OFBiz Training Handbook

By Praveen Sharma

About the Speaker

Praveen Sharma

Sr. Software Developer at HotWax Systems
4+ years of experience

@praveen--sharma-216014121

@praveen88784881

Member

Contributor

Agenda

a. What is Apache OFBiz?
b. Benefits of OFBiz
c. Framework Components and Development Flow.
d. What is Version Control System?.

What is an ERP and How does it help managing business activities?

ERP

Enterprise resource planning(ERP) is used to automate business processes such as accounting, supply chain management, procurement, order management, relationships management, products and e.t.c.

With ERP systems in place, all data is stored in centralized database. Process across core business are streamlined and automated. Means it enables us to merge different systems together like POS, Mobile Applications, Web Applications and e.t.c, As ERP maintains data in real-time allowing real-time reporting and analysis.

Apache OFBiz

Contents

Introduction

Open For Business(OFBiz) is a suite of enterprise applications allow flexibility to be used across any industry. The common architecture allows developers to easily extend or enhance it to create custom features. The loosely coupled nature of applications make these custom features straightforward, expand and alter.

The tools and architecture of OFBiz make it easy to maintain enterprise applications. This enables businesses to quickly release new functionality and maintain existing functionality without extensive effort. It makes it easy to customize when you have specific requirements.

Benefits

Apache OFBiz comes with loads of out-of-the-box(OOTB) and core modules including:

1. CRM, Order Management & E-Commerce
2. Warehousing and Inventory
3. Accounting (GL,AR,AP,FA)
4. Manufacturing & MRP

Benefits associated with using Apache OFBiz framework including:

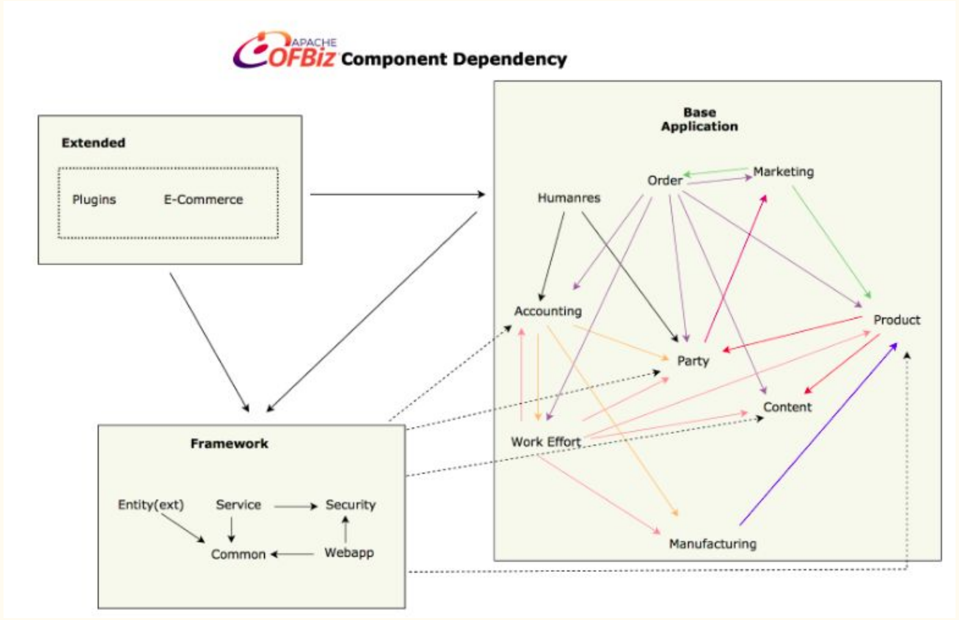
1. Free and open source - No upfront costs
2. Scalable, reliable enterprise solution
3. Fully Customisable
4. Flexible to grow with your business

Component Dependencies

1. Framework: This is the underlying data structure and utilities that supply the essential common features used by the base application and the extended components.

2. Base Applications: These components depend on each other as shown in the diagram. The reason these dependencies are needed is because the components are organized according to high-level business concept of read world business process. And design ensures that lower level components should not depend on higher level ones.

3. Plugins: These components should not depend on each other. If there is such need then that something should be put into the most appropriate component in the applications and both plugins can depend on applications component.



Framework Components

Entity Engine

- The Open For Business Entity Engine is a set of tools and patterns used to model and manage entity specific data.
- An entity is a piece of data defined by a set of fields and a set of relations to other entities.
- The goal of the entity engine is to simplify the enterprise wide use of entity data. This includes definition, maintenance, quality assurance, and development of entity related functionality.

The primary goal of the entity engine is to eliminate the need for entity specific persistence code in as many areas of a transactional application as possible. Granted that this sort of abstraction is a different issue for reporting and similar systems, but for transactional systems such as are used on a day to day basis in all businesses, an entity engine can save a great deal of development effort and dramatically reduce persistence related bugs in the system. These types of applications include everything from ecommerce to accounting to inventory and warehouse management to human resources and so on. These tools can be useful for reporting and analytical systems, but really aren't meant to allow for the wide variety of custom queries that often take place in such tools.

Service Engine

Services are independent pieces of logic which when placed together process many different types of business requirements. Services can be of many different types: Workflow, Rules, Java, SOAP, Groovy, etc. A service with the type Java is much like an event where it is a static method, however with the Services Framework we do not limit to web based applications. Services require input parameters to be in a Map and the results are returned in a Map as well. This is nice since a Map can be serialized and stored or passed via HTTP (SOAP).

Services are defined through the Service Definition and are assigned to a specific Service Engine. Each Service Engine is responsible for invoking the defined service in an appropriate way. Since services are not tied to web based applications this allows services to run when there is no response object available. This allows services to be scheduled to run at specific times to run in the background via the Job Scheduler.

Services have the ability to call other services. So, chaining small services together to accomplish a larger task makes reusing existing services much easier.

Job Scheduler

The scheduler is a multi-threaded component with a single thread used for job managing/scheduling and separate threads used for invocation of each service. When a job is scheduled to run, the scheduler will call the service dispatcher associated with the job to invoke the service in its own thread. This will prevent long or time consuming jobs from slowing down other jobs in the queue.

How it works:
The best usage example of the scheduler is an asynchronous service call. When an asynchronous service is invoked, it is passed to the Job Scheduler to be queued to run. A recurrence entry is created (RecurrenceInfo and RecurrenceRule entities are created), the job is stored, (JobSandbox entity is created) and the context (Map) is serialized and stored (RuntimeData entity is created). The scheduler then adds the job to the top of the list of scheduled jobs (asynchronous services do not have any delay time) and invoked.
Jobs are no longer defined in an XML file. This has been moved to the JobSandbox entity. There is a web based client in planning for adding predefined jobs to the queue, but currently the entities will have to be created by hand.

Request Handler

The Request Handler makes use of a RequestManager helper class to gather a list of request mappings defined in an XML configuration file. The configuration file lives in /WEB-INF/controller.xml for the appropriate context. The mapping consists of a request URI and an optional VIEW name. View names are mapped to in the configuration file as well. A request URI can also be associated with an Event. Events are used to process web related logic by either working directly with the Entity Engine through the Entity Delegator or invoking service(s) to handle the logic through the Service Dispatcher.

When a request is received through the Request Handler it is first looked up in the request mappings; if no mapping is found an 'unknown request error' is returned. Upon a successful lookup, the Security Handler is called to verify if the current request requires authentication and the user making the request is properly authenticated. If the current user is not authenticated the Request Handler redirects the request to a proper login form. After successful authentication or if no authentication is required the Request Handler then looks up a defined event for the request. If an event is found, the request is passed to the EventHandler for processing. After event processing is complete, the default view is looked up for this request, unless the EventHandler requests a specific view instead. The view is then checked in the view mappings and the value is passed to the defined ViewHandler for dispatching. If the view type is none it is expected that the event handles the response itself; if the view type is url the RequestHandler then redirects to the specified URL and no ViewHandler is called.

Event Handler

Event Handlers are defined in the XML config file such as: <handler name="java" type="request" class="org.ofbiz.webapp.event.JavaEventHandler"/>
As in the above request mapping the login event is defined with a type of java. The event type is mapped to an Event Handler using the handler definitions. Custom Event Handlers can be designed and would be implemented like the above example.
Java events are processed by locating the path of the event (package and classname); then, by using the Reflection API, the Event Handler invokes the method defined. A String object is returned to the Request Handler which is mapped to the response element of the request definition.

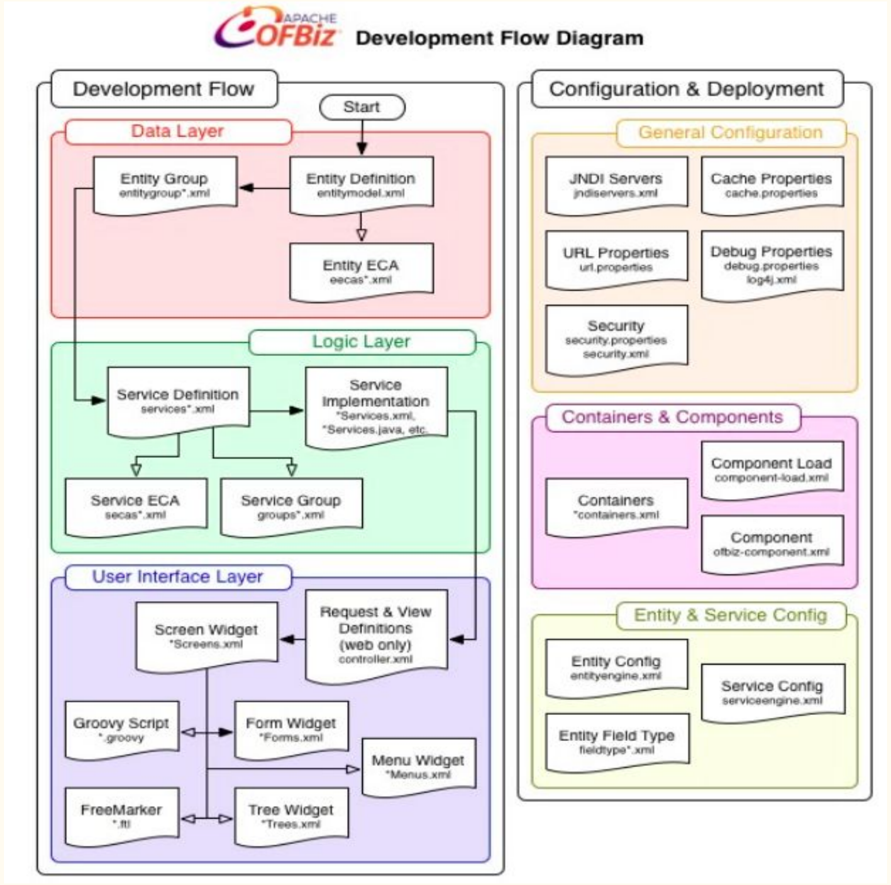
View Handler

View Handlers are defined just like Event Handlers, except the type is view: <handler name="region" type="view" class="org.ofbiz.webapp.view.RegionViewHandler"/>
The view handler handles rendering the next page the user will see. The default view handler supports standard html/jsp pages by dispatching to the page defined in the view definition. Other view handlers like region and velocity use special logic to render the page to the user.
Custom view handlers can be created easily by implementing the ViewHandler interface and setting a definition in the controller.xml file.

Let’s understand better same using diagrams

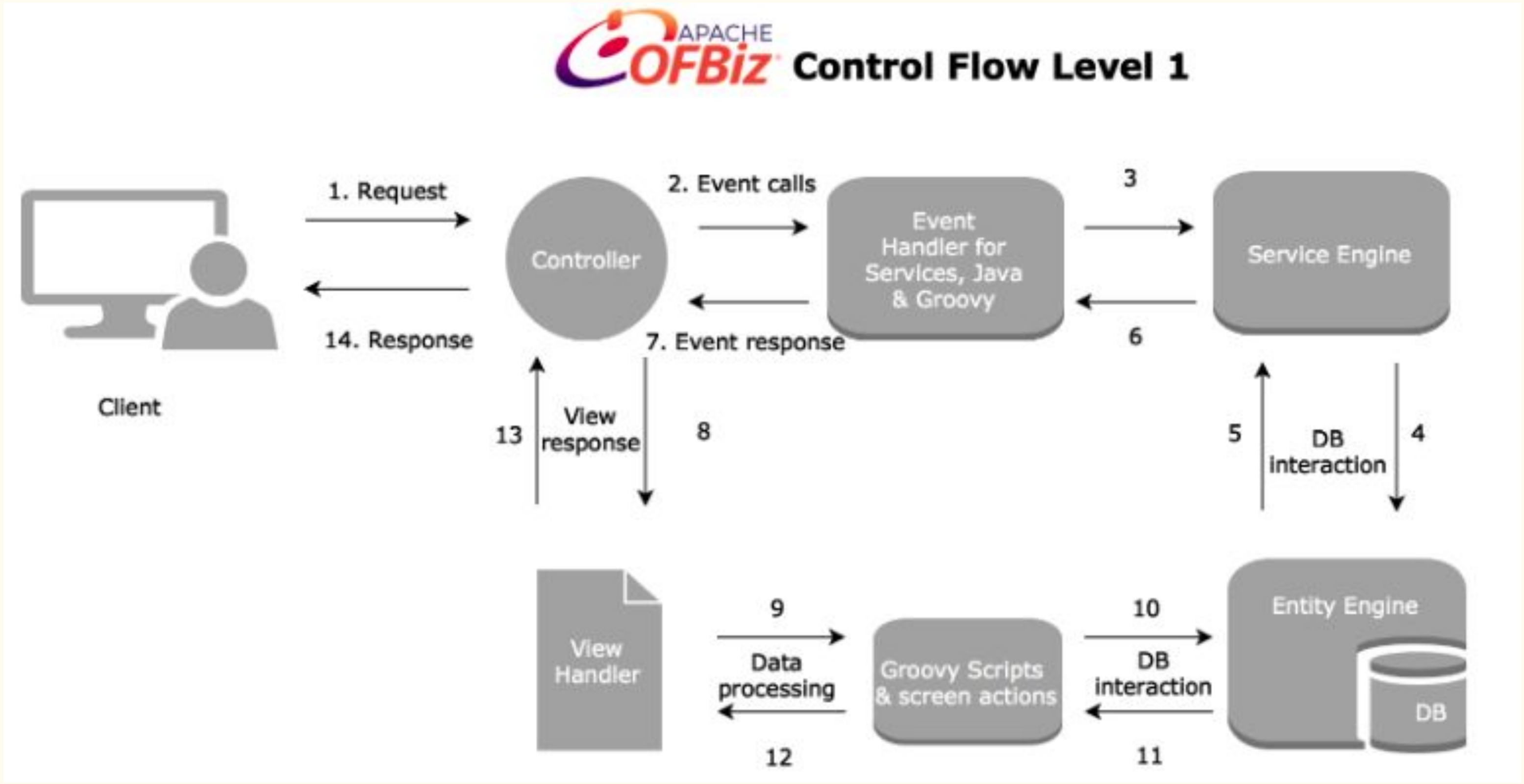
Development Flow

Development flow diagram represents the recommended process for development.



Development Flow Diagram
Development Flow
Start
Data Layer
Entity Group entitygroup*.xml
Entity Definition entitymodel.xml
Entity ECA eecas*.xml
Logic Layer
Service Definition services*.xml
Service Implementation *Services.xml, *Services.java, etc.
Service ECA secas*.xml
Service Group groups*.xml
User Interface Layer
Screen Widget *Screens.xml
Request & View Definitions (web only) controller.xml
Groovy Script *.groovy
Form Widget *Forms.xml
Menu Widget *Menus.xml
FreeMarker *.ftl
Tree Widget *Trees.xml
Configuration & Deployment
General Configuration
JNDI Servers jndiservers.xml
Cache Properties cache.properties
URL Properties url.properties
Debug Properties debug.properties log4j.xml
Security security.properties security.xml
Containers & Components
Containers *containers.xml
Component Load component-load.xml
Component ofbiz-component.xml
Entity & Service Config
Entity Config entityengine.xml
Service Config serviceengine.xml
Entity Field Type fieldtype*.xml

Control Flow



APACHE OFBiz Control Flow Level 1
Client
1. Request
14. Response
Controller
2. Event calls
7. Event response
Event Handler for Services, Java & Groovy
3
6
Service Engine
5
DB interaction
4
13
View response
8
View Handler
9
Data processing
12
Groovy Scripts & screen actions
10
DB interaction
11
Entity Engine
DB

What is Version Control?

What is Version Control

Version control is a name used for software which can help you record changes you make to the files in directory on your computer. Version control softwares and tools (such as Git and SVN) are widely used in software development, research and academic environments. Version control system work best with plain text files such as documents or computer code, but modern versions can be used to track changes in any type of file.

We can reason about and share those changes with others. Multiple contributors can collaborate and made necessary updates which can be maintained as we build up sets of changes overtime, we begin to see some benefits.

Benefits of using version control

- **Collaboration** - Version control allows us to define formalized ways we can work together and share writing and code. For example merging together sets of changes from different parties enables co-creation of documents and software across distributed teams.
- **Versioning** - Having a robust and rigorous log of changes to a file, without renaming files (v1, v2, *final_copy*)
- **Rolling Back** - Version control allows us to quickly undo a set of changes. This can be useful when new writing or new additions to code introduce problems.
- **Understanding** - Version control can help you understand how the code or writing came to be, who wrote or contributed particular parts, and who you might ask to help understand it better.
- **Backup** - While not meant to be a backup solution, using version control systems mean that your code and writing can be stored on multiple other computers.

There are many more reasons to use version control, and we'll explore some of these in the library context, but first let's learn a bit about a popular version control tool called Git.

Recommended tutorial for Git

1. Basic commands: https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html
2. Tutorials: https://www.atlassian.com/git/tutorials/setting-up-a-repository

Important References

Minimum System Requirement for OFBiz Setup

Best Practices

How to migrate OFBiz from Derby to MySQL database

Groovy DSL for OFBiz business logic

OFBiz Tutorial: A Beginners Development Guide for 17.12
Apache OFBiz Youtube Tutorials and Videos

Service Engine Guide

Entity Engine Guide

General Entity Overview

Control Servlet Guide

JSP Tag Library Guide

Glossary of OFBiz Terms and Concepts

Data Model Diagrams

Universal Data Model

Additional Resources

1. Best Practice Guide
2. OFBiz Tutorials on Youtube