Coding Conventions

General Formatting Conventions

The OFBiz project follows the Sun coding standards for Java source code. For information regarding this standard please visit http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

The line length of 150 is an exception to the Sun standard

Here is an example of a Java formatter for Eclipse

For Groovy we suggest you read http://www.groovy-lang.org/style-guide.html but we don't expect people to follow all details there, still a good read..

HTML code should be XHTML compliant. Groovy and Javascript files should follow the same conventions as Java files.

We don't want tab characters in text files in SVN, because different editors and tools interpret tabs differently. All indentation should be done purely with spaces.

In accordance with this standard we use 4 space indention rather than tabs, and set the tab size to 8 spaces to make code with tabs easier to pick out. Make sure no code contains tab characters.
For XML/HTML/FTL files we follow the standard XML/HTML/FTL format (TODO: add link to a reference here). XML/HTML/FTL files should use 2 spaces instead of 4, but there are still number of files using 4 space indention. So it's better to keep the indentation used in each file to avoid confusion when committing.

To change tabs to spaces in Eclipse, change these preferences in Window -> Preferences->

1. Java -> Code Style -> Code Formatter -> In dropdown pick -> Java Conventions built-in -> Indentation, select Tab policy as "Spaces Only". Enter a new name for this profile and click "OK".
2. General -> Editors -> Text Editors -> Insert spaces for tabs
3. XML -> XML Files -> Editors -> Text Editors -> Indent using spaces (4)
4. Unfortunately the main FreeMarker plugins has no similar functionnality yet. But there is somehow a (sometimes annoying) solution :
   - Install AnyEdit plugin (http://andrei.gmxhome.de/eclipse/) and set its parameters in General -> Editors -> AnyEdit Tools. When using AnyEdit plugin, you should set it to
     - Convert tab to space (it's by default in last versions)
     - Add filters (like *.js, *.css) for files you don't want to auto-convert)

In Eclipse, for Java, you may use Ctrl+I to be sure of correct indentation.

Files paths

Branches concerned

Only in trunk and R16.11 for now

- **Java**
  The OFBiz project follows the Maven Standard Directory Layout to organise its Java files. At the moment of writing there are only a few unit tests (but a lot of what we so far call integration tests) so you will mostly find the *../src/main/java/org/apache/ofbiz*.. path in components. As of today in few components in framework (base, entity, start, webapp) you will also find the *..src/test/java/org/apache/ofbiz*.. path
- **Groovy**
  We decided to use a *groovyScripts* sub-folder under components folders to put parsed Groovy scripts. We named it groovyScripts rather than *script* because other script languages could be used in the future
  Even if it's no used yet we reserved the *groovy* sub-folder under components folders for pre-compiled Groovy scripts. A distant goal is to replace Minilang by a Groovy DSL...
- **Freemarker**
  The Freemarker templates are in *template* sub-folders under components folders

- **Minilang**
  The Minilang map processors and minilang scripts are in *minilang* sub-folders under components folders

Misc.

In OFBiz we don't set serialVersionUID but let the system handles that for us and rather use @SuppressWarnings("serial") where necessary.
More at https://markmail.org/message/jjhf5p5hbhon7vam and above.

Entity Definitions

Entity Names

Entities are named in accordance with the Java naming conventions for classes (e.g. ExampleEntity).
The physical database names are generated by the system by applying the Java naming convention of constants to the entity names (e.g. ExampleEntity => EXAMPLE_ENTITY).
Important: database table names should never be longer than 30 characters, so you have to keep this in mind when you define a new entity name.
For example: the entity name "AnExampleOfALongEntityName" (that is is 26 characters long) is transformed into the database table name AN_EXAMPLE_OF_A_LONG_ENTITY_NAME, that is 32 characters long and exceeds the maximum length of 30 characters and so it is not a good choice.

Entity Field Names

Entity fields are named in accordance with the Java naming conventions for fields (e.g. exampleField). OOTB, they have the same length constraint than above : 30 characters max (including "_" separators). But your mileage may vary. For instance , for fields parts of **Primary Keys** it depends on the DB you use. The maximum length of a column name for Derby is 30 chars, but OOTB for an id you will get a truncation error from Derby jdbc drivers which limits them to the fieldype defined. So you will get a message like "A truncation error was encountered trying to shrink VARCHAR ... to length 20". Of course you can increase this value up to 30. This constraint does not exist with Postgres for example. Today (2011-06-15) on a 64 bits Debian system running Posgres 9.0.3, I tried a column name of length 200 and it worked... I guess it's more than enough...

Foreign Keys names have a length constraint of 18 characters (including "_" separators). It's better to set them by hand, to avoid indesirable effects. These constraints do not come from OFBiz but from some DB. To be able to use any DB those constraints should be respected to allow DB portability.

TODO: view-entity names and conventions

Service Definitions

Best Practices

Apache OFBiz Best Practices Guide

License Headers

Each and every source file should contain the ASL2.0 header:

`http://svn.apache.org/repos/asf/ofbiz/trunk/APACHE2_HEADER`

Adding author information to source files (.e.g. using the @author tag) is a discouraged practice: clearer information about the contributors of a file should be derived from the commit logs.

Naming Conventions for Artifacts and controller entries

Controller

Most of the time, a controller file (in OFBiz controller.xml) contains a lot of request-maps and view-maps. To help the reader, set a blank line between request-map or view-map element only when you change the functionnal group (entity, workflow or IHM treatment). Of course comments at the top of sections are welcome...

Request without event

In most of case, the request name is the same than the view-map called. For the request-map name use java naming convention for Objects (ex : "FindParty"), and declare all in one line.

```
    <request-map uri="FindParty"><security https="true" auth="true"/><response name="success" type="view"
value="FindParty"/></request-map>
```

Request with event

For the request-map name use java naming convention for field or method (ex : "editParty"), and declare each xml element on separate line

```
<request-map uri="createPerson">
     <security https="true" auth="true"/>
     <event type="service" invoke="createPerson"/>
     <response name="success" type="view" value="EditPerson"/>
     <response name="error" type="view" value="EditPerson"/>
 </request-map>
```

View

For the view-map name use java naming convention for Objects (ex : "FindParty"), and declare all in one line.

```
<view-map name="FindParty" type="screen" page="component://mycomponent/widget/PartyScreens.xml#FindParty"/>
```

widget screens

TODO: conventions for screen names (top-level screens, included screens, decorators)

TODO: conventions for form names

TODO: conventions for menu names

Improve MacroXXXRenderer and FtlMacroXXXLibrary

Whn you realize some modification on the MacroXXX Renderer (XXX can take value Screen, Form, Menu and Tree) and you change a macro ftl definition change on each MacroXXXlibrary corresponding.

Example ; if your change *renderTextField* on *htmlMacroFromLibrary*, propage the improvment on *{csv|xml|text|fop}MacroFromLibrary*

TODO: conventions for artifacts names such as ftl templates and bsh scripts