = entityNameFse.expandString(methodContext.getEnvMap());
    if (entityName.isEmpty()) {
        throw new MiniLangRuntimeException("Entity name not found.", this);
    }
    try {
        Delegator delegator = getDelegator(methodContext);
        delegator.removeByAnd(entityName, mapFma.get(methodContext.getEnvMap()));
    } catch (GenericEntityException e) {
        String errMsg = "Exception thrown while removing entities: " + e.getMessage();
        Debug.logWarning(e, errMsg, module);
        simpleMethod.addErrorMessage(methodContext, errMsg);
        return false;
    }
    return true;
}
```

In this you can find one important part of code, which is:

```
delegator.removeByAnd(entityName, mapFma.get(methodContext.getEnvMap()));
```

This tells you, that, if you're trying to convert the tag `<remove-by-and>`, you can use `delegator.removeByAnd()` in Groovy.

10. Testing

10.1. Unit Test (Using JUnit)

10.1.1. How to define a unit test?

1. Define test-suite in ofbiz-component.xml like this:

```
    <test-suite loader="main" location="testdef/servicetests.xml"/>
```

2. Create test-case in test-suite file.

```
    <test-suite suite-name="servicetests"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/test-
suite.xsd">
        ....
        ....
        <test-case case-name="test-case-name">
        ...
        </test-case>
        ....
        ....
    </test-suite>
```

10.1.2. How to create a test case?

Define a test-case in an XML test-suite file like this:

```
    <test-case case-name="test-case-name">
        ...
    </test-case>
```

The test-case tag contains details of each test type:

Entity XML

Specific entity xml and its action would like to be tested in entity-xml-url attribute and action attribute respectively like this:

```
    <test-case case-name="service-dead-lock-retry-assert-data">
        <entity-xml action="assert" entity-xml-
url="component://service/testdef/data/ServiceDeadLockRetryAssertData.xml"/>
    </test-case>
```

JUnit

Specific class's name which will be tested, in a class-name attribute like this:

```
    <test-case case-name="service-tests">
        <junit-test-suite class-
name="org.apache.ofbiz.service.test.ServiceEngineTests"/>
    </test-case>
```

Service

Specific service's name which will be tested in a service-name attribute like this:

```
    <test-case case-name="service-lock-wait-timeout-retry-test">
        <service-test service-name="testServiceLockWaitTimeoutRetry"/>
    </test-case>
```

Simple Method

Specific simple method's location and name which will be tested in a location and a name attribute respectively like this:

```
    <test-case case-name="auto-accounting-transaction-tests-PoReceipt">
        <simple-method-test
location="component://accounting/minilang/test/AutoAcctgTransTests.xml"
name="testAcctgTransOnPoReceipts"/>
    </test-case>
```

10.1.3. How to run a unit test?

You can run unit test by run 'gradle' with following target:

Execute all unit tests

`gradlew test`

Execute all integration tests

`gradlew testIntegration`

OR

`gradlew 'ofbiz --test'`

Execute integration tests with a different log level

It is possible to start integration tests with a log level different from the default one. The log levels allowed are listed below from most verbose to least verbose:

- always
- verbose
- timing
- info
- important
- warning
- error
- fatal

```
gradlew "ofbiz --test loglevel=fatal"
```

Execute an integration test case

run a test case, in this example the component is "entity" and the case name is "entity-tests"

```
gradlew "ofbiz --test component=entity --test suitename=entitytests --test case=entity-query-tests"
```

Execute an integration test case in debug mode with verbose log

listens on port **5005**

```
gradlew "ofbiz --test component=entity --test loglevel=verbose" --debug-jvm
```

Execute an integration test suite

```
gradlew "ofbiz --test component=entity --test suitename=entitytests"
```

Execute an integration test suite in debug mode

listens on port **5005**

```
gradlew "ofbiz --test component=entity --test suitename=entitytests" --debug-jvm
```

10.1.4. Possible error messages

Some error messages in the log are not important and often caused by the test.

Roll back

Roll back error message occurred when transaction roll back data. This error message will be in between starting and finished test cass line like this:

```
    [java] 2009-12-22 16:05:28,349 (main) [   TestRunContainer.java:238:INFO ] [JUNIT]
: [test case's name] starting...
    [java] 2009-12-22 16:05:28,355 (main) [    TransactionUtil.java:336:ERROR]
    ....
    ....
    ....
    [java] ---- exception report
----------------------------------------------------------
    [java] [TransactionUtil.rollback]
    [java] Exception: java.lang.Exception
    [java] Message: Stack Trace
    [java] ---- stack trace
---------------------------------------------------------------
    [java] java.lang.Exception: Stack Trace
    [java]
org.apache.ofbiz.entity.transaction.TransactionUtil.rollback(TransactionUtil.java:335)
    [java]
org.apache.ofbiz.entity.transaction.TransactionUtil.rollback(TransactionUtil.java:317)
    [java]
org.apache.ofbiz.entity.test.EntityTestSuite.testTransactionUtilRollback(EntityTestSui
te.java:437)
    [java] sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    [java]
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
    [java]
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
    [java] java.lang.reflect.Method.invoke(Method.java:597)
    [java] junit.framework.TestCase.runTest(TestCase.java:154)
    [java] junit.framework.TestCase.runBare(TestCase.java:127)
    [java] junit.framework.TestResult$1.protect(TestResult.java:106)
    [java] junit.framework.TestResult.runProtected(TestResult.java:124)
    [java] junit.framework.TestResult.run(TestResult.java:109)
    [java] junit.framework.TestCase.run(TestCase.java:118)
    [java] junit.framework.TestSuite.runTest(TestSuite.java:208)
    [java] junit.framework.TestSuite.run(TestSuite.java:203)
    [java] junit.framework.TestSuite.runTest(TestSuite.java:208)
    [java] junit.framework.TestSuite.run(TestSuite.java:203)
    [java]
org.apache.ofbiz.testtools.TestRunContainer.start(TestRunContainer.java:146)
    [java]
org.apache.ofbiz.base.container.ContainerLoader.start(ContainerLoader.java:100)
    [java] org.apache.ofbiz.base.start.Start.startStartLoaders(Start.java:272)
    [java] org.apache.ofbiz.base.start.Start.startServer(Start.java:322)
    [java] org.apache.ofbiz.base.start.Start.start(Start.java:326)
    [java] org.apache.ofbiz.base.start.Start.main(Start.java:411)
    [java]
--------------------------------------------------------------------------------
    [java]
    ....
    ....
    ....
    [java] 2009-12-22 16:05:28,366 (main) [    TransactionUtil.java:346:INFO ]
[TransactionUtil.rollback] transaction rolled back
    [java] 2009-12-22 16:05:28,370 (main) [   TestRunContainer.java:234:INFO ] [JUNIT]
: [test case's name] finished.
```

10.1.5. Test result

After you run unit test, you can see the result of the testing.
If you use run-tests target, you can see test result web page by view runtime/logs/test-results/html/index.html file in web browser and see JUnit Test Result files for each test suite in runtime/logs/test-results directory.
If you use other target you only see JUnit Test Result file in runtime/logs/test-results.

10.2. Integration Tests

11. Deployment

For a core configuration guide check the OFBiz configuration Guide (some points are not up to date).

11.1. email

11.1.1. HowTo use email in OFBiz

Receiving Email.

OFBiz can receive email for multiple email addresses and via an MCA can create Communication events for the involved parties of the email.
Email attachments, via again the MCA are stored in the content component and can be accessed via the content Id.
Examples of an MCA can be found in the Party and Content Component.

To receive email a single POP/IMAP mailbox is polled at regular intervals.
This is configured in the ${ofbiz install dir}/framework/service/ofbiz-component.xml file in the commented section JavaMail Listener Container.

Any email address you want to be handled by OFBiz need to be forwarded to this single mailbox by an external mail server.
OFBiz then will try to match the email addresses to existing parties and will create a single communication event referring to the found parties.

If an incoming email address cannot be matched against a party, the communication event will get a special status and the receiving party can either delete the communication event or can ask the system to automatically create a party from the incoming email address.

By default the configuaration file has the email poller commented out.
The parameters to this function are pretty self explanatory.

Sending Email in OFBiz

Various parts in the OFBiz application are sending out email for various reasons.

Sending out email is controlled in the ${ofbiz install dir}/framework/common/config/general.properties file with the following parameters:

- SMTP Server (relay host): mail.smtp.relay.host
- SMTP Username (if needed): mail.smtp.auth.user
- SMTP Password (if needed): mail.smtp.auth.password
- Turn on email notifications by setting the mail.notifications.enabled property to "Y".

12. Security

12.1. OFBiz in production

In matter of security, to be sure to be up to date, the first place to look at is https://ofbiz.apache.org/security.html

For more details you may be also interested by https://issues.apache.org/jira/browse/OFBIZ-1525

If you look for how to handle access permissions, this page should help you: https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Security+Permissions

Last but not least, you will certainly find useful, the security section of The Apache OFBiz Technical Production Setup Guide

12.1.1. Gradle Update

OFBiz uses Gradle for many things, including building and running OFBiz.

Out Of The Box (OOTB) you get versions of third parties libraries which might need to be updated from time to time. For that you may take as an example to follow https://issues.apache.org/jira/browse/OFBIZ-10213

12.2. Passwords and JWT (JSON Web Tokens) usage

12.2.1. How are set and used passwords and JWT in Apache OFBiz

The Apache OFBiz Project Release trunk

Passwords

Demo and seed passwords are stored in files loaded through security ofbiz-component.xml. To know more about that be sure to read:

- The technical production setup guide notably "Initial Data Loading" and "Security Settings" sections
- How to secure your deployment

> These configuration steps are not to be neglected for the security of a **production environment**

JWT usage

As says Wikipedia:

> JSON Web Token (JWT) is an Internet standard for creating JSON-based access tokens that assert some number of claims.

We currently use JWT in 2 places:

1. To let users safely recreate passwords (in backend and frontend)
2. To allow SSO (Single Sign-on) jumpings from an OFBiz instance to another on another domain, by also using CORS (Cross-origin resource sharing) on the target server

How to secure JWT

When you use JWT, in order to sign your tokens, you have the choice of using a sole so called secret key or a pair of public/private keys: https://jwt.io/introduction/.

You might prefer to use pair of public/private keys, for now by default OFBiz uses a simple secret key. Remains the way how to store this secret key. This is an interesting introduction about this question.

1. The first idea which comes to mind is to use a property in the security.properties file. It's safe as long as your file system is not compromised.
2. You may also pick a SystemProperty entity (overrides the file property). It's safe as long as your DB is not compromised.
3. We recommend to not use an environment variable as those can be considered weak:
   - http://movingfast.io/articles/environment-variables-considered-harmful
   - https://security.stackexchange.com/questions/49725/is-it-really-secure-to-store-api-keys-in-environment-variables
4. You may want to tie the encryption key to the logged in user. This is used by the password recreation feature. The JWT secret key is salted with a combination of the current logged in user and her/his password. This is a simple and effective safe way.
5. Use a JTI (JWT ID). A JTI prevents a JWT from being replayed. This auth0 blog article get deeper in that. The same is kinda achieved with the password recreation feature. When the user log in after the new password creation, the password has already been changed. So the link (in the sent email) containing the JWT for the creation of the new password can't be reused.
6. Tie the encryption key to the hardware. You can refer to this Wikipedia page for more information.
7. If you want to get deeper in this get to this OWASP documentation

Note: if you want to use a pair of public/private keys you might want to consider leveraging the Java Key Store that is also used by the "catalina" component to store certificates. Then don't miss to read:

- https://cryptosense.com/blog/mighty-aphrodite-dark-secrets-of-the-java-keystore/
- https://neilmadden.blog/2017/11/17/java-keystores-the-gory-details/

Also remember that like everything a JWT can be attacked and, though not used or tried in OFBiz yet, a good way is to mitigate an attack by using a KeyProvider. I have created OFBIZ-11187 for that.

Properties

The *security.properties* file contains five related properties:

```
# -- If false, then no externalLoginKey parameters will be added to cross-webapp urls
security.login.externalLoginKey.enabled=true
```

```
# -- Security key used to encrypt and decrypt the autogenerated password in forgot
password functionality.
#    Read Passwords and JWT (JSON Web Tokens) usage documentation to choose the way
you want to store this key
login.secret_key_string=login.secret_key_string
```

```
# -- Time To Live of the token send to the external server in seconds
security.jwt.token.expireTime=1800
```

```
# -- Enables the internal Single Sign On feature which allows a token based login
between OFBiz instances
# -- To make this work you also have to configure a secret key with security.token.key
security.internal.sso.enabled=false
```

```
# -- The secret key for the JWT token signature. Read Passwords and JWT (JSON Web
Tokens) usage documentation to choose the way you want to store this key
security.token.key=security.token.key
```

There are also SSO related SystemProperties in *SSOJWTDemoData.xml*:

```
    <SystemProperty systemResourceId="security"
systemPropertyId="security.internal.sso.enabled" systemPropertyValue="false"/>
    <SystemProperty systemResourceId="security" systemPropertyId="security.token.key"
systemPropertyValue="security.token.key"/>
    <SystemProperty systemResourceId="security"
systemPropertyId="SameSiteCookieAttribute" systemPropertyValue="strict"/>
```

Internal SSO

The introduction of the same-site attribute set to 'strict' for all cookies prevents the internal Single Sign On feature. Why is clearly explained here.

So same-site attribute set to 'none' is necessary for the internal SSO to work, 'lax' is not enough. So if someone wants to use the internal SSO feature s/he also needs to use the CSRF token defense. If s/he wants to be safe from CSRF attacks. Unfortunately, due backporting difficulties, this option is currently (2020-04-15) only available in trunk.

Fecth API

An alternative would be to use the Fetch Javascript API with the

```
credentials: "include"
```

option to enable CORS. Here is an example

For those interested, there are more information in https://issues.apache.org/jira/browse/OFBIZ-11594

Last but not least

Be sure to read Keeping OFBiz secure

12.3. CSRF defense

12.3.1. How is done the CSRF defense in Apache OFBiz and how to adapt it if needed

The Apache OFBiz Project Release trunk

The same-Site attribute

> The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
>
> — According to OWASP ZAP

By default OOTB the SameSiteFilter property sets the same-site attribute value to 'strict. SameSiteFilter allows to change to 'lax' if needed. If you use 'lax' we recommend that you set the csrf.defense.strategy property to org.apache.ofbiz.security.CsrfDefenseStrategy in order to provide an effective defense against CSRF attacks.

Properties

The *security.properties* file contains related properties:

```
# -- By default the SameSite value in SameSiteFilter is 'strict'.
# -- This property allows to change to 'lax' if needed.
# -- If you use 'lax' we recommend that you set
# -- org.apache.ofbiz.security.CsrfDefenseStrategy
# -- for csrf.defense.strategy (see below)
SameSiteCookieAttribute=
```

```
# -- The cache size for the Tokens Maps that stores the CSRF tokens.
# -- RemoveEldestEntry is used when it's get above csrf.cache.size
# -- Default is 5000
# -- TODO: possibly separate tokenMap size from partyTokenMap size
csrf.cache.size=
# -- Parameter name for CSRF token. Default is "csrf" if not specified
csrf.tokenName.nonAjax=
```

```
# -- The csrf.entity.request.limit is used to show how to avoid cluttering the Tokens
Maps cache with URIs starting with "entity/"
# -- It can be useful with large Database contents, ie with a large numbers of tuples,
like "entity/edit/Agreement/10000, etc.
# -- The same principle can be extended to other cases similar to "entity/" URIs
(harcoded or using similar properties).
# -- Default is 3
csrf.entity.request.limit=
```

```
# -- CSRF defense strategy.
# -- Because OFBiz OOTB also sets the SameSite attribute to 'strict' for all cookies,
# -- which is an effective CSRF defense,
# -- default is org.apache.ofbiz.security.NoCsrfDefenseStrategy if not specified.
# -- Use org.apache.ofbiz.security.CsrfDefenseStrategy
# -- if you need to use a 'lax' for SameSiteCookieAttribute
csrf.defense.strategy=
```

There is also a SystemProperty in *SSOJWTDemoData.xml*:

```
<SystemProperty systemResourceId="security" systemPropertyId="SameSiteCookieAttribute"
systemPropertyValue="strict"/>
```

12.4. Impersonation

12.4.1. What is Impersonation in Apache OFBiz

The Apache OFBiz Project Release trunk

Introduction to User impersonation

User Impersonation is a feature that offer a way to select a user login and impersonate it, i.e. see what the user could see navigating through the application in his name.

How do this work ?

An authorized user *(see security and controls section for configuration)*, can select a user that will be impersonated.

The impersonation start, if everything is well configured, in current application (partymgr for the demo). Everything appears like if we were logged in with the userLoginId and the valid password (though we know nothing about it)

The only thing showing that we currently are impersonating a user is the little bottom-right image :

This icon indicates, when clicking on it, the user impersonated, and offer a way to depersonate.

The impersonate period is stored for audit purpose, and if the impersonator forgot to depersonate, the period is terminated *one hour* after impersonation start.

Security

This feature can draw some concerns about security aspect. This paragraph will introduce every controls and properties that have been implemented around the impersonation feature.

These configuration steps are not to be neglected for a **production environment** since this feature offer a way to act in place of another user.

Properties

The *security.properties* file introduce two properties that control impersonation feature :

```
security.disable.impersonation = true
```

This property, set by default to **true**, controls the activation of impersonation feature. If no configuration is done any user trying to use impersonation will face an error message, indicating that the feature is disabled.

To enable impersonation this property need to be set to **false**

```
security.login.authorised.during.impersonate = false
```

This property controls the way impersonation occurred to the impersonated user :

In default configuration, the impersonated user see nothing and can use the application without knowing that he is currently impersonated. Several authorized user can impersonate a same login without any issue.

This configuration is intended for testing/QA environment allowing any authorized user to impersonate a login to validate its configuration, test the application etc.

Set to **true**, this configuration improve the control of the data generated by the impersonated user. Indeed, Only one authorized user can impersonate a login at the same time, and during the impersonation process, the impersonated user is unable to act within the application.

Since the impersonation period is stored in database, the actions done by the authorized user can be identified if there is the need to do so.

This configuration is intended for production environment

Controls

The permission

First, to be able to use impersonation, a user need to possess *IMPERSONATE_ADMIN* permissions. Demo data offer *IMPERSONATION* security group for this purpose.
In demo data, *FULLADMIN* security group also possess the permission.

Permission based user restriction

An authorized user cannot impersonate any user. There are two main controls that will restrict the impersonation feature.

Cannot impersonate Admin user

It is impossible to impersonate a user that is granted any of the admin permission :

```
"IMPERSONATE_ADMIN"
"ARTIFACT_INFO_VIEW"
"SERVICE_MAINT"
"ENTITY_MAINT"
"UTIL_CACHE_VIEW"
"UTIL_DEBUG_VIEW"
```

Cannot impersonate more privileged user

It is impossible to impersonate a user that has more permission than your user. Even if the missing persmission is a minor one.

12.5. Gradle Dependency Verification

The Apache OFBiz Project Release trunk

This feature is for now disabled. You may use it locally if you want…

Here is the Gradle documentation about dependency verification

As it’s a long read you might prefer this summary:

the dependency verification is an incubating feature. So we will wait before backporting from trunk…

By default OFBiz comes with OOTB Gradle dependency verification.

This means that it embeds a verification-metadata.xml file and a verification-keyring.gpg in OFBiz gradle sub-directory which is used during builds and other tasks to verify dependencies.

These files are initially created using :

gradlew --write-verification-metadata pgp,sha256 help
gradlew --write-verification-metadata pgp,sha256 --export-keys

These command creates or updates the verification-metadata.xml and verification-keyring.gpg files which respectively contains the checksums for each of declared dependencies and the related keys

Currently the status is it’s incomplete in OFBiz. You get this message:

- Some artifacts aren’t signed or the signature couldn’t be retrieved.
- Some signature verification failed. Checksums were generated for those artifacts but you MUST check if there’s an actual problem. Look for entries with the following comment: PGP verification failed PGP verification failed

Only 6 keys are concerned. This does not prevent the verification to work using metadata, though it’s better to check the situation in case of doubts (OK OTTB). You may use

gradlew build --refresh-keys

To recreate the keys

The verification-metadata.xml file contains 2 entries that can be set to true or false to check or ignore the 2 functionalities:

<verify-metadata>true</verify-metadata>
<verify-signatures>true</verify-signatures>

Finally, you may refer to https://issues.apache.org/jira/browse/OFBIZ-12186 for more information.

13. Appendices

14. Apache OFBiz Plugins for Developer

The Apache OFBiz Project

14.1. Deployment

14.1.1. OFBiz Single Sign On using CAS and LDAP

Setup the Java Key Store

From the directory in which you want to create the keystore, run keytool with the following parameters.

1. Generate the server certificate.

   ```
       $ keytool -genkey -alias tomcat -keyalg RSA -keypass changeit -storepass
   changeit -keystore keystore.jks
   ```

   When you press Enter, keytool prompts you to enter the server name, organizational unit, organization, locality, state, and country code.

   Note that you must enter the server name in response to keytool’s first prompt, in which it asks for first and last names.

   For testing purposes, this can be localhost.

2. Export the generated server certificate in keystore.jks into the file server.cer.

   ```
       $ keytool -export -alias tomcat -storepass changeit -file server.cer -keystore
   keystore.jks
   ```

3. To create the trust-store file cacerts.jks and add the server certificate to the trust-store, run keytool from the directory where you created the keystore and server certificate. Use the following parameters:

   ```
       $ keytool -import -v -trustcacerts -alias tomcat -file server.cer -keystore
   cacerts.jks -keypass changeit -storepass changeit
   ```

4. Information on the certificate, such as that shown next, will display.

```
    $ keytool -import -v -trustcacerts -alias tomcat -file server.cer -keystore
cacerts.jks -keypass changeit -storepass changeit

    Owner: CN=localhost, OU=Sun Micro, O=Docs, L=Santa Clara, ST=CA, C=US
    Issuer: CN=localhost, OU=Sun Micro, O=Docs, L=Santa Clara, ST=CA, C=US
    Serial number: 3e932169
    Valid from: Tue Apr 08
    Certificate fingerprints:
    MD5: 52:9F:49:68:ED:78:6F:39:87:F3:98:B3:6A:6B:0F:90
    SHA1: EE:2E:2A:A6:9E:03:9A:3A:1C:17:4A:28:5E:97:20:78:3F:
    Trust this certificate? [no]:
```

5. Enter yes, and then press the Enter or Return key. The following information displays:

```
    Certificate was added to keystore
    [Saving cacerts.jks]
```

CAS Server

1. Download CAS server from the CAS web site.
2. Deploy cas-server-webapp-[version].war to Tomcat
3. Set key store file to Tomcat

```
    keystoreFile="path/to/keystore.jks"
```

4. Start Tomcat

OFBiz Certificate

Set trust store’s file to Java Virtual Machine (JVM) before start OFBiz.

```
    -Djavax.net.ssl.trustStore=path/to/cacerts.jks
```

LDAP Component

OFBiz uses the LDAP component in the plugins to check the security in a web application.

LDAP Properties

LDAP properties file is plugins/ldap/config/ldap.xml.
You can change a filter condition you want.

Attributes

1. Attribute : LDAP attbitue for filter e.g. uid=%u

2. AuthenType : LDAP authentication method e.g. simple
3. AuthenticaionHandler : CAS handler class e.g. org.apache.ofbiz.ldap.cas.OFBizCasAuthenticationHandler
4. AutoPartyId : Party’s id for user login e.g. admin
5. AutoSecurityGroupId : Security group’s id for user login e.g. FULLADMIN
6. BaseDN : The top level ofbiz LDAP directory tree e.g. dc=example,dc=com
7. Filter : LDAP search filter e.g. (objectclass=*)
8. Scope : LDAP search scope parameter e.g. sub,one, etc.
9. URL : LDAP server’s url e.g. ldap://localhost:389
10. UserOFBizLoginWhenLDAPFail : indicate that if LDAP fail then login with normal OFBiz’s user or not. (true/false)

CAS

1. CasLoginUri : URI to CAS login e.g. /login
2. CasLogoutUri : URI to CAS logout e.g. /logout
3. CasUrl : CAS Server’s URL e.g. https://localhost:8443/cas
4. CasValidateUri : URI to CAS validate e.g. /validate
5. CasLdapHandler : LDAP hanlder class e.g. org.apache.ofbiz.ldap.openldap.OFBizLdapAuthenticationHandler
6. CasTGTCookieName : CAS TGT’s cookie name e.g. CASTGC

OpenLDAP

The LDAP component need data from LDAP server (OpenLDAP). The server needs to install, configure and populate OpenLDAP: see at the OpenLDAP web site.

Web Application Security Mapping

Every web application you need to use LDAP (single sign on) feature, you need to change the event’s path of some the security request mappings to org.apache.ofbiz.ldap.LdapLoginWorker class.

checkLogin

```
    <request-map uri="checkLogin" edit="false">
        <description>Verify a user is logged in.</description>
        <security https="true" auth="false"/>
        <event type="java" path="org.apache.ofbiz.ldap.LdapLoginWorker"
invoke="checkLogin"/>
        <response name="success" type="view" value="main"/>
        <response name="error" type="view" value="login"/>
    </request-map>
```

login

```
    <request-map uri="login">
        <security https="true" auth="false"/>
        <event type="java" path="org.apache.ofbiz.ldap.LdapLoginWorker" invoke=
"login"/>
        <response name="success" type="view" value="main"/>
        <response name="requirePasswordChange" type="view"
value="requirePasswordChange"/>
        <response name="error" type="view" value="login"/>
    </request-map>
```

logout

```
    <request-map uri="logout">
            <security https="true" auth="true"/>
            <event type="java" path="org.apache.ofbiz.ldap.LdapLoginWorker"
invoke="logout"/>
            <response name="success" type="request-redirect" value="main"/>
            <response name="error" type="view" value="main"/>
    </request-map>
```