

# *OFBiz An Insider View*

Prepared By:

Basil Argasosy  
Senior Computer Engineering Student  
King Fahd University of Petroleum & Minerals (K.F.U.P.M)  
September 01, 2005

## Contact Information

[s208603@kfupm.edu.sa](mailto:s208603@kfupm.edu.sa)  
[st208603@ccse.kfupm.edu.sa](mailto:st208603@ccse.kfupm.edu.sa)

or through my [personal webpage](#)

# OFBiz : An Insider View

## Introduction:

The OFBiz framework utilizes the common Three-Tier “Layers” Architecture model in all its applications. It has the Data Layer, the Business “logic” layer, and the Presentation “user interface” layer. The Data Layer and the Service layer have their own engines that are responsible for interaction with the layer.

- 1) Data Model Layer: It represents the database. There is an Entity Engine that is responsible of this layer that includes database connection, data retrieval, data storage...etc. It used the java Generic Delegator class to connect with the database, and it uses the java Generic Value to represent an entity row to be inserted in the database.
- 2) Business Logic Layer: It represents the logic, or the services provided to the user and performed on the data layer "database". There can be services of many types like java, SOAP, simple, workflow, etc. and each type of service has its own handler. There is a Service Engine that is responsible for dealing with services, calling the service, etc.
- 3) Presentation Layer: OFBiz has moved to use "Screens" to represent the OFBiz pages. So, each page should normally be represented as a screen. An OFBiz page consists of many components like headers, footer, appheader,..etc, so when rendering the page, these are all combined in the order they were placed, or included, in the screen.
- 4) Servlet Container : This is the main servlet that controls all the application “controller.xml” .The controller defines the event handlers and the view handler, type of the services, the location of the views..etc. Web.xml is an important file to configure the main servlet(s) and also to control to tomcat server.

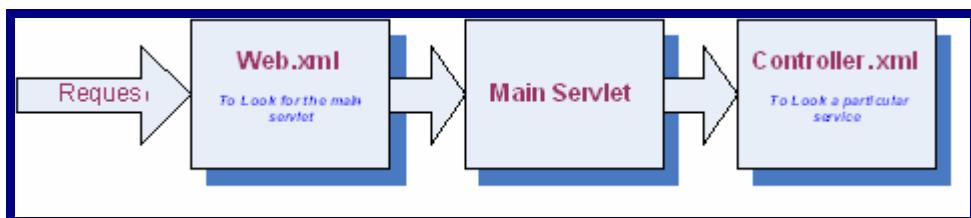
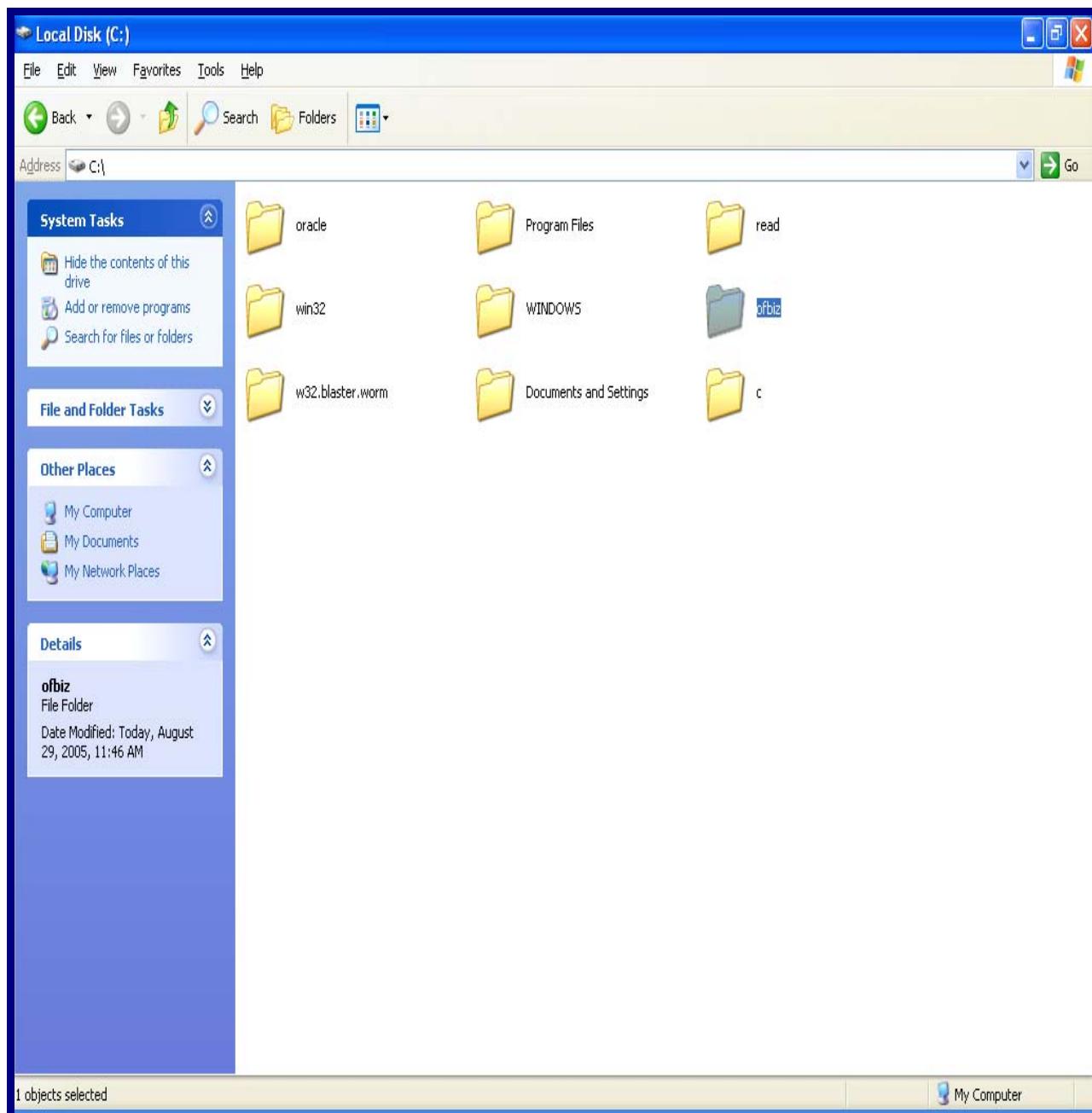


Figure 1

## Practical Overview:

Before starting to build our new application, let's have a look inside the OFBiz. Here is the OFBiz application folder on the C drive.



**Figure 2**

Having a look inside the OFBiz folder, we would see the following :

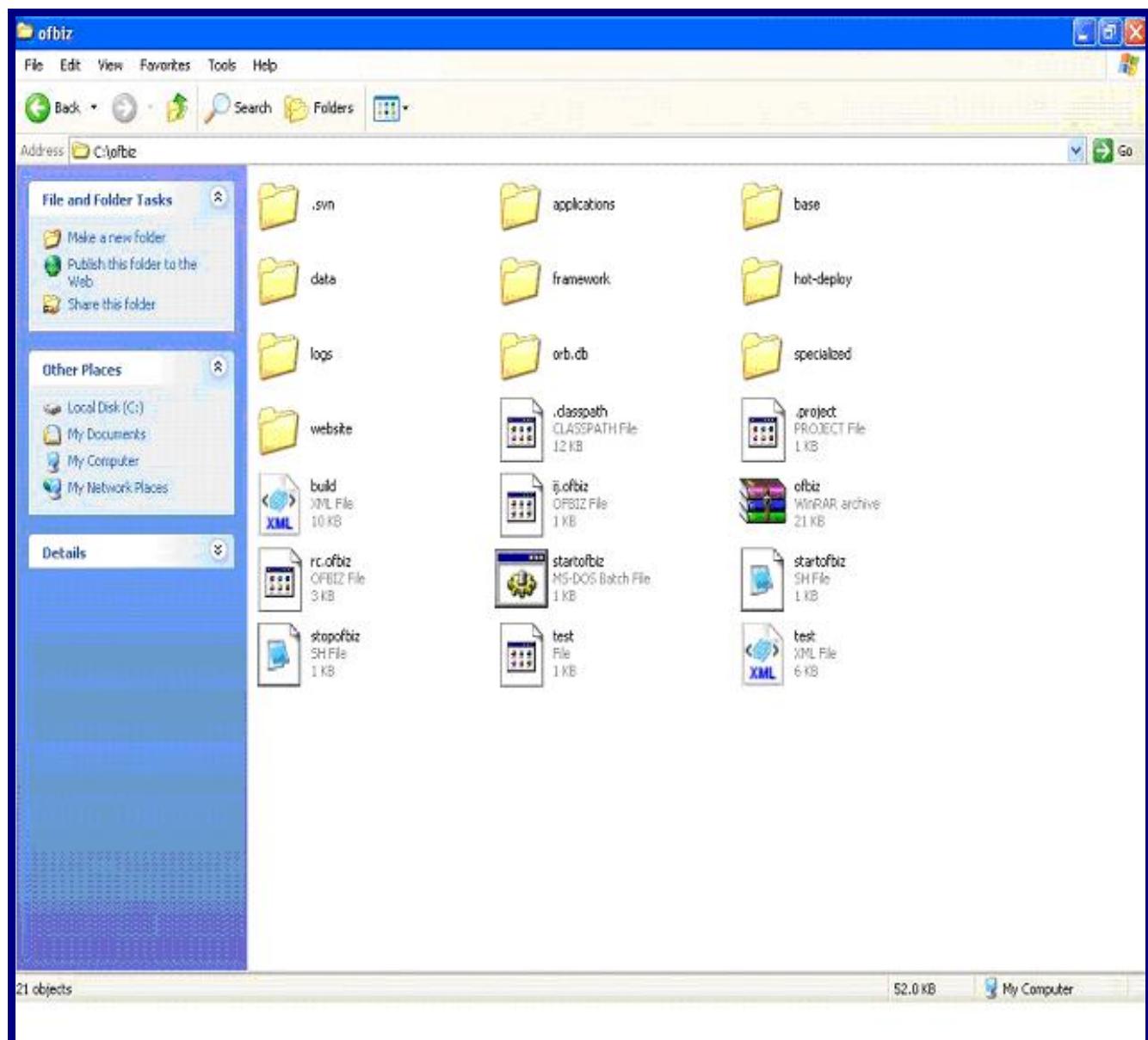


Figure 3

These folders are as follows: [1]

.svn folder : contains the weekly update batches for the OFBiz framework and applications.

applications folder : contains the application components created with OFBiz, when you create your own application, it *should* be placed completely in this folder or the application can be placed in the hot-deploy or the specialized folder.

base folder : contains java classes , xml files and xml schema files, for OFBiz starting up and configuration.

data folder : contains some files for the database specification.

framework folder : contains the OFBiz framework components , like the Entity Engine, the Service Engine, the common folder that contains files that is common for any application..etc.

hot-deploy folder : this folder can also hold some applications , where the components of these application are loaded automatically without the need for loading them explicitly as we will see later when looking inside an application.

logs folder : the OFBiz uses [the log4j project](#) for its logging System, this folder contains the log files.

specialized folder : contains some extra applications like “community” and “wholesale” which are not part of the OFBiz core.

website folder : contains the www.ofbiz.org website html pages.

startofbiz.bat : this file is used to start running the OFBiz.

Now , we would look at an application, so having a look inside the applications folder :

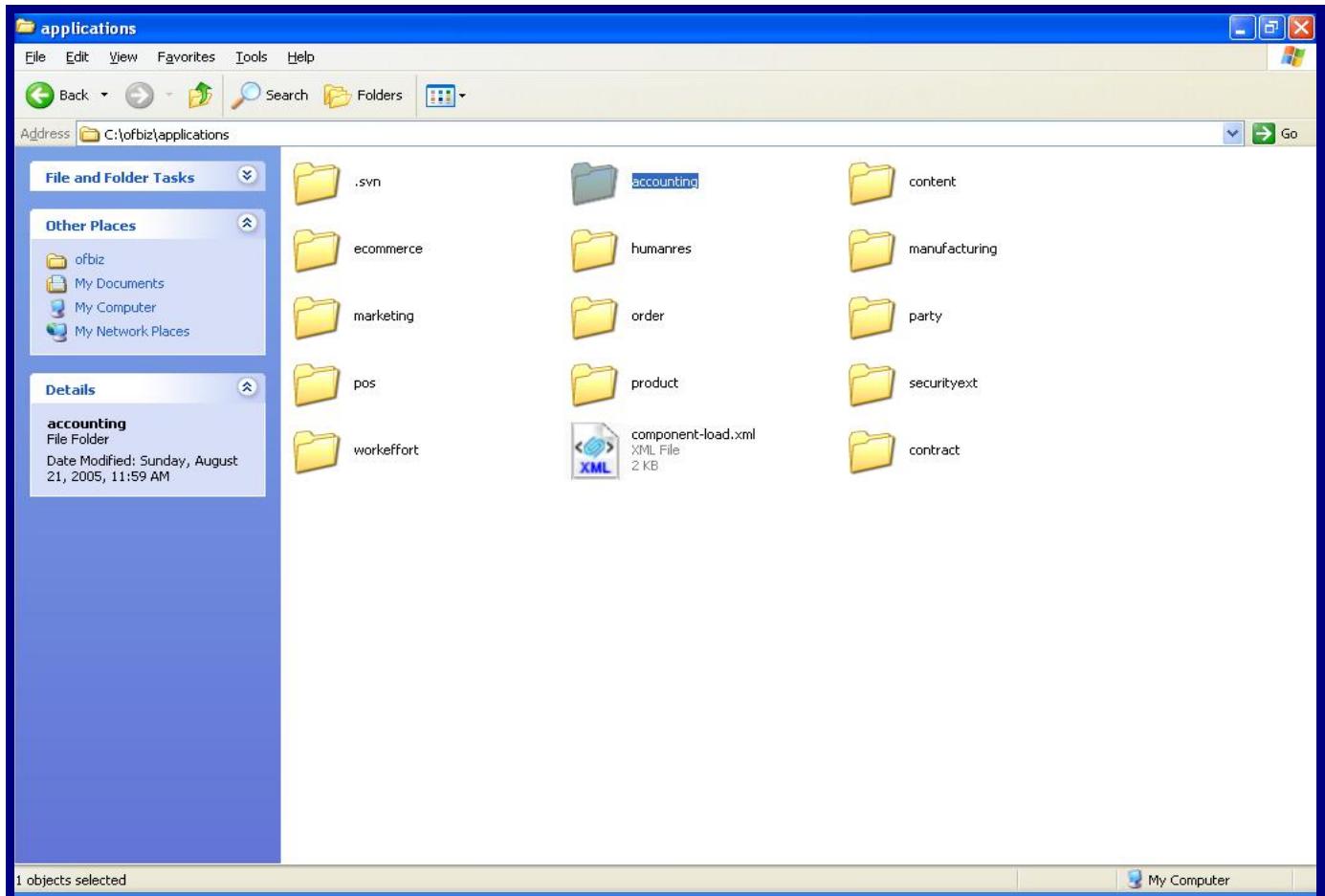
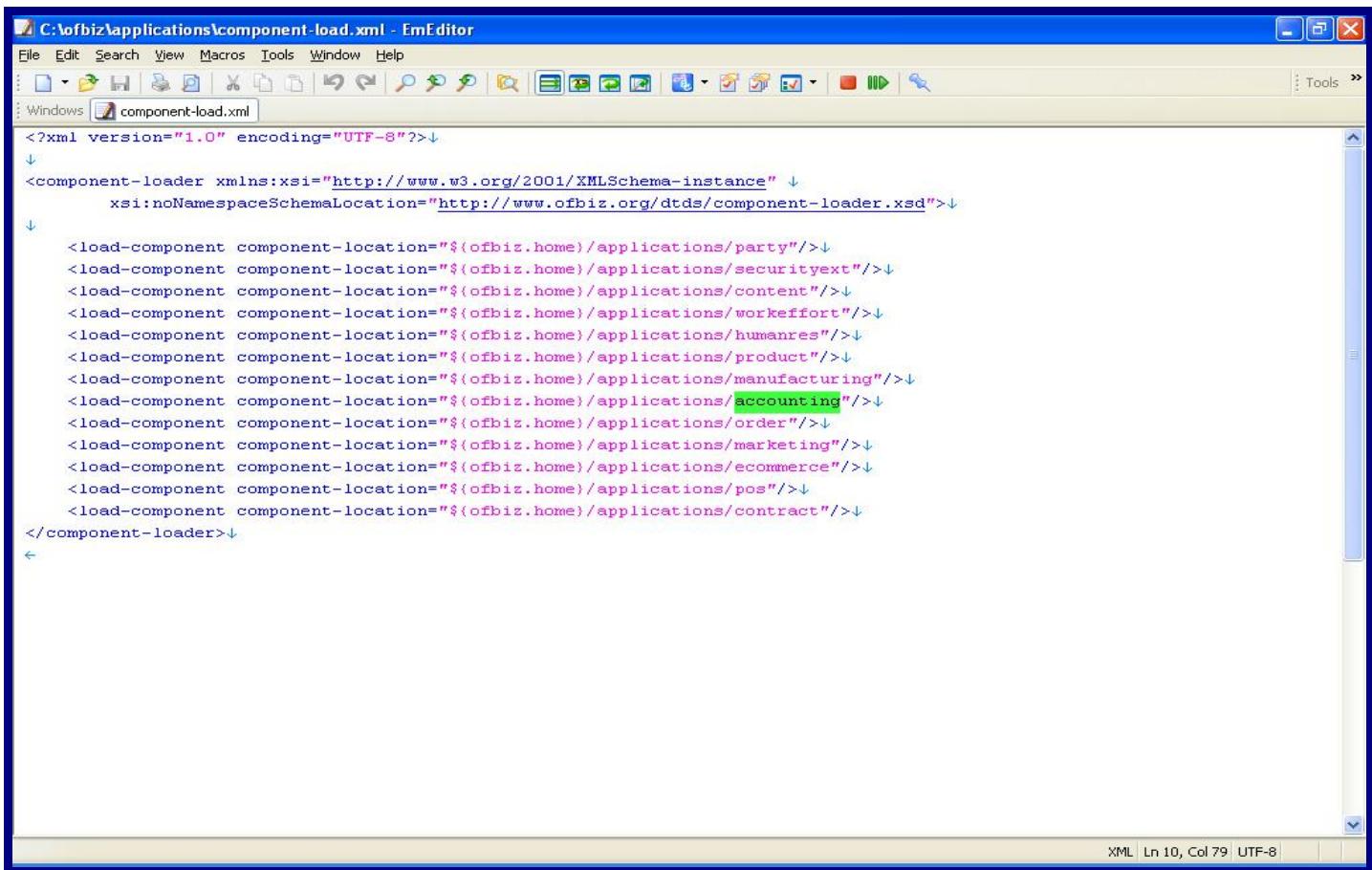


Figure 4

here are some applications, the accounting application ,the party application, the the order application, ...etc.

The component-load.xml file is a very important file, because without it, the OFBiz can not load any application “unless this application is placed in the hot-deploy folder as mentioned earlier”. Whenever you create a new application, that is you add a new folder beside these other folders “party, order,...etc” , you need to tell the OFBiz to load this application, and this is done with the component-load.xml file. It defines the location for all applications that needs to be loaded when the OFBiz starts.

Here is the load-component file:



The screenshot shows the EmEditor application window with the file 'component-load.xml' open. The code in the editor is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<component-loader xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/component-loader.xsd">
    <load-component component-location="${ofbiz.home}/applications/party"/>
    <load-component component-location="${ofbiz.home}/applications/securityext"/>
    <load-component component-location="${ofbiz.home}/applications/content"/>
    <load-component component-location="${ofbiz.home}/applications/workeffort"/>
    <load-component component-location="${ofbiz.home}/applications/humanres"/>
    <load-component component-location="${ofbiz.home}/applications/product"/>
    <load-component component-location="${ofbiz.home}/applications/manufacturing"/>
    <load-component component-location="${ofbiz.home}/applications/accounting"/>
    <load-component component-location="${ofbiz.home}/applications/order"/>
    <load-component component-location="${ofbiz.home}/applications/marketing"/>
    <load-component component-location="${ofbiz.home}/applications/ecommerce"/>
    <load-component component-location="${ofbiz.home}/applications/pos"/>
    <load-component component-location="${ofbiz.home}/applications/contract"/>
</component-loader>
```

The XML declaration at the top includes a schema location reference to 'component-loader.xsd'. The file contains a single element, 'component-loader', which contains multiple 'load-component' elements. Each 'load-component' element specifies a component location relative to the 'ofbiz.home' directory.

Figure 5

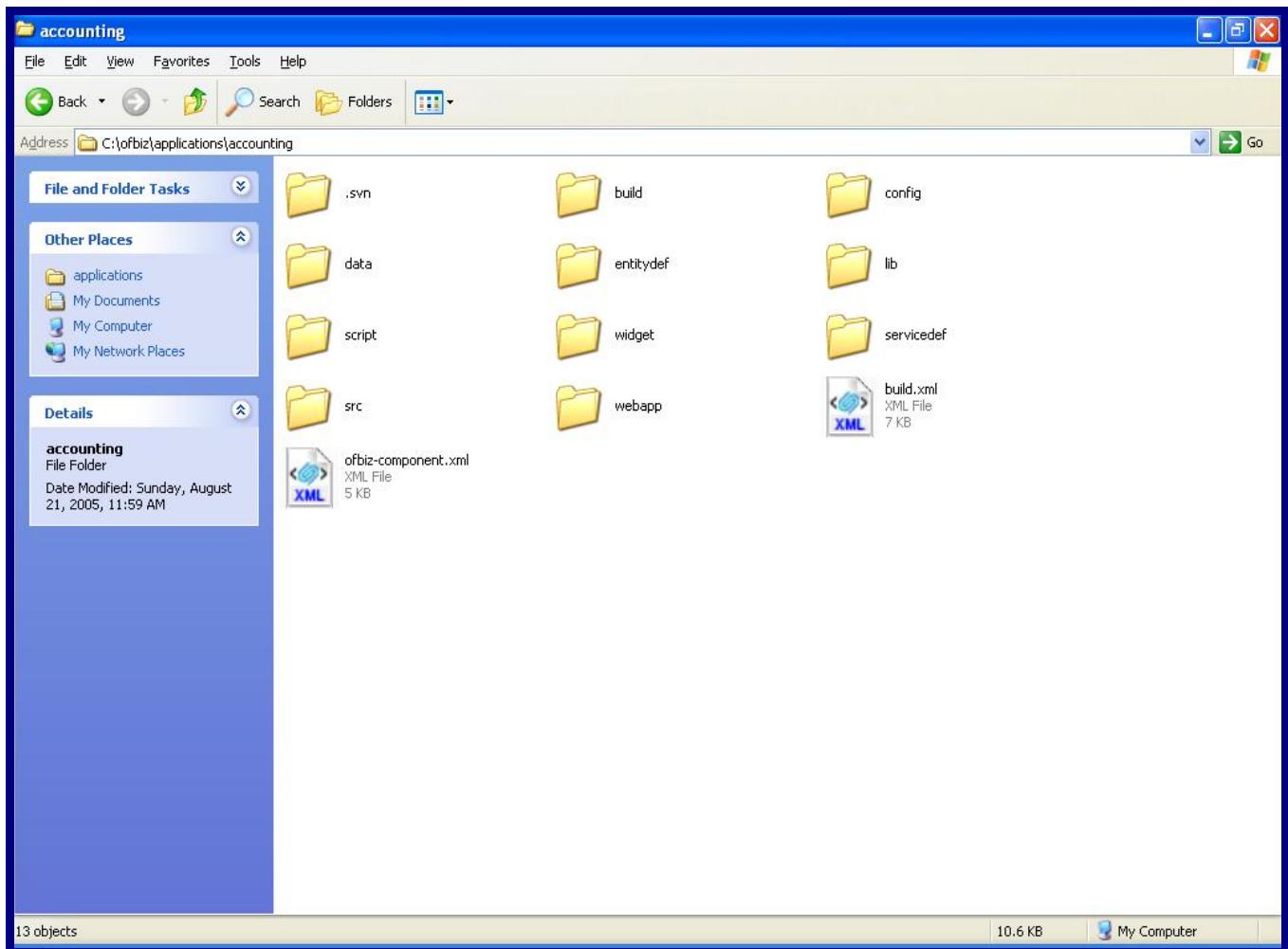
### Note : [1]

In OFBiz , any application is placed inside a component, that is the OFBiz deals with components that contain one or more applications ,I guess this is why the file is a “load-component” file not a “load-application” file.

Now we will have a look at an application, we will take the “Accounting application” as an example , and all the application have the same structure , generally.

## Case Study : Accounting Application :

The accounting application holds many smaller applications inside , one of them is the Agreement . We would go through the three –tiers of this application, i.e., the Data Layer, The Business Logic Layer and the Presentation Layer and the Controller.



**Figure 6**

These folders are as follows : [1]

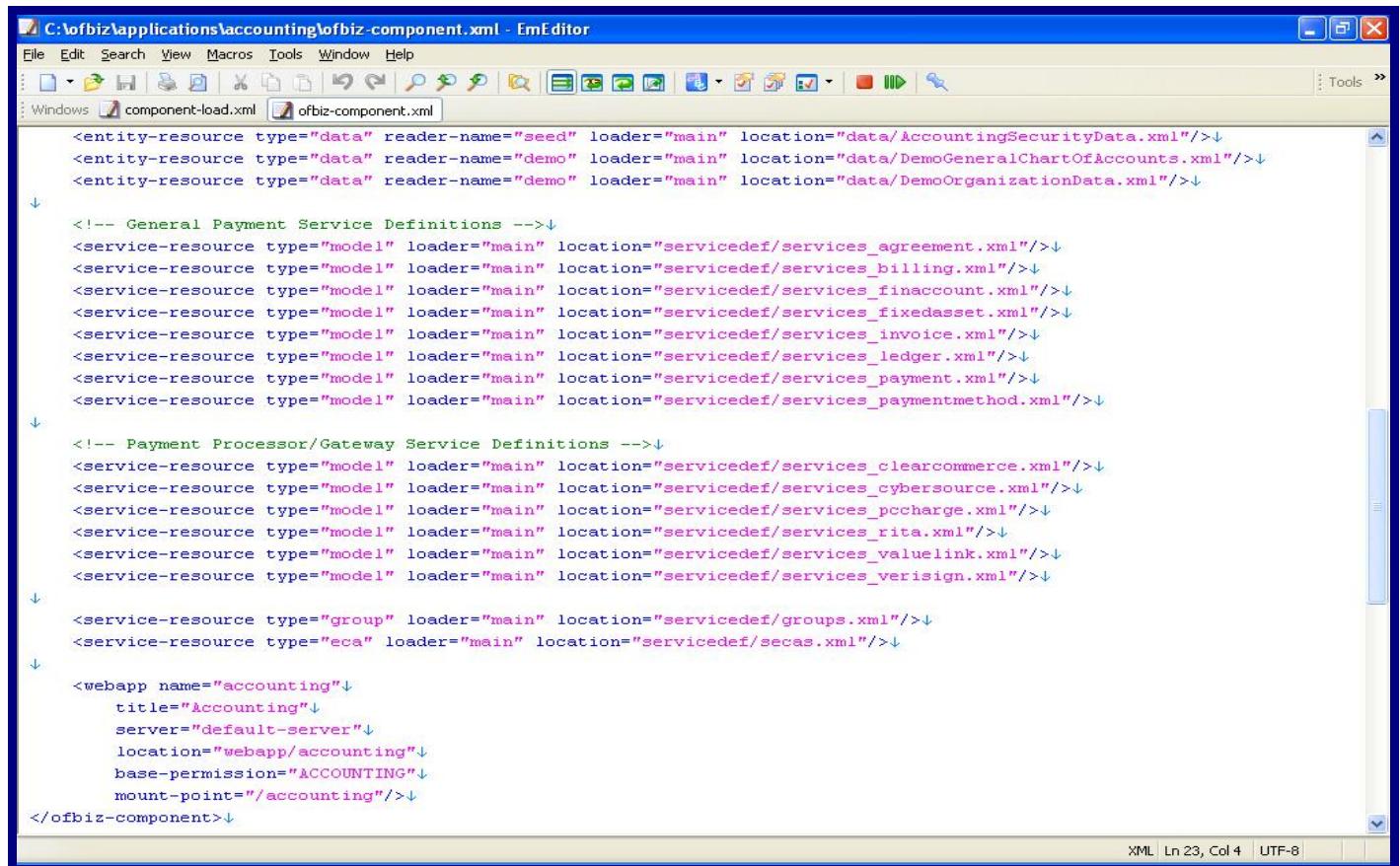
ofbiz-component.xml : defines this application by specifying where is the location of its

data model : <entity-resource>

business logic : <service-resource>

web applications <webapp .....> .

It is very important to notice that any entity resource file or service resource file should be referenced to in the ofbiz-component.xml.



```
C:\ofbiz\applications\accounting\ofbiz-component.xml - EmEditor
File Edit Search View Macros Tools Window Help
Windows component-load.xml ofbiz-component.xml
<entity-resource type="data" reader-name="seed" loader="main" location="data/AccountingSecurityData.xml"/>↓
<entity-resource type="data" reader-name="demo" loader="main" location="data/DemoGeneralChartOfAccounts.xml"/>↓
<entity-resource type="data" reader-name="demo" loader="main" location="data/DemoOrganizationData.xml"/>↓
↓
<!-- General Payment Service Definitions -->↓
<service-resource type="model" loader="main" location="servicedef/services_agreement.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_billing.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_finaccount.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_fixedasset.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_invoice.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_ledger.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_payment.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_paymentmethod.xml"/>↓
↓
<!-- Payment Processor/Gateway Service Definitions -->↓
<service-resource type="model" loader="main" location="servicedef/services_clearcommerce.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_cybersource.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_pccharge.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_rita.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_valuelink.xml"/>↓
<service-resource type="model" loader="main" location="servicedef/services_verisign.xml"/>↓
↓
<service-resource type="group" loader="main" location="servicedef/groups.xml"/>↓
<service-resource type="eca" loader="main" location="servicedef/secas.xml"/>↓
↓
<webapp name="accounting">
  title="Accounting">↓
  server="default-server">↓
  location="webapp/accounting">↓
  base-permission="ACCOUNTING">↓
  mount-point="/accounting"/>↓
</webapp>↓
</ofbiz-component>↓
```

Figure 7

build.xml : as its name states, this file is used to tell the ant program how to build the OFBiz application.

.svn : contains the weekly updates batches to this application.

build : contains the java compiled codes “.class files “ and the libraries for the accounting application.

**config** : used generally for data configuration , an example is , it is used to support different languages interfaces , inside it you will find some files for different languages, and based on the user interface language, one of these files will be used.

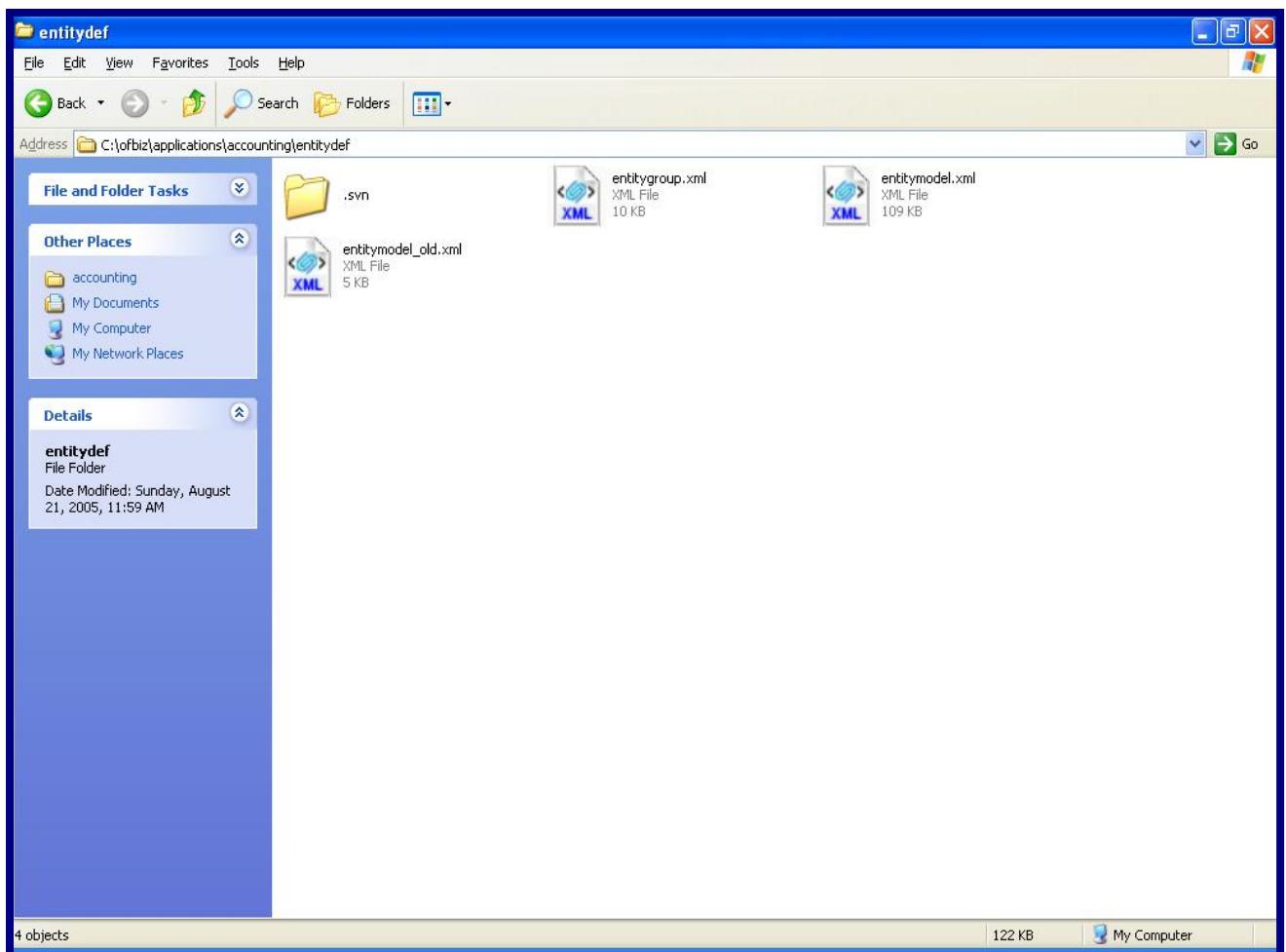
**data** : contains the seed data “data loaded when ofbiz starts” and the demo data.

*Finally, we are last with the entitydef and the servicedef. For these two we always have two parts : definition and implementation.*

**entitydef** :

contains the data layer definitions and implementations, i.e. the database relational tables and their relationships.

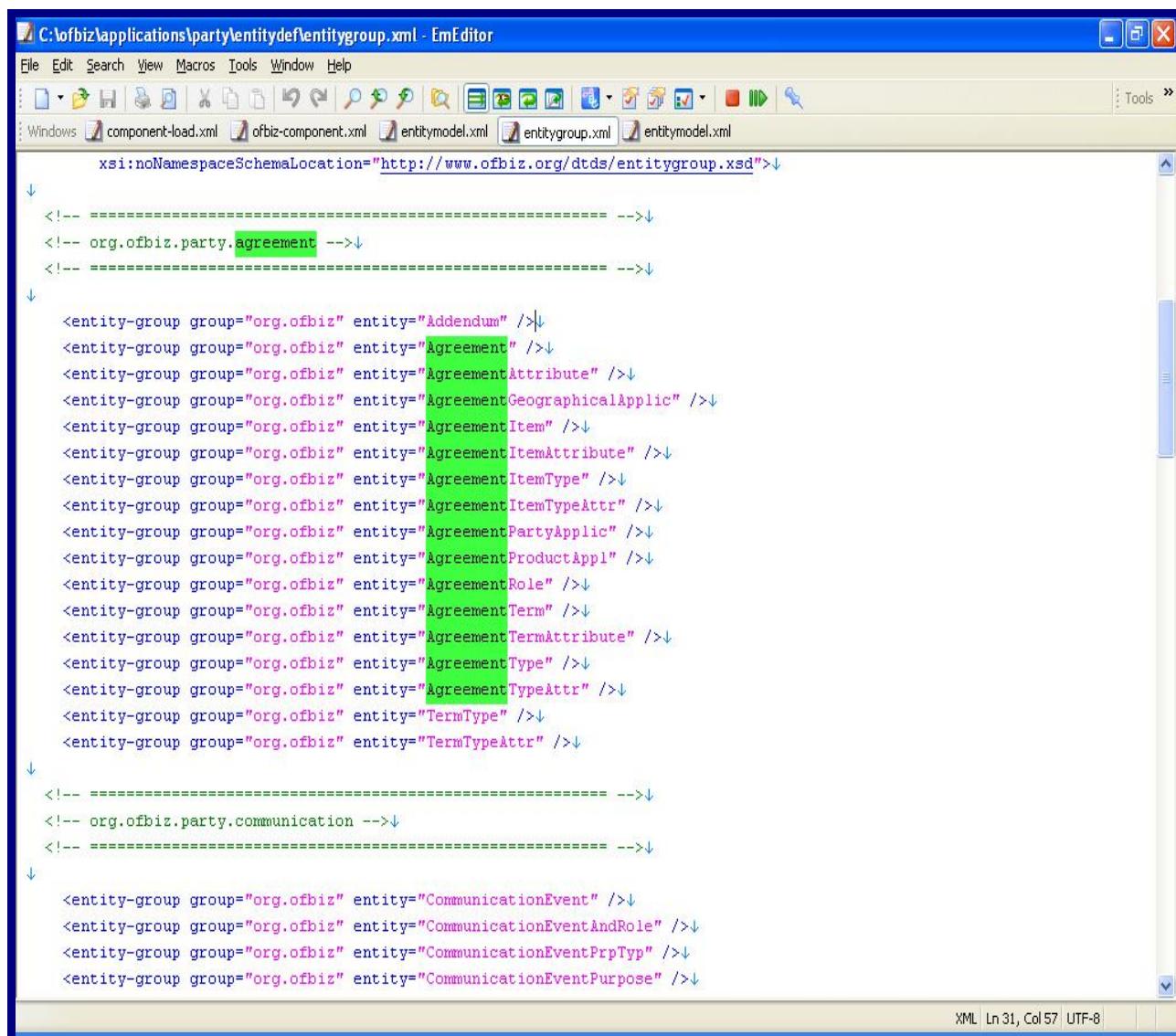
Inside this folder , there will always be two main files, one for definition and the other for implementation.



**Figure 8**

## entitygroup.xml :

It defines the tables of the database. For example, we have Agreement , AgreementAttribue,AgreementItem,AgreementItemAttribute,....etc . It is strange as we are studying the Agreement application, which is under the accounting application , but its definition is located in the entitydef directory of the party application!. May be it will be moved to the accounting application soon!. Nevertheless, all the applications follow the same pattern, and even if the Agreement Entities were taken out from the party Application and replaced in the accounting Application it would work exactly the same.



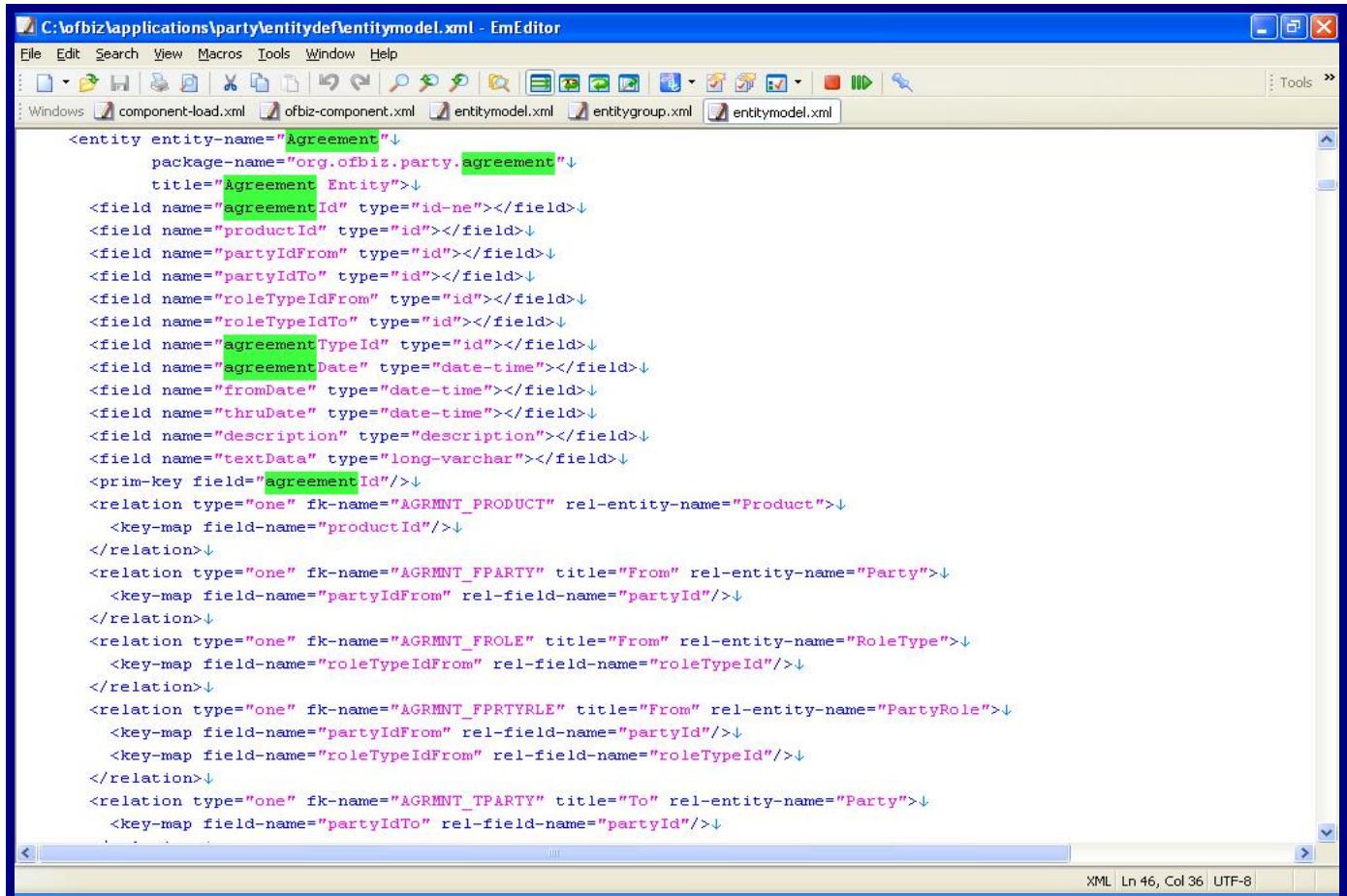
The screenshot shows the EmEditor XML editor window with the file 'entitygroup.xml' open. The window title is 'C:\ofbiz\applications\party\entitydef\Entitygroup.xml - EmEditor'. The menu bar includes File, Edit, Search, View, Macros, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Find, Copy, Paste, etc. The status bar at the bottom right shows 'XML | Ln 31, Col 57 | UTF-8'. The code in the editor is an XML document defining entity groups for the 'org.ofbiz.party.agreement' and 'org.ofbiz.party.communication' modules. The code is color-coded for readability, with green highlighting specific entities like 'Agreement', 'AgreementItem', etc.

```
xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/entitygroup.xsd">>↓
<!-- ===== -->↓
<!-- org.ofbiz.party.agreement -->↓
<!-- ===== -->↓
↓
<entity-group group="org.ofbiz" entity="Addendum" />↓
<entity-group group="org.ofbiz" entity="Agreement" />↓
<entity-group group="org.ofbiz" entity="AgreementAttribute" />↓
<entity-group group="org.ofbiz" entity="AgreementGeographicalApplc" />↓
<entity-group group="org.ofbiz" entity="AgreementItem" />↓
<entity-group group="org.ofbiz" entity="AgreementItemAttribute" />↓
<entity-group group="org.ofbiz" entity="AgreementItemType" />↓
<entity-group group="org.ofbiz" entity="AgreementItemTypeAttr" />↓
<entity-group group="org.ofbiz" entity="AgreementPartyApplc" />↓
<entity-group group="org.ofbiz" entity="AgreementProductAppl" />↓
<entity-group group="org.ofbiz" entity="AgreementRole" />↓
<entity-group group="org.ofbiz" entity="AgreementTerm" />↓
<entity-group group="org.ofbiz" entity="AgreementTermAttribute" />↓
<entity-group group="org.ofbiz" entity="AgreementType" />↓
<entity-group group="org.ofbiz" entity="AgreementTypeAttr" />↓
<entity-group group="org.ofbiz" entity="TermType" />↓
<entity-group group="org.ofbiz" entity="TermTypeAttr" />↓
↓
<!-- ===== -->↓
<!-- org.ofbiz.party.communication -->↓
<!-- ===== -->↓
↓
<entity-group group="org.ofbiz" entity="CommunicationEvent" />↓
<entity-group group="org.ofbiz" entity="CommunicationEventAndRole" />↓
<entity-group group="org.ofbiz" entity="CommunicationEventPrpTyp" />↓
<entity-group group="org.ofbiz" entity="CommunicationEventPurpose" />↓
```

Figure 9

## entitymodel.xml :

It implements these tables that were just defined in the entitygroup.xml, i.e., it gives details about their fields, types, relationships, ...etc. As an Example, the Agreement table or Agreement entity:



The screenshot shows the EmEditor application window with the file 'entitymodel.xml' open. The XML code defines the 'Agreement' entity with various fields and relationships. The 'agreementId' field is highlighted in green, indicating it is the primary key. Other fields like 'productId', 'partyIdFrom', 'partyIdTo', 'roleTypeIdFrom', 'roleTypeIdTo', 'agreementType', 'agreementDate', 'fromDate', 'thruDate', 'description', and 'textData' are also highlighted in green. Relationships are defined with 'product' (one-to-one), 'party' (one-to-one), 'RoleType' (one-to-one), and 'PartyRole' (one-to-one) entities.

```
<entity entity-name="Agreement">
    package-name="org.ofbiz.party.agreement">
        title="Agreement Entity">
<field name="agreementId" type="id-ne"></field>
<field name="productId" type="id"></field>
<field name="partyIdFrom" type="id"></field>
<field name="partyIdTo" type="id"></field>
<field name="roleTypeIdFrom" type="id"></field>
<field name="roleTypeIdTo" type="id"></field>
<field name="agreementType" type="id"></field>
<field name="agreementDate" type="date-time"></field>
<field name="fromDate" type="date-time"></field>
<field name="thruDate" type="date-time"></field>
<field name="description" type="description"></field>
<field name="textData" type="long-varchar"></field>
<prim-key field="agreementId"/>
<relation type="one" fk-name="AGRMNT_PRODUCT" rel-entity-name="Product">
    <key-map field-name="productId"/>
</relation>
<relation type="one" fk-name="AGRMNT_FPARTY" title="From" rel-entity-name="Party">
    <key-map field-name="partyIdFrom" rel-field-name="partyId"/>
</relation>
<relation type="one" fk-name="AGRMNT_FROLE" title="From" rel-entity-name="RoleType">
    <key-map field-name="roleTypeIdFrom" rel-field-name="roleTypeId"/>
</relation>
<relation type="one" fk-name="AGRMNT_FPARTYRLE" title="From" rel-entity-name="PartyRole">
    <key-map field-name="partyIdFrom" rel-field-name="partyId"/>
    <key-map field-name="roleTypeIdFrom" rel-field-name="roleTypeId"/>
</relation>
<relation type="one" fk-name="AGRMNT_TPARTY" title="To" rel-entity-name="Party">
    <key-map field-name="partyIdTo" rel-field-name="partyId"/>

```

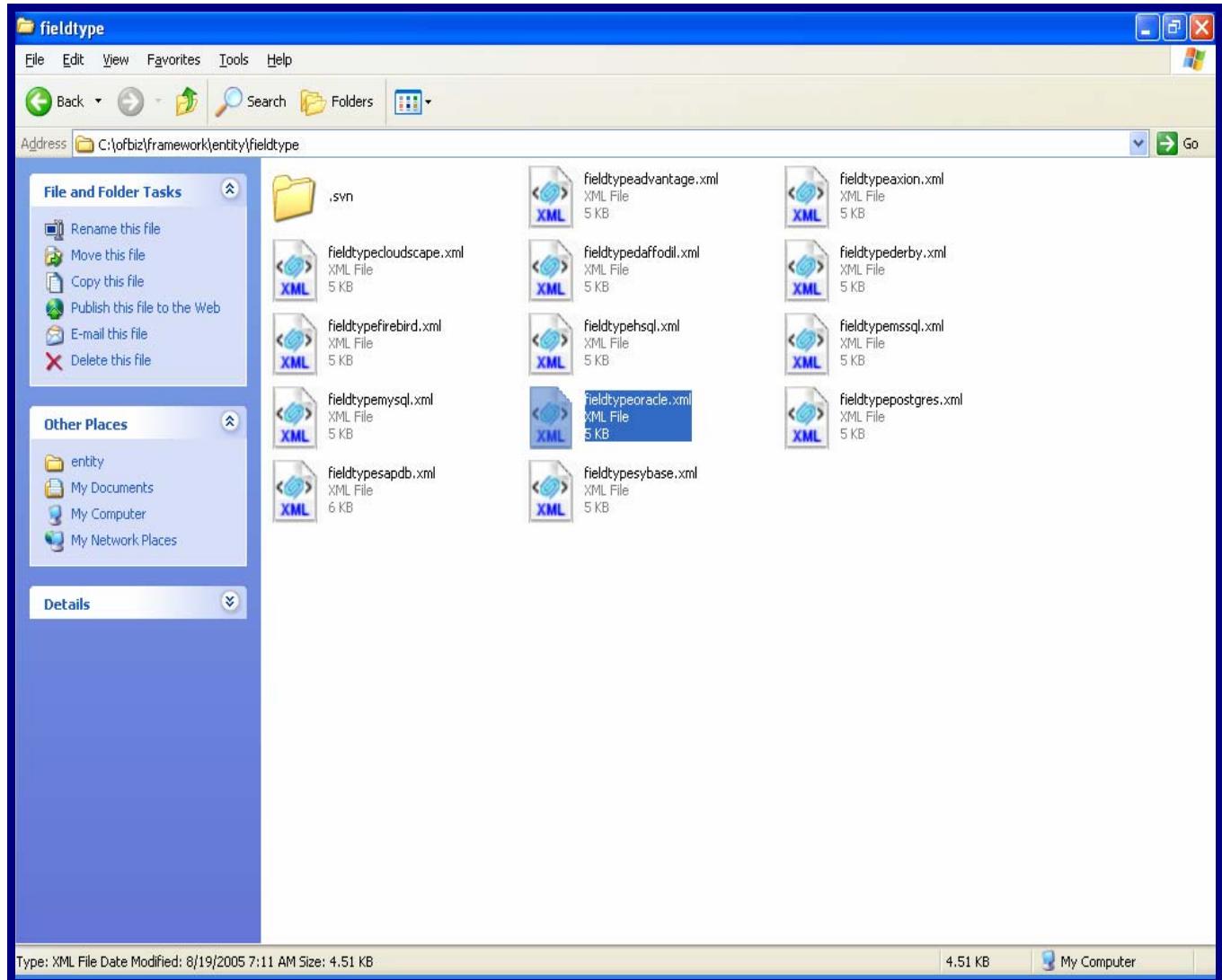
Figure 10

Note that each filed has a field-type. Field types might differ based on the type of the database. Thus, based on the database you are using “the default for OFBiz is the Derby database” you would decide what types to choose for your fields.

To know the different types for the database, you could follow the directory :

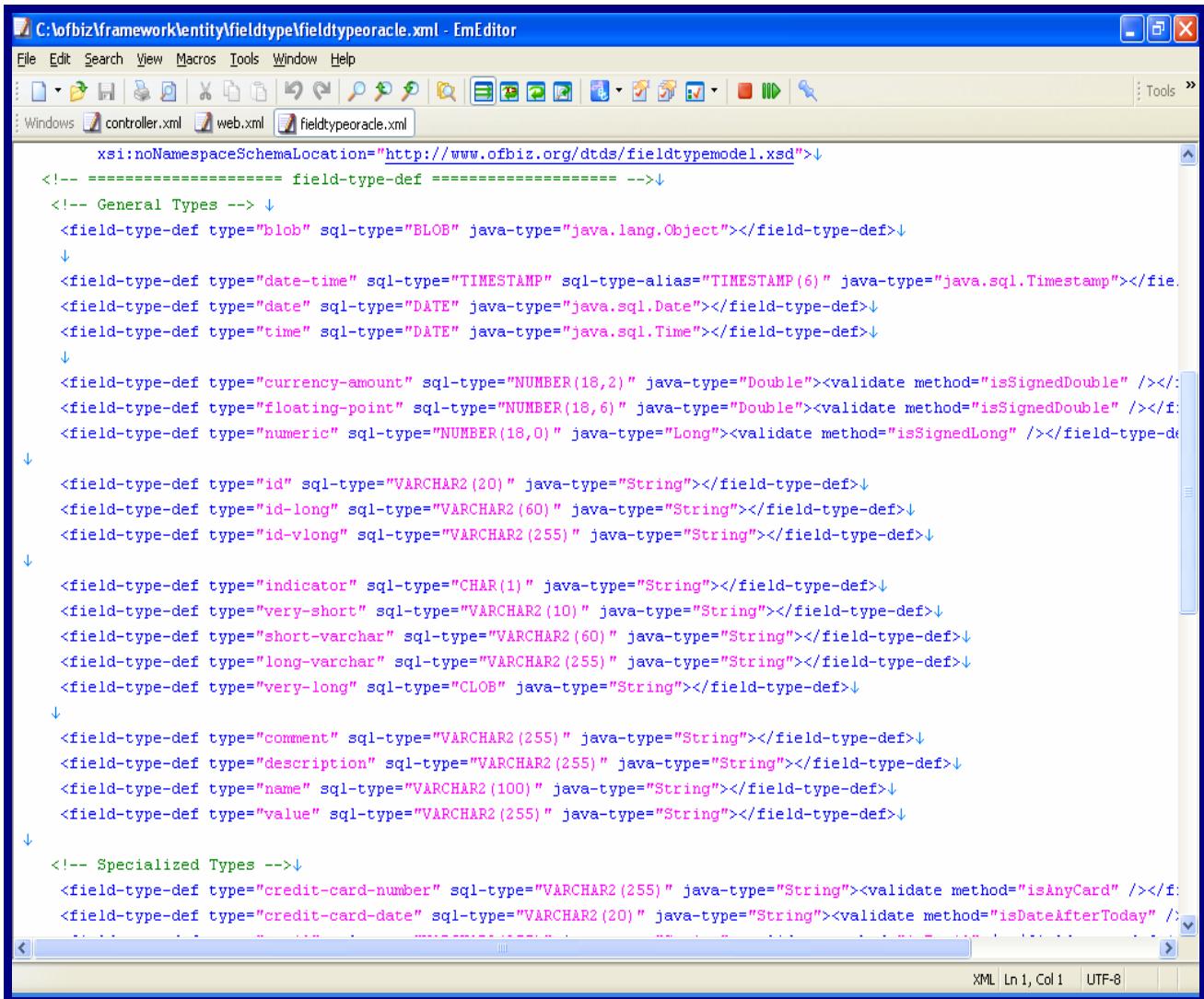
*C:\ofbiz\framework\entity\fieldtype*

Inside , you would see many files, each for a particular database.



**Figure 11**

Assuming you are using the Oracle database , you would check the fieldtypeoracle.xml file , as shown below.



The screenshot shows the EmEditor application window with the file 'fieldtypeoracle.xml' open. The code is an XML configuration for field types, primarily defining various SQL types and their corresponding Java representations. It includes sections for general types like blob, date-time, date, time, currency-amount, floating-point, numeric, and specialized types like id, indicator, and very-long. Validation methods like isSignedDouble, isSignedLong, isAnyCard, and isDateAfterToday are also defined. The XML uses xsd:namespace declarations and validate attributes to map database types to Java classes.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ===== field-type-def ===== -->
<!-- General Types -->
<field-type-def type="blob" sql-type="BLOB" java-type="java.lang.Object"></field-type-def>
<!-- Date-Time Types -->
<field-type-def type="date-time" sql-type="TIMESTAMP" sql-type-alias="TIMESTAMP(6)" java-type="java.sql.Timestamp"></field-type-def>
<field-type-def type="date" sql-type="DATE" java-type="java.sql.Date"></field-type-def>
<field-type-def type="time" sql-type="TIME" java-type="java.sql.Time"></field-type-def>
<!-- Numerical Types -->
<field-type-def type="currency-amount" sql-type="NUMBER(18,2)" java-type="Double"><validate method="isSignedDouble" /></field-type-def>
<field-type-def type="floating-point" sql-type="NUMBER(18,6)" java-type="Double"><validate method="isSignedDouble" /></field-type-def>
<field-type-def type="numeric" sql-type="NUMBER(18,0)" java-type="Long"><validate method="isSignedLong" /></field-type-def>
<!-- String Types -->
<field-type-def type="id" sql-type="VARCHAR2(20)" java-type="String"></field-type-def>
<field-type-def type="id-long" sql-type="VARCHAR2(60)" java-type="String"></field-type-def>
<field-type-def type="id-vlong" sql-type="VARCHAR2(255)" java-type="String"></field-type-def>
<field-type-def type="indicator" sql-type="CHAR(1)" java-type="String"></field-type-def>
<field-type-def type="very-short" sql-type="VARCHAR2(10)" java-type="String"></field-type-def>
<field-type-def type="short-varchar" sql-type="VARCHAR2(60)" java-type="String"></field-type-def>
<field-type-def type="long-varchar" sql-type="VARCHAR2(255)" java-type="String"></field-type-def>
<field-type-def type="very-long" sql-type="CLOB" java-type="String"></field-type-def>
<!-- Specialized Types -->
<field-type-def type="comment" sql-type="VARCHAR2(255)" java-type="String"></field-type-def>
<field-type-def type="description" sql-type="VARCHAR2(255)" java-type="String"></field-type-def>
<field-type-def type="name" sql-type="VARCHAR2(100)" java-type="String"></field-type-def>
<field-type-def type="value" sql-type="VARCHAR2(255)" java-type="String"></field-type-def>
```

Figure 12

Notice that you are not restricted to these type, you can add your own ones, all what you need to do is to add a new <field-type-def> tag.

For more information about the entity model and entity definition, you can visit : [Entity Model](#) .

## servicedef :

It defines the services used in the business “logic” layer, it contains the services.xml file , which define the services. In our case , the Accounting application has many sub-applications in it, one of which is the Agreement as mentioned earlier. Thus, the services file is services\_agreement.xml

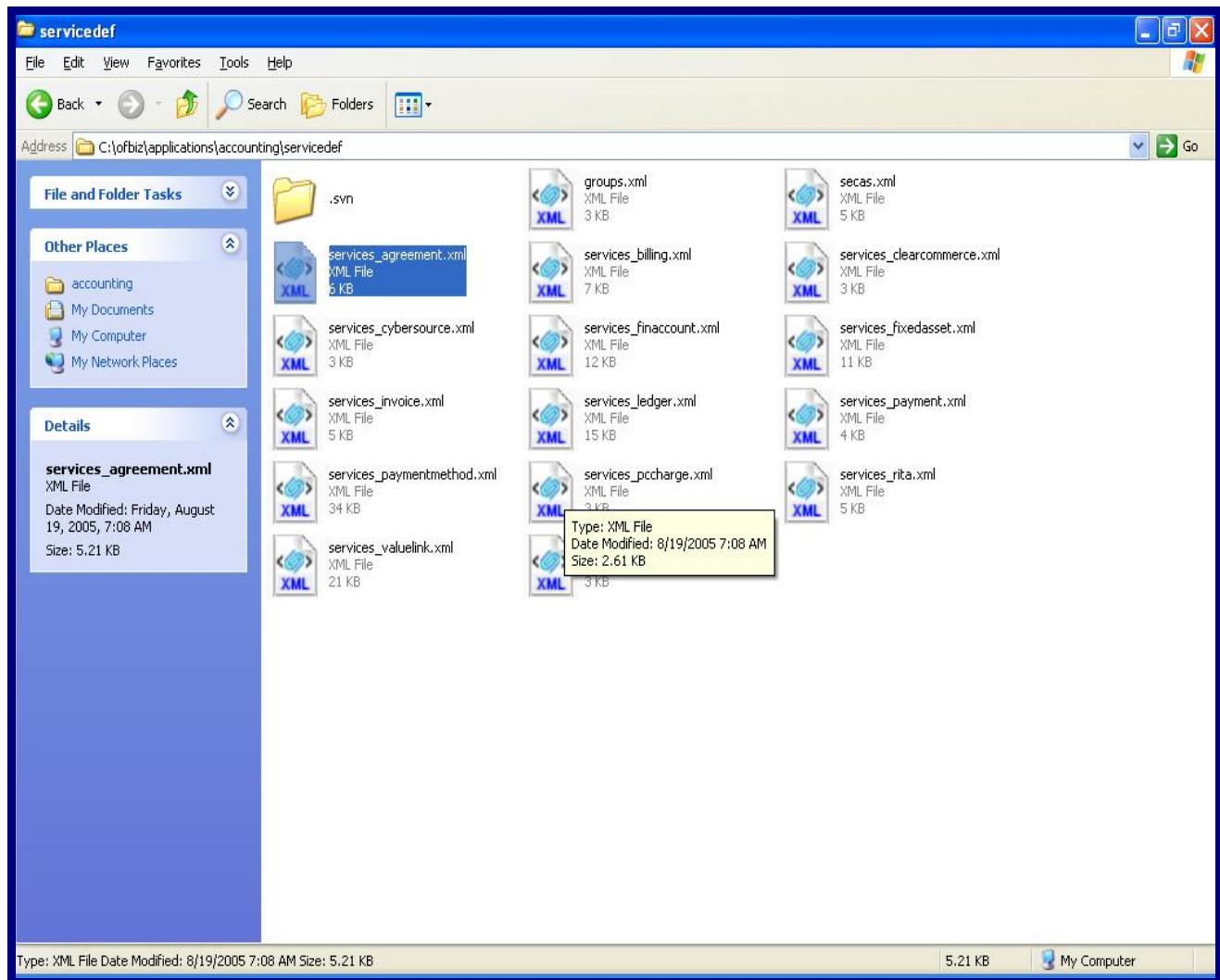
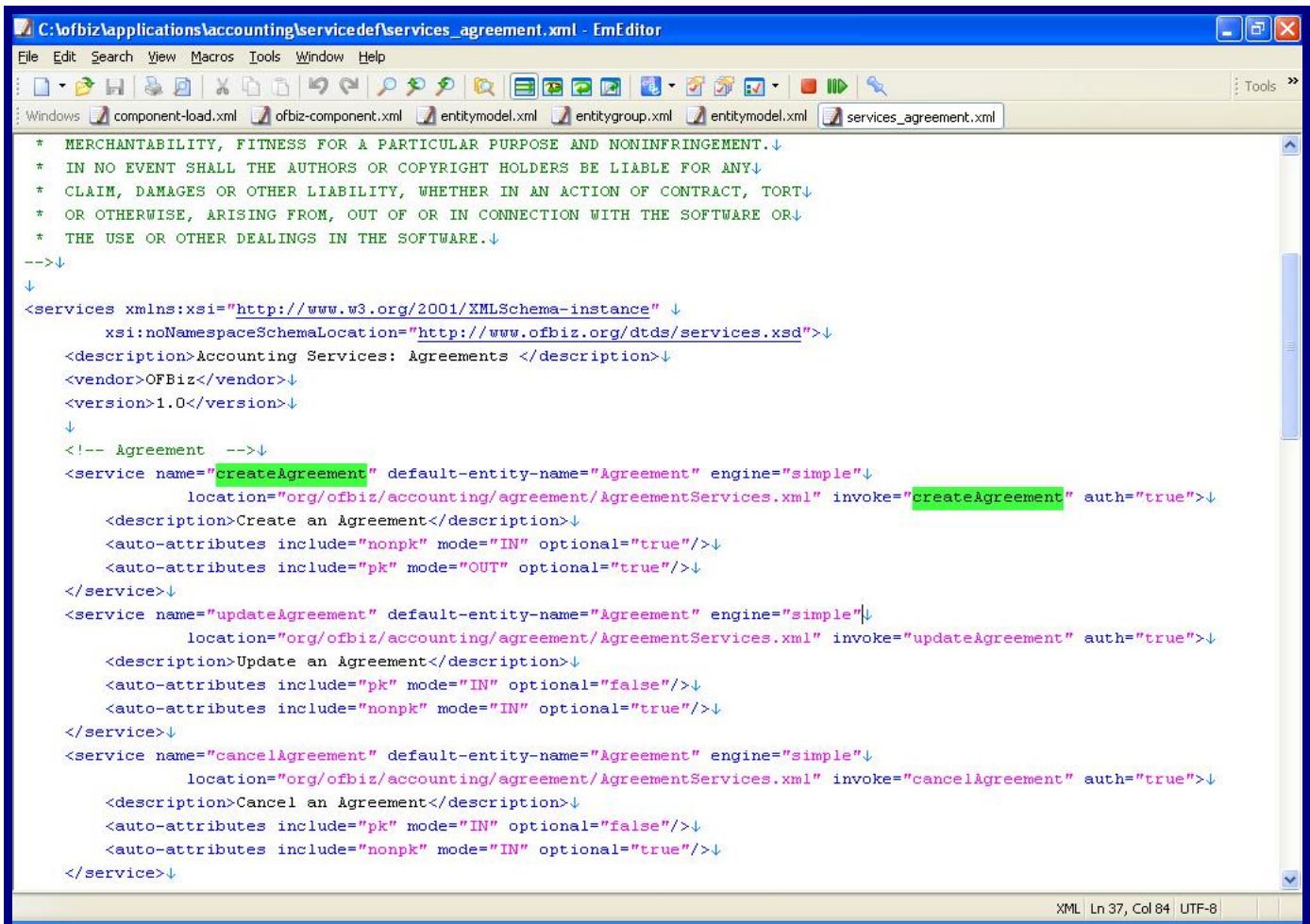


Figure 13

## Important Note :

Whenever you add a new service file , like the service\_agreement.xml or any service definition file, you need to include a reference to it in the ofbiz-component.xml file “Figure 7”

Having a look inside this file, service\_agreement.xml , we would see the definition of all the services used by the agreement application.



```
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.↓
* IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY↓
* CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT↓
* OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR↓
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.↓
-->↓
↓
<services xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↓
    xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/services.xsd">↓
    <description>Accounting Services: Agreements </description>↓
    <vendor>OFBiz</vendor>↓
    <version>1.0</version>↓
    ↓
    <!-- Agreement -->↓
    <service name="createAgreement" default-entity-name="Agreement" engine="simple">↓
        location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="createAgreement" auth="true">↓
            <description>Create an Agreement</description>↓
            <auto-attributes include="nonpk" mode="IN" optional="true"/>↓
            <auto-attributes include="pk" mode="OUT" optional="true"/>↓
    </service>↓
    <service name="updateAgreement" default-entity-name="Agreement" engine="simple">↓
        location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="updateAgreement" auth="true">↓
            <description>Update an Agreement</description>↓
            <auto-attributes include="pk" mode="IN" optional="false"/>↓
            <auto-attributes include="nonpk" mode="IN" optional="true"/>↓
    </service>↓
    <service name="cancelAgreement" default-entity-name="Agreement" engine="simple">↓
        location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="cancelAgreement" auth="true">↓
            <description>Cancel an Agreement</description>↓
            <auto-attributes include="pk" mode="IN" optional="false"/>↓
            <auto-attributes include="nonpk" mode="IN" optional="true"/>↓
    </service>↓

```

Figure 14

After defining the services, we need to implement them. Normally, services are implemented using the OFBiz mini-language. However, if the service cannot be implemented with “xml” , we can use java to implement it.

script : contains the implementation for the services using the OFBiz mini-language, and it contains some scripts.

Inside this folder, we will find many subfolders containing the service implementation for the different accounting subapplications. We are interested in the Agreement subfolder :

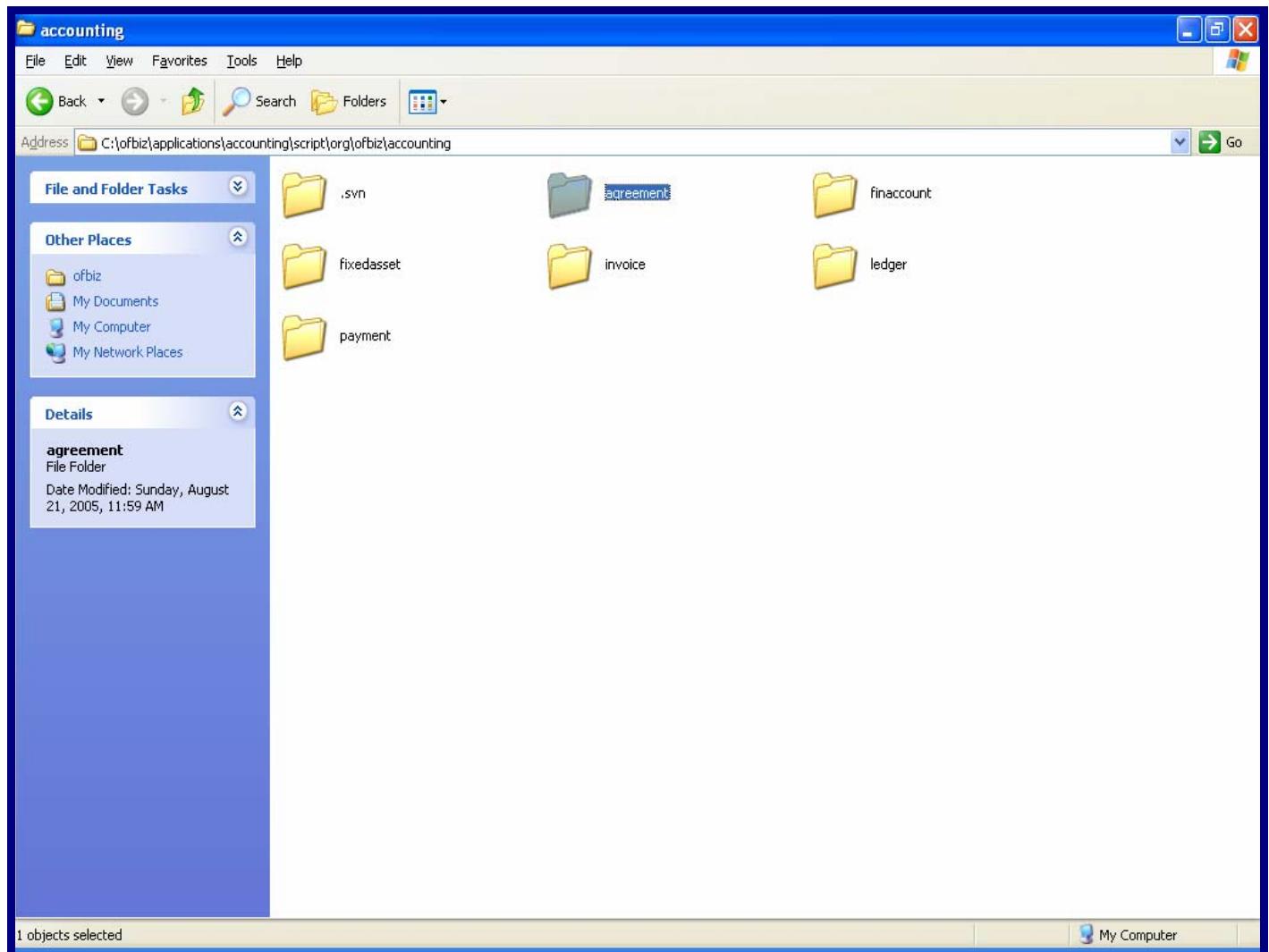
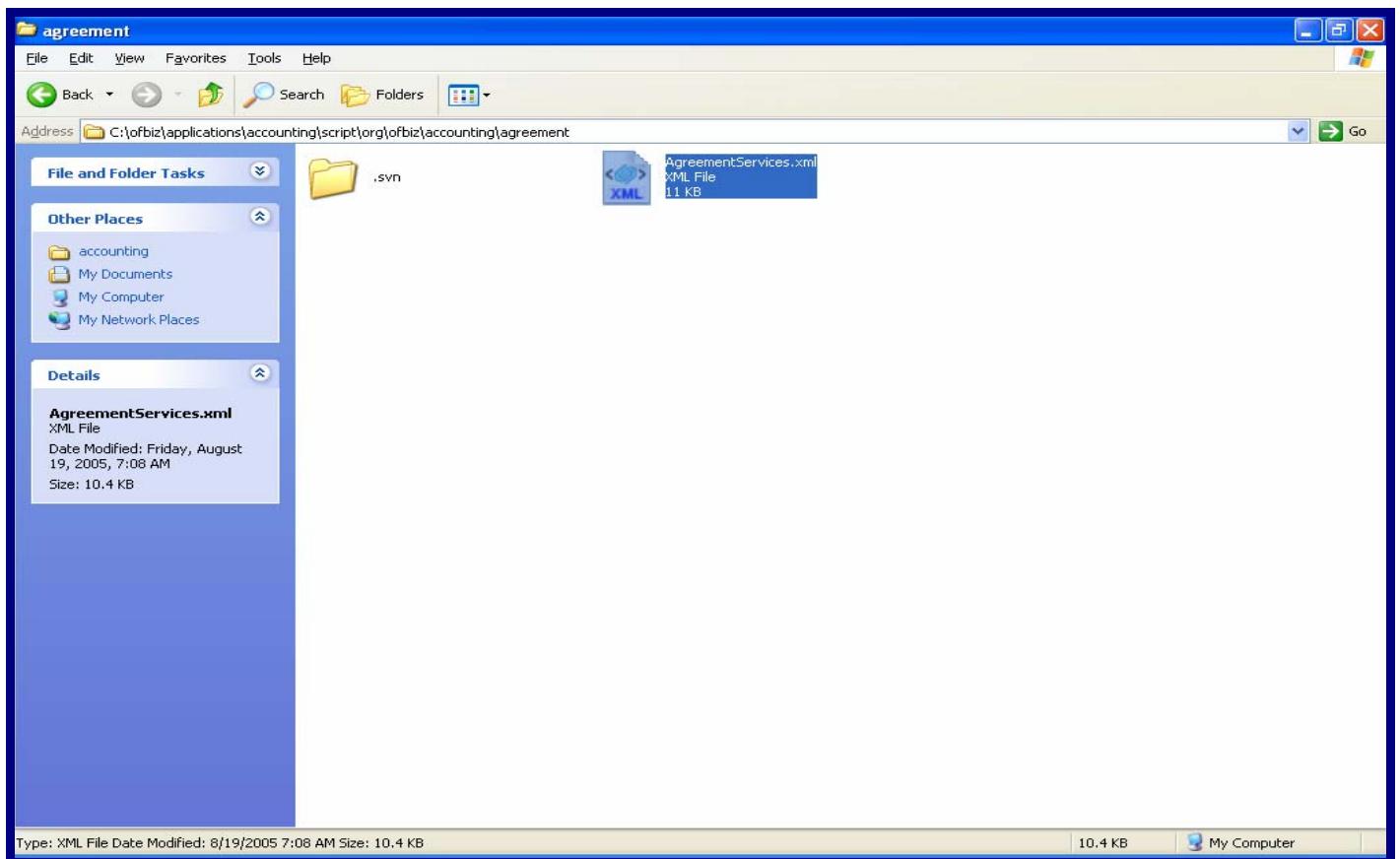


Figure 15



**Figure 16**

and having a look into this file :

for example. It is implementing the createAgreement service that was defined in the servicedef , as was shown in Figure 14.

```

<!-- ===== -->
<!-- Agreement Services -->
<!-- ===== -->
<!-- create a new Agreement -->
<simple-method method-name="createAgreement" short-description="Create an Agreement">
    <check-permission permission="ACCOUNTING" action="_CREATE">
        <alt-permission permission="ACCOUNTING_ROLE" action="_CREATE"/>
        <fail-message message="Security Error: to run CreateAgreement you must have the ACCOUNTING_CREATE or ACCOUNTING_ROLE permission" />
    </check-permission>
    <check-errors/>

    <!-- create new entity and create all the fields -->
    <make-value value-name="newEntity" entity-name="Agreement"/>
    <set-nonpk-fields map-name="parameters" value-name="newEntity"/>

    <!-- create a non existing ID -->
    <sequenced-id-to-env sequence-name="agreement" env-name="agreementId"/>
    <env-to-field env-name="agreementId" map-name="newEntity"/>
    <field-to-result field-name="agreementId" result-name="agreementId"/>

    <if-empty field-name="fromDate" map-name="newEntity">
        <now-timestamp-to-env env-name="nowTimestamp"/>
        <env-to-field env-name="nowTimestamp" field-name="fromDate" map-name="newEntity"/>
    </if-empty>

    <!-- finally create the record (should not exist already) -->
    <create-value value-name="newEntity"/>
    <check-errors/>
</simple-method>

```

Figure 17

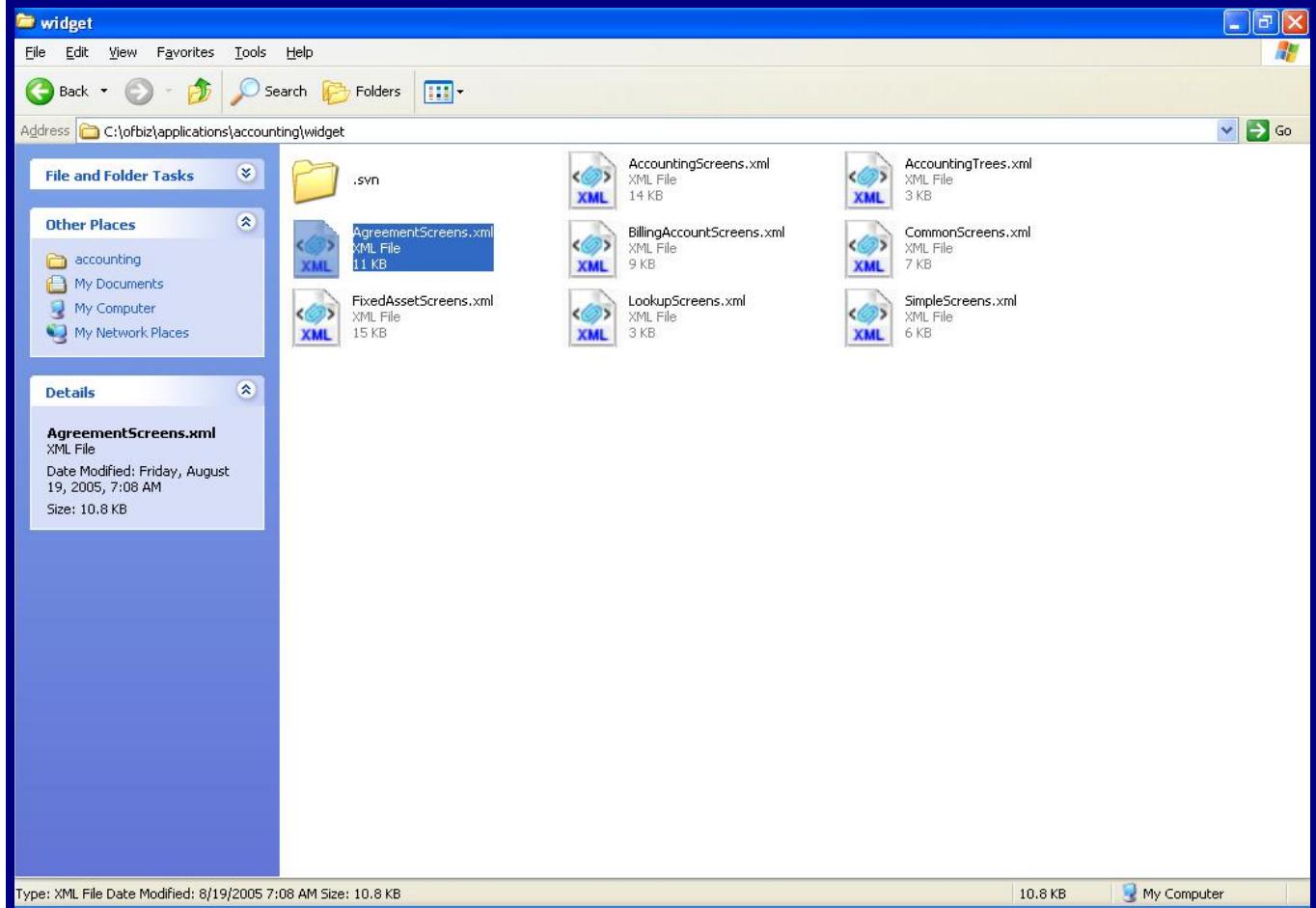
For more Information on how to define services , you can visit :  
[Service Engine Guide.](#)

src : contains the java source files for the services that were implemented with java.

widget : recently the OFBiz presentation layer pages are defined as "Screens". This directory holds "widgets" for the user interface screens. OFBiz allows the user interface design to be created as "generic screens" rather than just web pages, so they could be reused eventually for some other platforms. The widgets/ directory's contents mirror those of the webapp [1]\* So, each application will have its own screens , as so the Agreement application does. Inside this folder, we would find the AgreementScreens.xml file that defines the Agreement screens.

In the Figure below, Figure 18 , we would see the AgreementScreens.xml file among the other applications screens files.

In Figure 19, we will see the findAgreementScreen that allows to search for a particular agreement.



**Figure 18**

Screens are divided into two parts : actions and widgets. Actions are responsible for data retrieval while widgets are responsible for data display.

The screenshot shows the EmEditor application window with the file 'AgreementScreens.xml' open. The code is an XML configuration for a screen named 'FindAgreement'. It defines sections for actions, widgets, and a main-decorator. The main-decorator includes a body section with containers for a header label and links to edit or list agreements. The XML code is color-coded for syntax highlighting.

```
</decorator-screen>↓
</widgets>↓
</section>↓
↓
<!-- list all assets in a tabular format -->↓
<screen name="FindAgreement">↓
<section>↓
    <actions>↓
        <set field="title" value="Find Agreements"/>↓
        <set field="headerItem" value="agreement"/>↓
        <set field="viewIndex" from-field="parameters.VIEW_INDEX" type="Integer"/>↓
        <set field="viewSize" from-field="parameters.VIEW_SIZE" type="Integer" default-value="50"/>↓
    ↓
    </actions>↓
    <widgets>↓
        <decorator-screen name="main-decorator" location="component://accounting/widget/CommonScreens.xml">↓
            <decorator-section name="body">↓
                <container>↓
                    <label style="head1">${uiLabelMap.AccountingAgreements}</label>↓
                </container>↓
                <container>↓
                    <link target="EditAgreement" text="[ ${uiLabelMap.AccountingNewAgreement} ]" style="buttoncontext"/>↓
                </container>↓
                <include-form name="FindAgreements" location="component://accounting/webapp/accounting/agreement/AgreementListForm.html"/>↓
                <include-form name="ListAgreements" location="component://accounting/webapp/accounting/agreement/AgreementListForm.html"/>↓
            </decorator-section>↓
        </decorator-screen>↓
    </widgets>↓
</section>↓
</screen>↓
```

Figure 19

webapp : contains web application pages and forms . With OFBiz, pages are divided into smaller pieces which are re-combined to create the final product. Thus, many pages can share common elements such as page headers, sidebars, and navigation bars. This is called the "decorator pattern." There is a further separation of the activities of a page into "actions," such as getting data from a database, and "presentation," the display of that data to the visitor. [I]\*

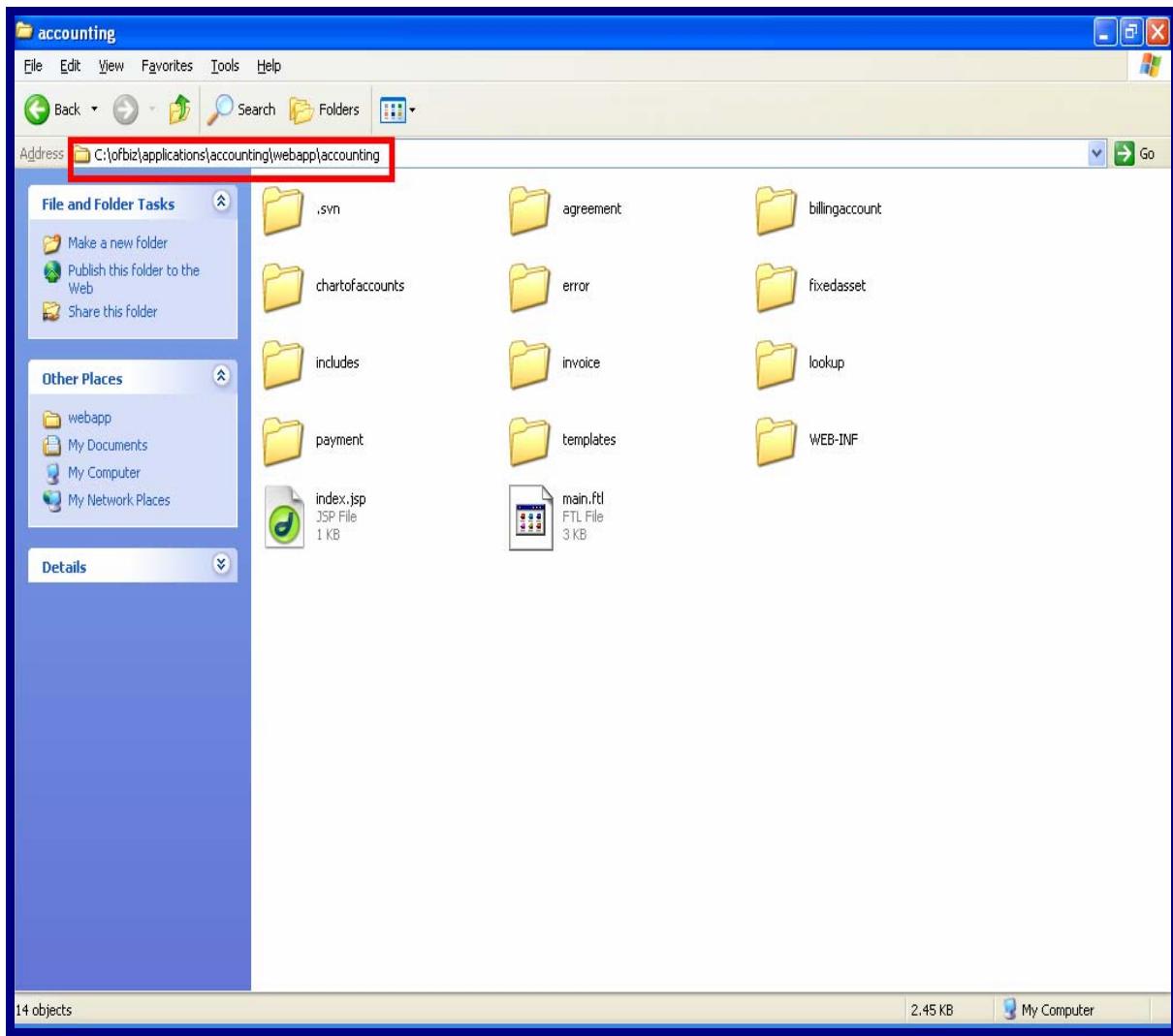


Figure 20

The basic files/folders in our application are :

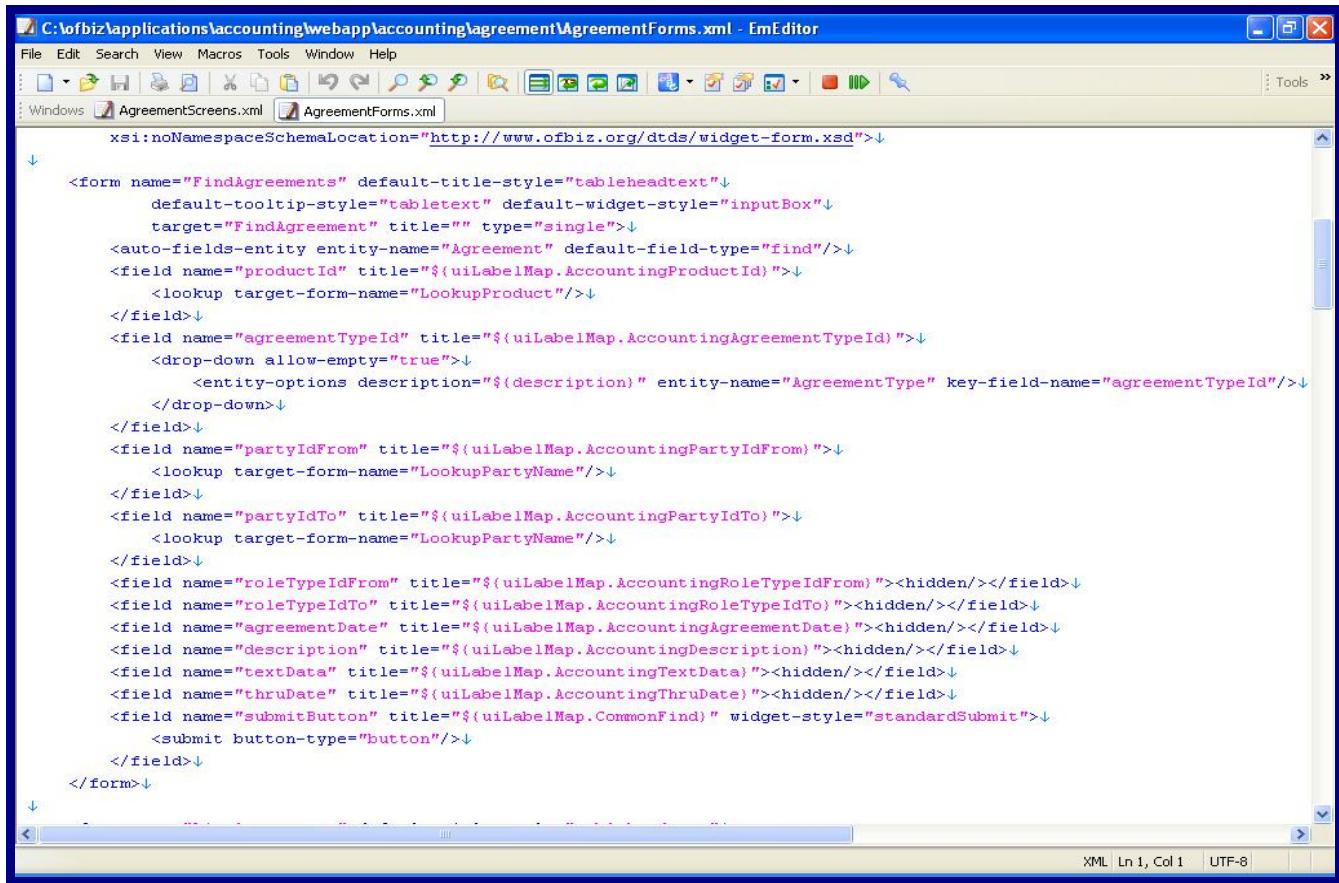
Index.jsp : used to redirect the controller to the main page.

main.ftl : The main page for the accounting application, written with FreeMarker Template Language (FTL) .

includes folder : contains the appheader.ftl file that is common for all the accounting application .It can also contain some other ftl files if needed to be used by the application.

error folder : contains the error pages to be displayed when a particular error occurs.

Agreement folder : contains the agreement forms that are used /called by the agreement screens or agreement ftl files. Here is the findAgreement form as an example



```
xsi:namespaceSchemaLocation="http://www.ofbiz.org/dtds/widget-form.xsd">↓
<form name="FindAgreements" default-title-style="tableheadtext"↓
    default-tooltip-style="tabletext" default-widget-style="inputBox"↓
    target="FindAgreement" title="" type="single">↓
<auto-fields-entity entity-name="Agreement" default-field-type="find"/>↓
<field name="productId" title="${uiLabelMap.AccountingProductId}">↓
    <lookup target-form-name="LookupProduct"/>↓
</field>↓
<field name="agreementTypeId" title="${uiLabelMap.AccountingAgreementTypeId}">↓
    <drop-down allow-empty="true">↓
        <entity-options description="${description}" entity-name="AgreementType" key-field-name="agreementTypeId"/>↓
    </drop-down>↓
</field>↓
<field name="partyIdFrom" title="${uiLabelMap.AccountingPartyIdFrom}">↓
    <lookup target-form-name="LookupPartyName"/>↓
</field>↓
<field name="partyIdTo" title="${uiLabelMap.AccountingPartyIdTo}">↓
    <lookup target-form-name="LookupPartyName"/>↓
</field>↓
<field name="roleTypeIdFrom" title="${uiLabelMap.AccountingRoleTypeIdFrom}"><hidden/></field>↓
<field name="roleTypeIdTo" title="${uiLabelMap.AccountingRoleTypeIdTo}"><hidden/></field>↓
<field name="agreementDate" title="${uiLabelMap.AccountingAgreementDate}"><hidden/></field>↓
<field name="description" title="${uiLabelMap.AccountingDescription}"><hidden/></field>↓
<field name="textData" title="${uiLabelMap.AccountingTextData}"><hidden/></field>↓
<field name="thruDate" title="${uiLabelMap.AccountingThruDate}"><hidden/></field>↓
<field name="submitButton" title="${uiLabelMap.CommonFind}" widget-style="standardSubmit">↓
    <submit button-type="button"/>↓
</field>↓
</form>↓

```

Figure 21

WEB-INF : the most important directory , it contains very important files and directories.

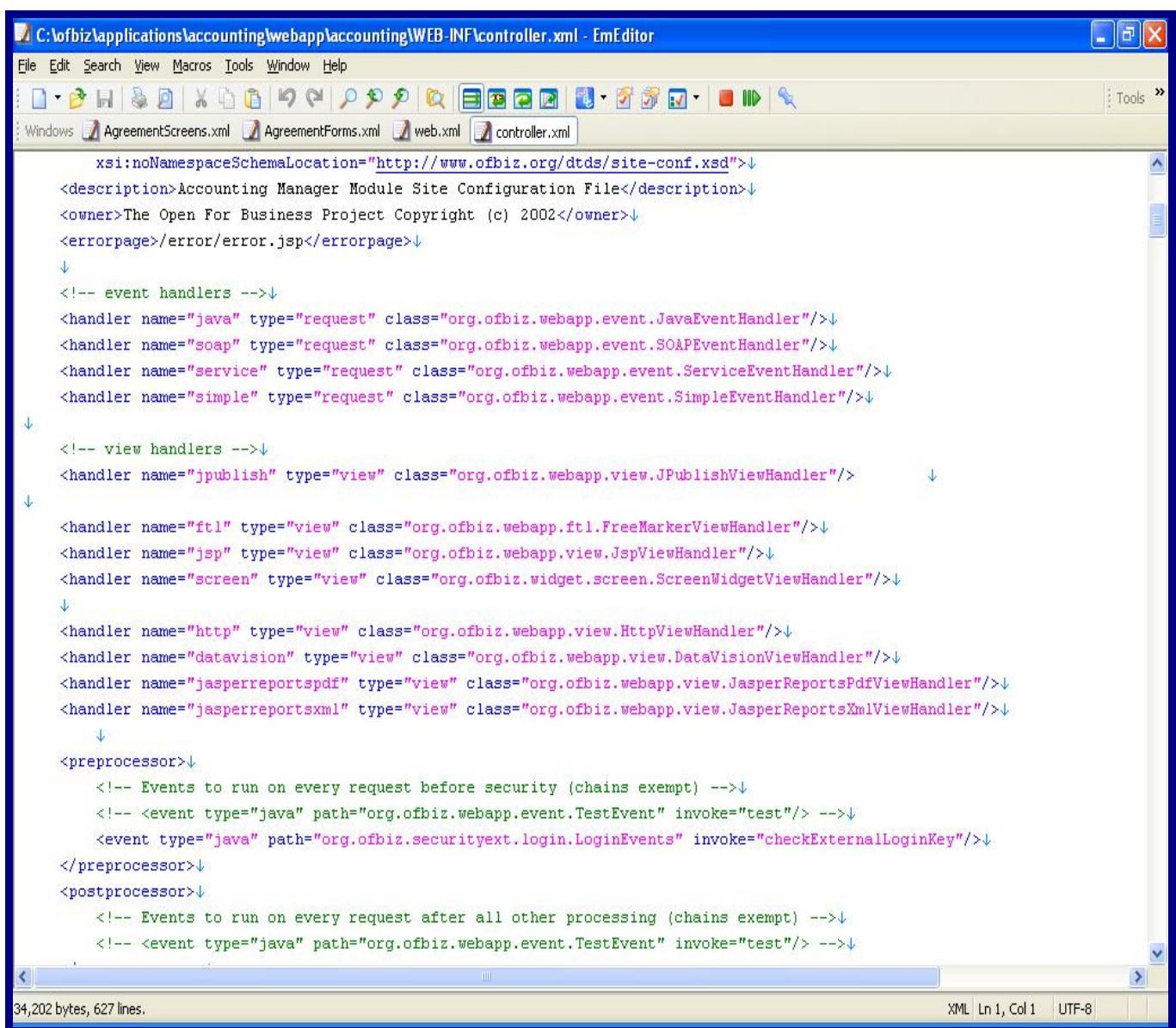
-actions folder : has beanshell scripts that are used to process and gather data from the database .

-web.xml file : discussed earlier.

-controller.xml : Responsible for controlling the coming request .Any request to the application, whther it is a screen request, service request, event..etc, it should be passed through the controller.

## Inside the controller.xml file :

1) Defining the different handlers for different types of events.



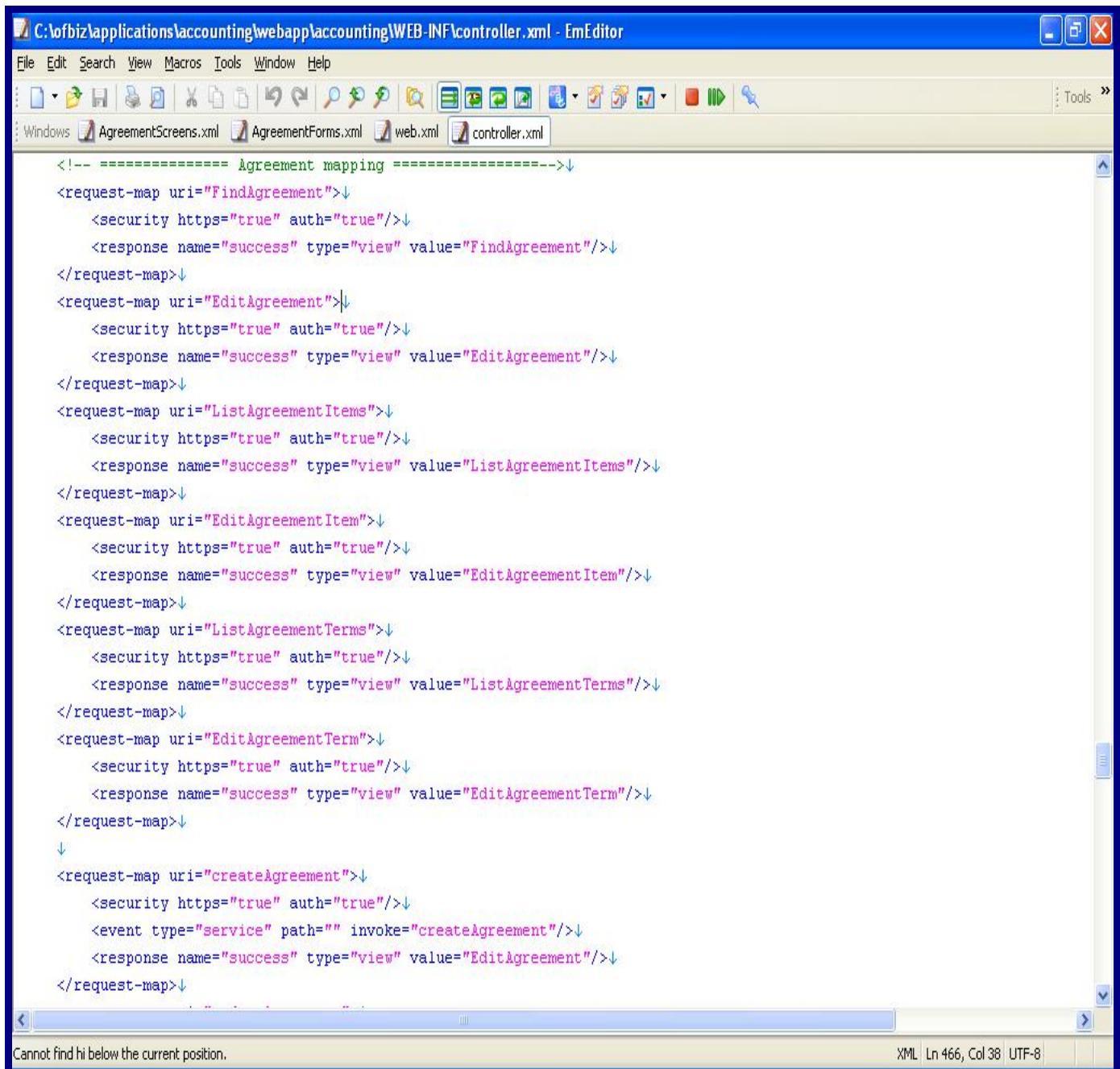
The screenshot shows the EmEditor XML Editor window with the file 'controller.xml' open. The window has a toolbar at the top with various icons for file operations like Open, Save, Find, and Copy. Below the toolbar is a menu bar with File, Edit, Search, View, Macros, Tools, Window, and Help. The main area displays the XML code for the controller configuration. The code defines various event handlers for request, service, simple, view, and other types. It also includes sections for preprocessors and postprocessors. The XML uses the 'xsi:noNamespaceSchemaLocation' attribute to point to 'http://www.ofbiz.org/dtds/site-conf.xsd'. The code is color-coded for readability, with tags in blue and attributes in green. The status bar at the bottom indicates '34,202 bytes, 627 lines.' and shows the XML tab is selected.

```
xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/site-conf.xsd">>↓
<description>Accounting Manager Module Site Configuration File</description>↓
<owner>The Open For Business Project Copyright (c) 2002</owner>↓
<errorpage>/error/error.jsp</errorpage>↓
↓
<!-- event handlers -->↓
<handler name="java" type="request" class="org.ofbiz.webapp.event.JavaEventHandler"/>↓
<handler name="soap" type="request" class="org.ofbiz.webapp.event.SOAPEventHandler"/>↓
<handler name="service" type="request" class="org.ofbiz.webapp.event.ServiceEventHandler"/>↓
<handler name="simple" type="request" class="org.ofbiz.webapp.event.SimpleEventHandler"/>↓
↓
<!-- view handlers -->↓
<handler name="jpublish" type="view" class="org.ofbiz.webapp.view.JPublishViewHandler"/> ↓
↓
<handler name="ftl" type="view" class="org.ofbiz.webapp.ftl.FreeMarkerViewHandler"/>↓
<handler name="jsp" type="view" class="org.ofbiz.webapp.view.JspViewHandler"/>↓
<handler name="screen" type="view" class="org.ofbiz.widget.screen.ScreenWidgetViewHandler"/>↓
↓
<handler name="http" type="view" class="org.ofbiz.webapp.view.HttpViewHandler"/>↓
<handler name="datavision" type="view" class="org.ofbiz.webapp.view.DataVisionViewHandler"/>↓
<handler name="jasperreportspdf" type="view" class="org.ofbiz.webapp.view.JasperReportsPdfViewHandler"/>↓
<handler name="jasperreportsxml" type="view" class="org.ofbiz.webapp.view.JasperReportsXmlViewHandler"/>↓
↓
<preprocessor>↓
  <!-- Events to run on every request before security (chains exempt) -->↓
  <!-- <event type="java" path="org.ofbiz.webapp.event.TestEvent" invoke="test"/> -->↓
  <event type="java" path="org.ofbiz.securityext.login.LoginEvents" invoke="checkExternalLoginKey"/>↓
</preprocessor>↓
<postprocessor>↓
  <!-- Events to run on every request after all other processing (chains exempt) -->↓
  <!-- <event type="java" path="org.ofbiz.webapp.event.TestEvent" invoke="test"/> -->↓

```

Figure 22

2) Defining the request mapping for the application, it can be a screen “view” request or a service request , as example : FindAgreement and createAgreement.



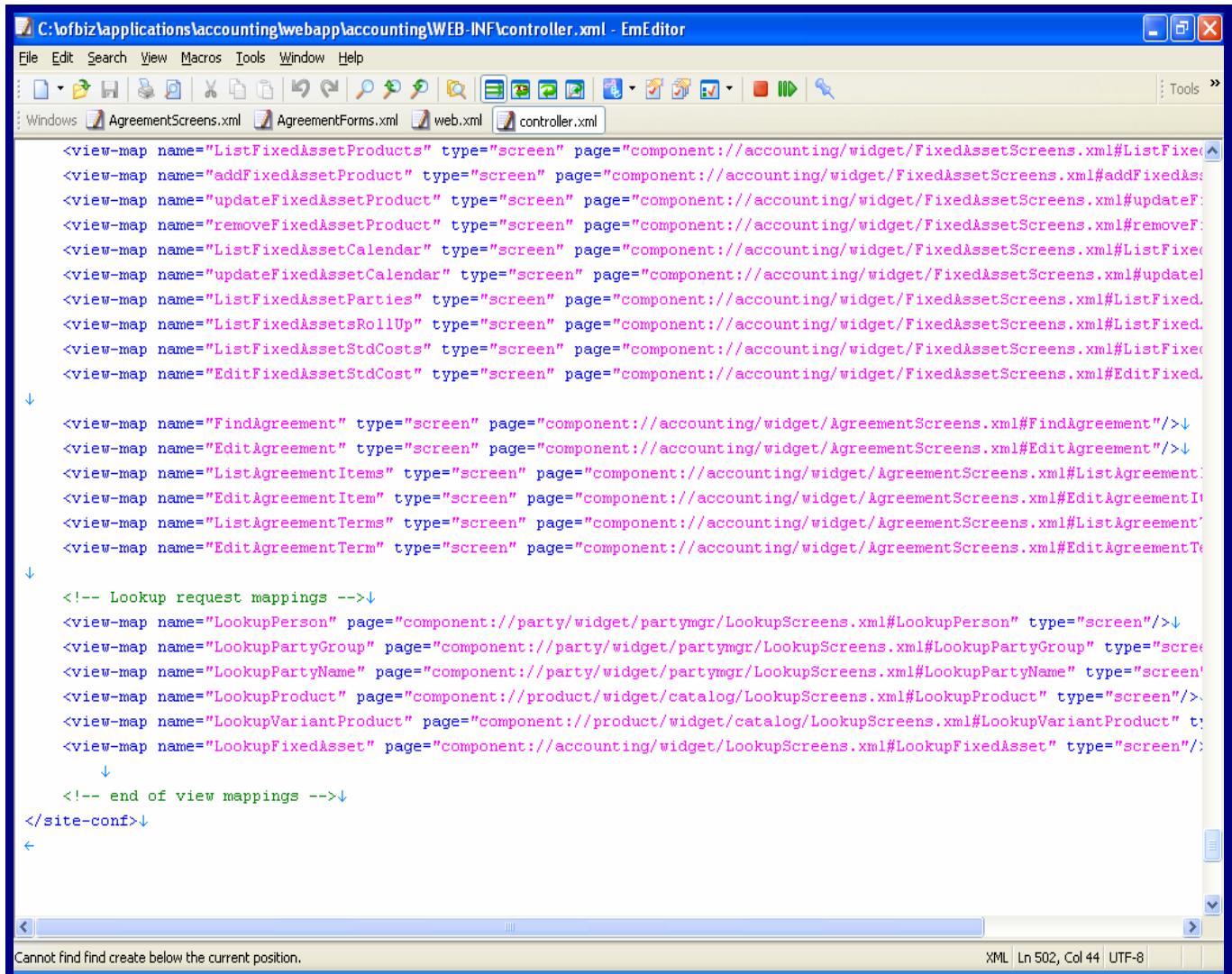
The screenshot shows the EmEditor XML editor window with the file 'controller.xml' open. The window title is 'C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml - EmEditor'. The menu bar includes File, Edit, Search, View, Macros, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Copy, Paste, and Find. The status bar at the bottom right shows 'XML | Ln 466, Col 38 | UTF-8'. The code in the editor is an XML configuration for request mappings:

```
<!-- ===== Agreement mapping =====-->
<request-map uri="FindAgreement">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="FindAgreement"/>↓
</request-map>↓
<request-map uri="EditAgreement">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
<request-map uri="ListAgreementItems">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementItems"/>↓
</request-map>↓
<request-map uri="EditAgreementItem">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementItem"/>↓
</request-map>↓
<request-map uri="ListAgreementTerms">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementTerms"/>↓
</request-map>↓
<request-map uri="EditAgreementTerm">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementTerm"/>↓
</request-map>↓
↓
<request-map uri="createAgreement">↓
    <security https="true" auth="true"/>↓
    <event type="service" path="" invoke="createAgreement"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
```

The status bar at the bottom left says 'Cannot find hi below the current position.'

Figure 23

### 3) The controller tells where to look for the requested screen or service .



The screenshot shows the EmEditor XML editor window with the file 'controller.xml' open. The window title is 'C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml - EmEditor'. The menu bar includes File, Edit, Search, View, Macros, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Find, Copy, Paste, etc. Below the toolbar is a tab bar with 'Windows', 'AgreementScreens.xml', 'AgreementForms.xml', 'web.xml', and 'controller.xml'. The main area contains the XML code for the controller configuration.

```
<view-map name="ListFixedAssetProducts" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#ListFixedAssetProducts" />
<view-map name="addFixedAssetProduct" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#addFixedAssetProduct" />
<view-map name="updateFixedAssetProduct" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#updateFixedAssetProduct" />
<view-map name="removeFixedAssetProduct" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#removeFixedAssetProduct" />
<view-map name="ListFixedAssetCalendar" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#ListFixedAssetCalendar" />
<view-map name="updateFixedAssetCalendar" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#updateFixedAssetCalendar" />
<view-map name="ListFixedAssetParties" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#ListFixedAssetParties" />
<view-map name="ListFixedAssetsRollUp" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#ListFixedAssetsRollUp" />
<view-map name="ListFixedAssetStdCosts" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#ListFixedAssetStdCosts" />
<view-map name="EditFixedAssetStdCost" type="screen" page="component://accounting/widget/FixedAssetScreens.xml#EditFixedAssetStdCost" />
↓
<view-map name="FindAgreement" type="screen" page="component://accounting/widget/AgreementScreens.xml#FindAgreement" />↓
<view-map name="EditAgreement" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreement" />↓
<view-map name="ListAgreementItems" type="screen" page="component://accounting/widget/AgreementScreens.xml#ListAgreementItems" />
<view-map name="EditAgreementItem" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreementItem" />
<view-map name="ListAgreementTerms" type="screen" page="component://accounting/widget/AgreementScreens.xml#ListAgreementTerms" />
<view-map name="EditAgreementTerm" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreementTerm" />
↓
<!-- Lookup request mappings --&gt;↓
&lt;view-map name="LookupPerson" page="component://party/widget/partymgr/LookupScreens.xml#LookupPerson" type="screen" /&gt;
&lt;view-map name="LookupPartyGroup" page="component://party/widget/partymgr/LookupScreens.xml#LookupPartyGroup" type="screen" /&gt;
&lt;view-map name="LookupPartyName" page="component://party/widget/partymgr/LookupScreens.xml#LookupPartyName" type="screen" /&gt;
&lt;view-map name="LookupProduct" page="component://product/widget/catalog/LookupScreens.xml#LookupProduct" type="screen" /&gt;
&lt;view-map name="LookupVariantProduct" page="component://product/widget/catalog/LookupScreens.xml#LookupVariantProduct" type="screen" /&gt;
&lt;view-map name="LookupFixedAsset" page="component://accounting/widget/LookupScreens.xml#LookupFixedAsset" type="screen" /&gt;
↓
--&gt;<!-- end of view mappings --&gt;↓
&lt;/site-conf&gt;↓
←</pre>

Cannot find find create below the current position.      XML | Ln 502, Col 44 | UTF-8


```

Figure 24

## Demo :

- 1) Double click on the startofbiz.bat file in the directory C:/ofbiz or :

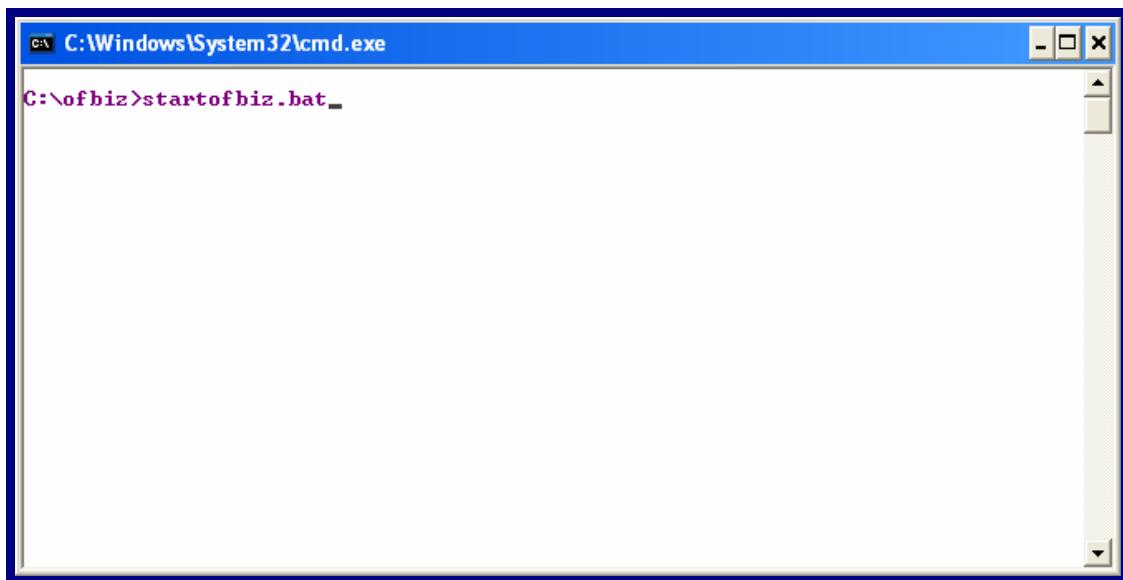


Figure 25

- 2) Wait until the OFBiz runs fully, then in the browser type the following :

<https://localhost:8443/accounting/control/main>

- 3) A user name and a password are required : the defaults are

User : admin

Password : ofbiz

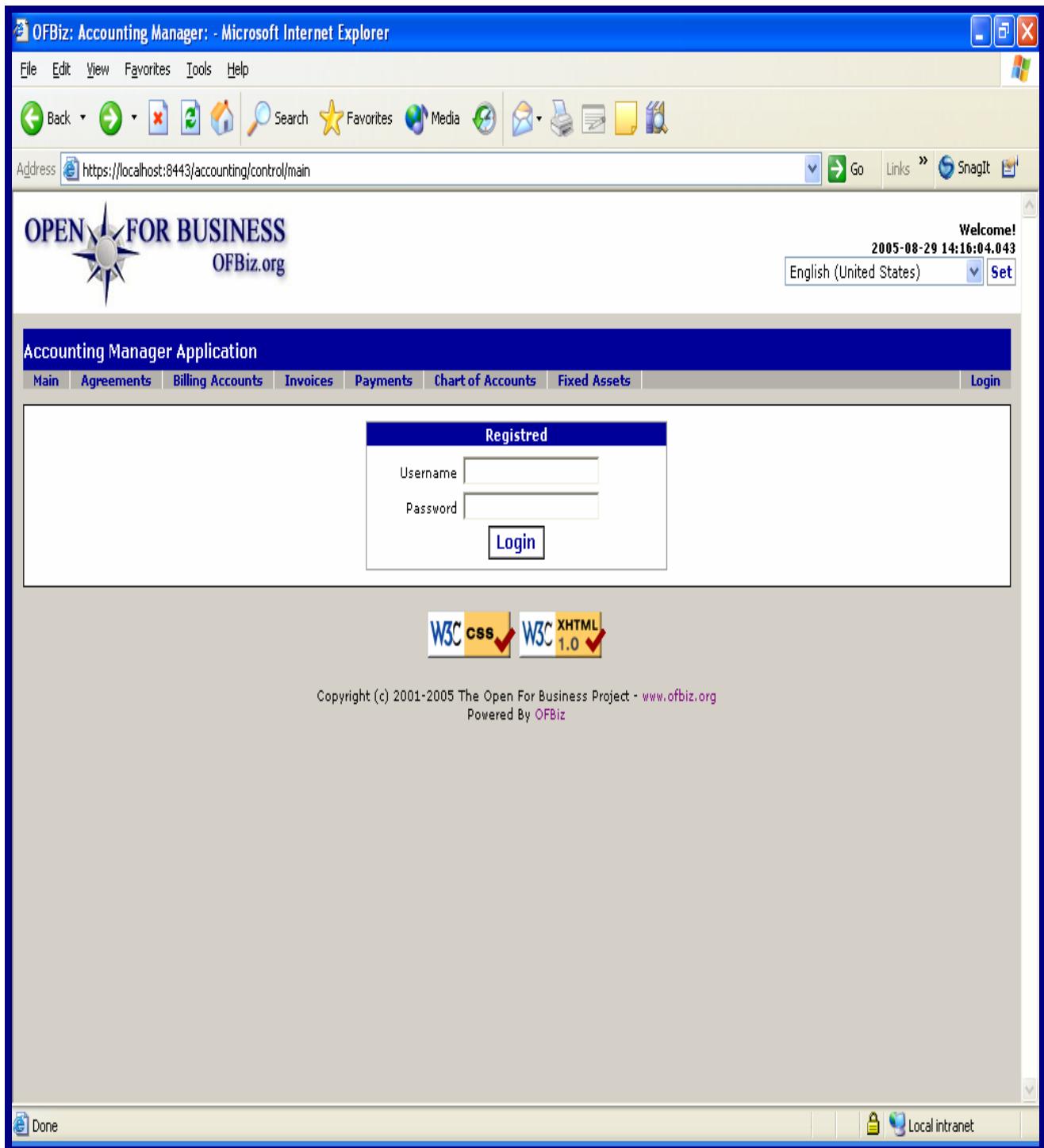


Figure 26

Now we are inside the application , the first page will be the main page . It is obvious! The requested page is the main page, and it is requested from the controller.

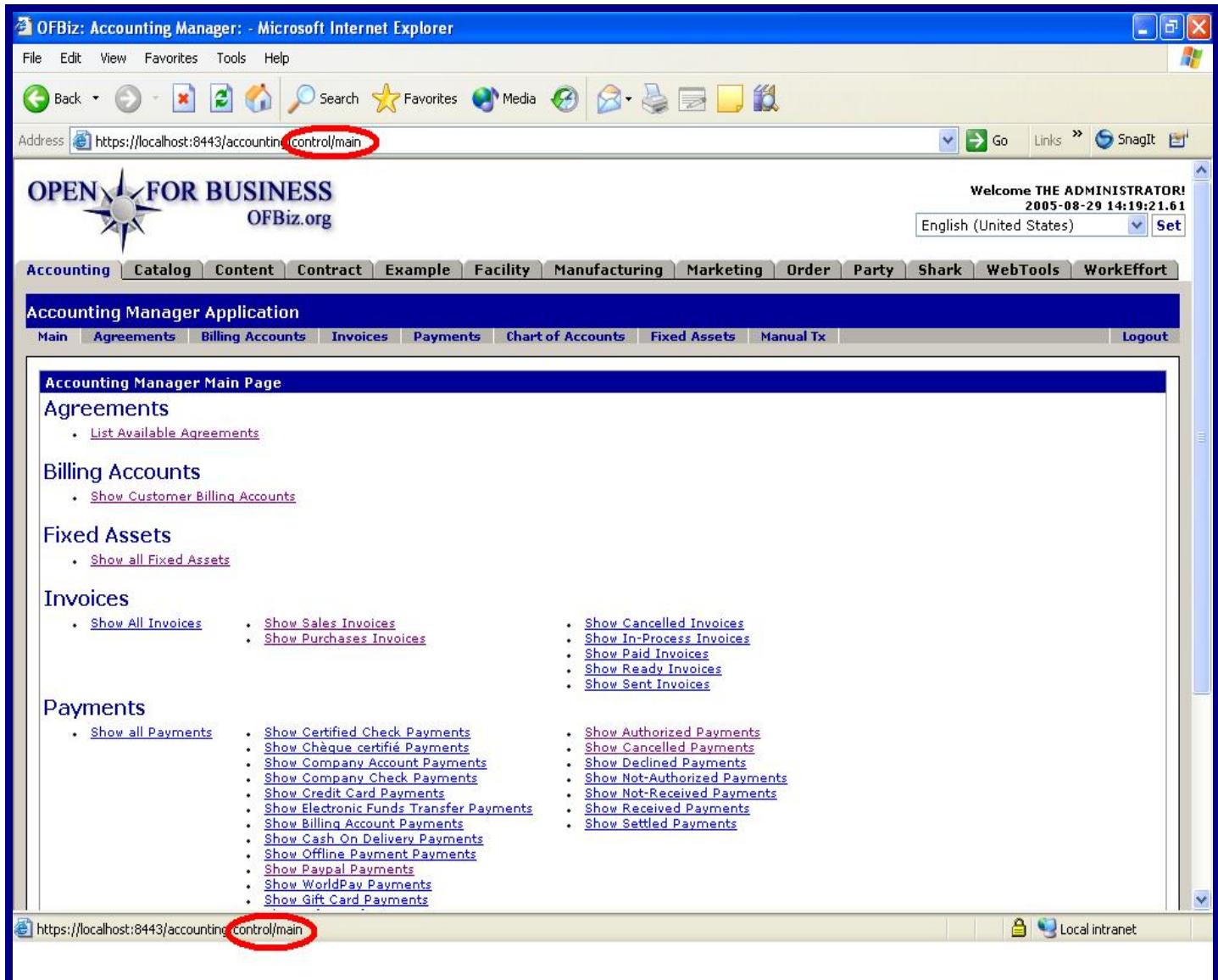
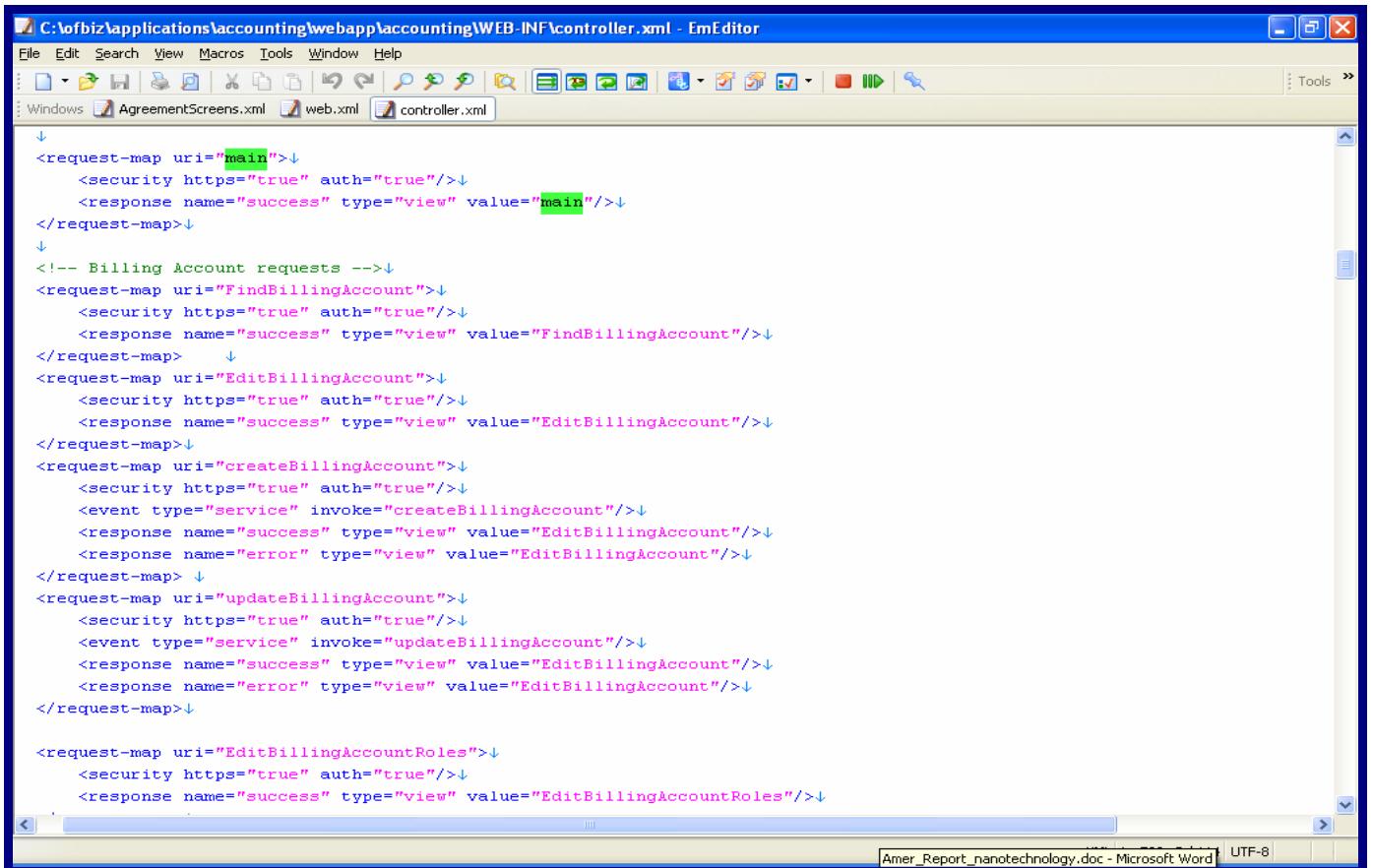


Figure 27

Now we will look for the “main” in the controller.xml file :



The screenshot shows the EmEditor application window with the file 'controller.xml' open. The code in the editor is as follows:

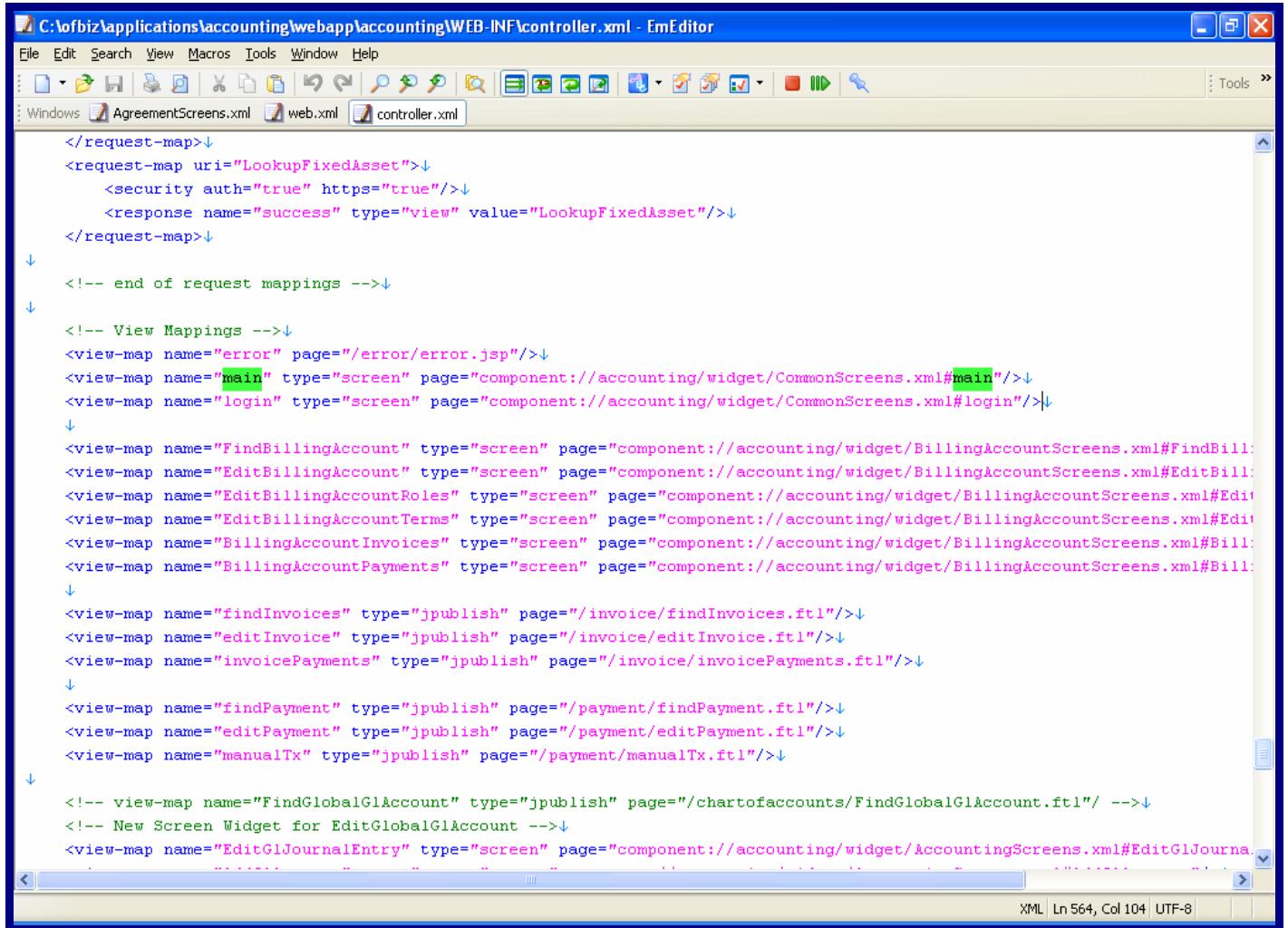
```
<request-map uri="main">↓
  <security https="true" auth="true"/>↓
  <response name="success" type="view" value="main"/>↓
</request-map>↓
↓
<!-- Billing Account requests -->↓
<request-map uri="FindBillingAccount">↓
  <security https="true" auth="true"/>↓
  <response name="success" type="view" value="FindBillingAccount"/>↓
</request-map> ↓
<request-map uri="EditBillingAccount">↓
  <security https="true" auth="true"/>↓
  <response name="success" type="view" value="EditBillingAccount"/>↓
</request-map>↓
<request-map uri="createBillingAccount">↓
  <security https="true" auth="true"/>↓
  <event type="service" invoke="createBillingAccount"/>↓
  <response name="success" type="view" value="EditBillingAccount"/>↓
  <response name="error" type="view" value="EditBillingAccount"/>↓
</request-map> ↓
<request-map uri="updateBillingAccount">↓
  <security https="true" auth="true"/>↓
  <event type="service" invoke="updateBillingAccount"/>↓
  <response name="success" type="view" value="EditBillingAccount"/>↓
  <response name="error" type="view" value="EditBillingAccount"/>↓
</request-map>↓
<request-map uri="EditBillingAccountRoles">↓
  <security https="true" auth="true"/>↓
  <response name="success" type="view" value="EditBillingAccountRoles"/>↓
```

Figure 28

the requested map is “main” and when success , this “main” is of type “view” which means it is a screen, not a service , and its value is “main” .

Then we will search for this view at the end of the controller.xml file, whose value is “main”

we will find :



The screenshot shows the EmEditor application window displaying the XML file C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml. The file contains various XML elements for request mappings and view mappings. A specific line of code is highlighted with a green background: <view-map name="main" type="screen" page="component://accounting/widget/CommonScreens.xml#main"/>. This line defines a view map named "main" of type "screen" pointing to the "main" screen in the CommonScreens.xml file located in the accounting/widget directory.

```
</request-map>↓
<request-map uri="LookupFixedAsset">↓
    <security auth="true" https="true"/>↓
    <response name="success" type="view" value="LookupFixedAsset"/>↓
</request-map>↓
↓
<!-- end of request mappings -->↓
↓
<!-- View Mappings -->↓
<view-map name="error" page="/error/error.jsp"/>↓
<view-map name="main" type="screen" page="component://accounting/widget/CommonScreens.xml#main"/>↓
<view-map name="login" type="screen" page="component://accounting/widget/CommonScreens.xml#login"/>↓
↓
<view-map name="FindBillingAccount" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#FindBilli:↓
<view-map name="EditBillingAccount" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#EditBilli:↓
<view-map name="EditBillingAccountRoles" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#EditBili:↓
<view-map name="EditBillingAccountTerms" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#EditBili:↓
<view-map name="BillingAccountInvoices" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#Billi:↓
<view-map name="BillingAccountPayments" type="screen" page="component://accounting/widget/BillingAccountScreens.xml#Billi:↓
↓
<view-map name="findInvoices" type="jpublish" page="/invoice/findInvoices.ftl"/>↓
<view-map name="editInvoice" type="jpublish" page="/invoice/editInvoice.ftl"/>↓
<view-map name="invoicePayments" type="jpublish" page="/invoice/invoicePayments.ftl"/>↓
↓
<view-map name="findPayment" type="jpublish" page="/payment/findPayment.ftl"/>↓
<view-map name="editPayment" type="jpublish" page="/payment/editPayment.ftl"/>↓
<view-map name="manualTx" type="jpublish" page="/payment/manualTx.ftl"/>↓
↓
<!-- view-map name="FindGlobalGlAccount" type="jpublish" page="/chartoffaccounts/FindGlobalGlAccount.ftl"/ -->↓
<!-- New Screen Widget for EditGlobalGlAccount -->↓
<view-map name="EditGlJournalEntry" type="screen" page="component://accounting/widget/AccountingScreens.xml#EditGlJourn:↓
.....
```

Figure 29

Here it gives the path for the screen. We should follow the path in the page=".....". There is a CommonScreen.xml file and inside this file there is a screen called “main”. “the name after the # symbol is the screen name”.

Now we will follow this screen, we will go to the widget directory inside the accounting application :

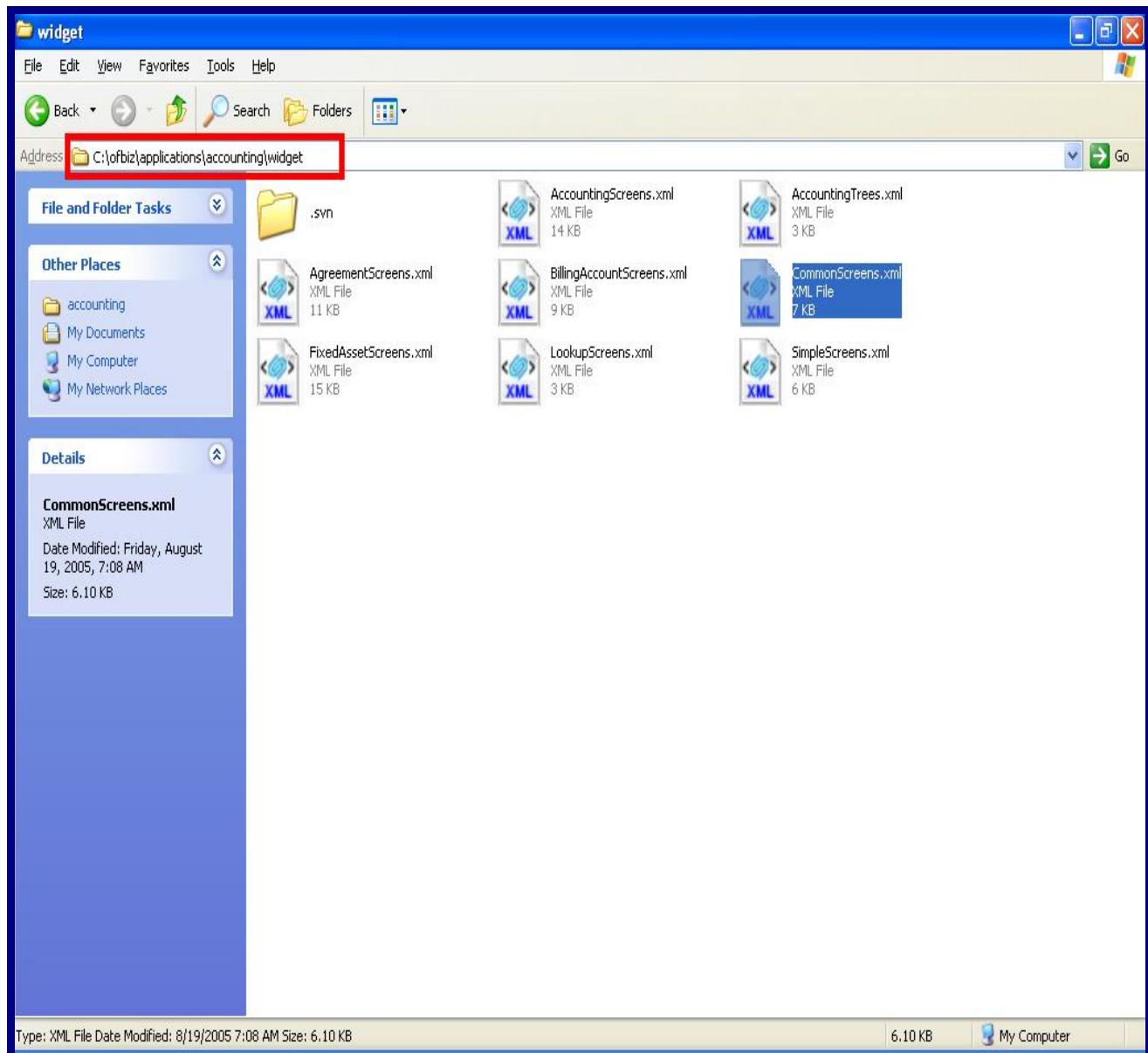


Figure 30

inside the CommonScreen.xml file, we will look for the “main” screen.

```
C:\ofbiz\applications\accounting\widget\CommonScreens.xml - EmEditor
File Edit Search View Macros Tools Window Help
Windows AgreementScreens.xml web.xml controller.xml CommonScreens.xml Tools >
</screen>↓
↓
<screen name="main">↓
<section>↓
<actions>↓
<set field="headerItem" value="main"/>↓
<entity-condition entity-name="PaymentMethodType" list-name="paymentMethodTypes">↓
<order-by field-name="paymentMethodTypeId"/>↓
</entity-condition>↓
<entity-condition entity-name="StatusItem" list-name="invoiceStatus">↓
<condition-expr field-name="statusTypeId" operator="equals" value="INVOICE_STATUS"/>↓
<order-by field-name="statusId"/>↓
</entity-condition>↓
<entity-condition entity-name="StatusItem" list-name="paymentStatus">↓
<condition-expr field-name="statusTypeId" operator="equals" value="PAYMENT_PREF_STATUS"/>↓
<order-by field-name="statusId"/>↓
</entity-condition>↓
</actions>↓
<widgets>↓
<decorator-screen name="main-decorator">↓
<decorator-section name="body">↓
<container style="screenlet">↓
<container style="screenlet-header">↓
<label style="boxhead" text="${uiLabelMap.AccountingMainPage}"/>↓
</container>↓
<platform-specific><html><html-template location="component://accounting/webapp/accounting/main:</html></platform-specific>↓
</container>↓
</decorator-section>↓
</decorator-screen>↓
</widgets>↓
</section>↓
</screen>↓
```

Figure 31

This is just a simple start, the structure if the screens will be easier to see in the coming examples.

*Back to the application :*

Now, we would visit the Agreements page. If you move the mouse above the “Agreements” tab you would notice that it makes a request to the “FindAgreement” map of the controller, as shown below.

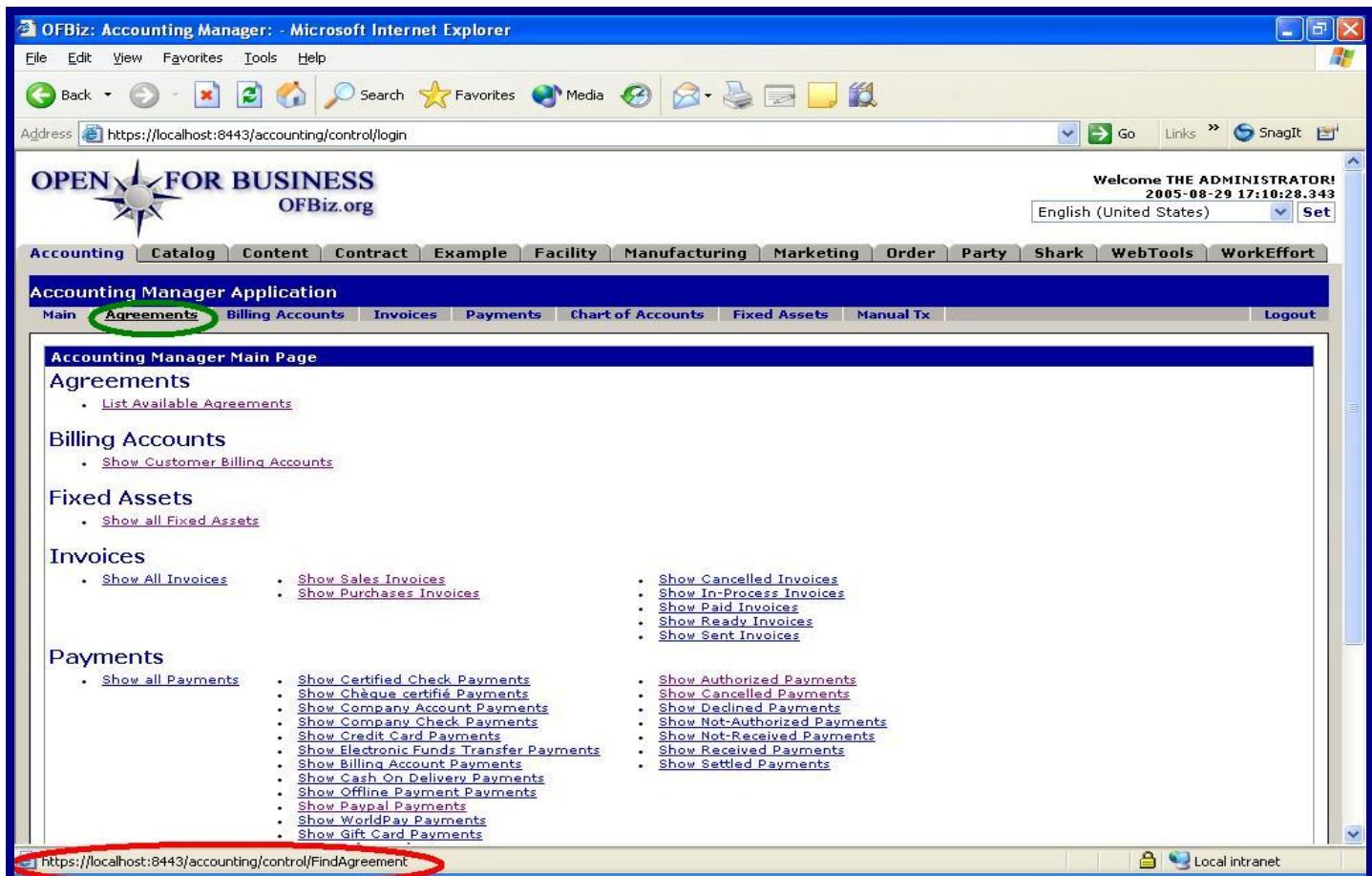


Figure 32

One click, and you will be forwarded to the FindAgreement page ...

The screenshot shows the 'Find Agreements' page. At the top, there's a search form with fields for Agreement Id, Product Id, Party Id From, Party Id To, Agreement Type Id, and From Date. Below the form is a table listing agreements. The table has columns for Edit, Product Id, Party Id From, Party Id To, Role Type Id To, Agreement Type Id, From Date, Thru Date, and Description. Each row in the table includes a '[Cancel]' link. The table data is as follows:

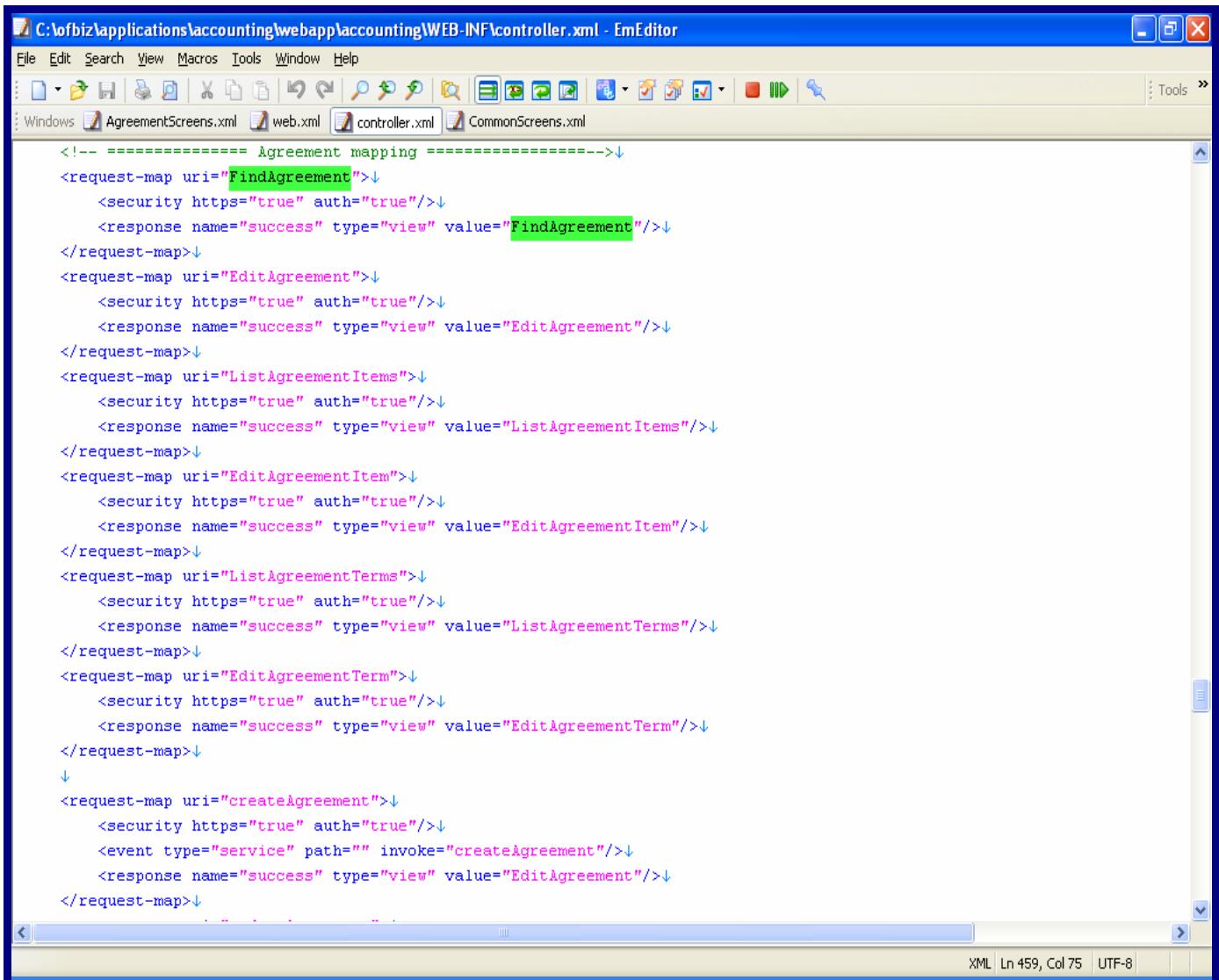
Edit	Product Id	Party Id From	Party Id To	Role Type Id To	Agreement Type Id	From Date	Thru Date	Description	
10000	GZ-1000			Person		2005-08-14 10:56:08.554			[Cancel]
10043						2005-08-23 11:27:19.265			[Cancel]
AGR_TEST	Company	DemoSupplier	Supplier	Purchase				Agreement for DemoSupplier	[Cancel]
1000	Company	BigSupplier	Supplier	Purchase				Purchasing Agreement with BigSupplier	[Cancel]
1001	Company	EuroSupplier	Supplier	Purchase				Purchasing Agreement with EuroSupplier-Milan	[Cancel]
1002	Company	EuroSupplier	Supplier	Purchase				Purchasing Agreement with EuroSupplier-New York	[Cancel]
10030						2005-08-17 19:04:34.875			[Cancel]

Figure 33

How that forwarding happened? We would follow it step by step :

1)Search the controller for the FindAgreement , and again the controller.xml is here :

C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF or you can see it in the path in the Figure.



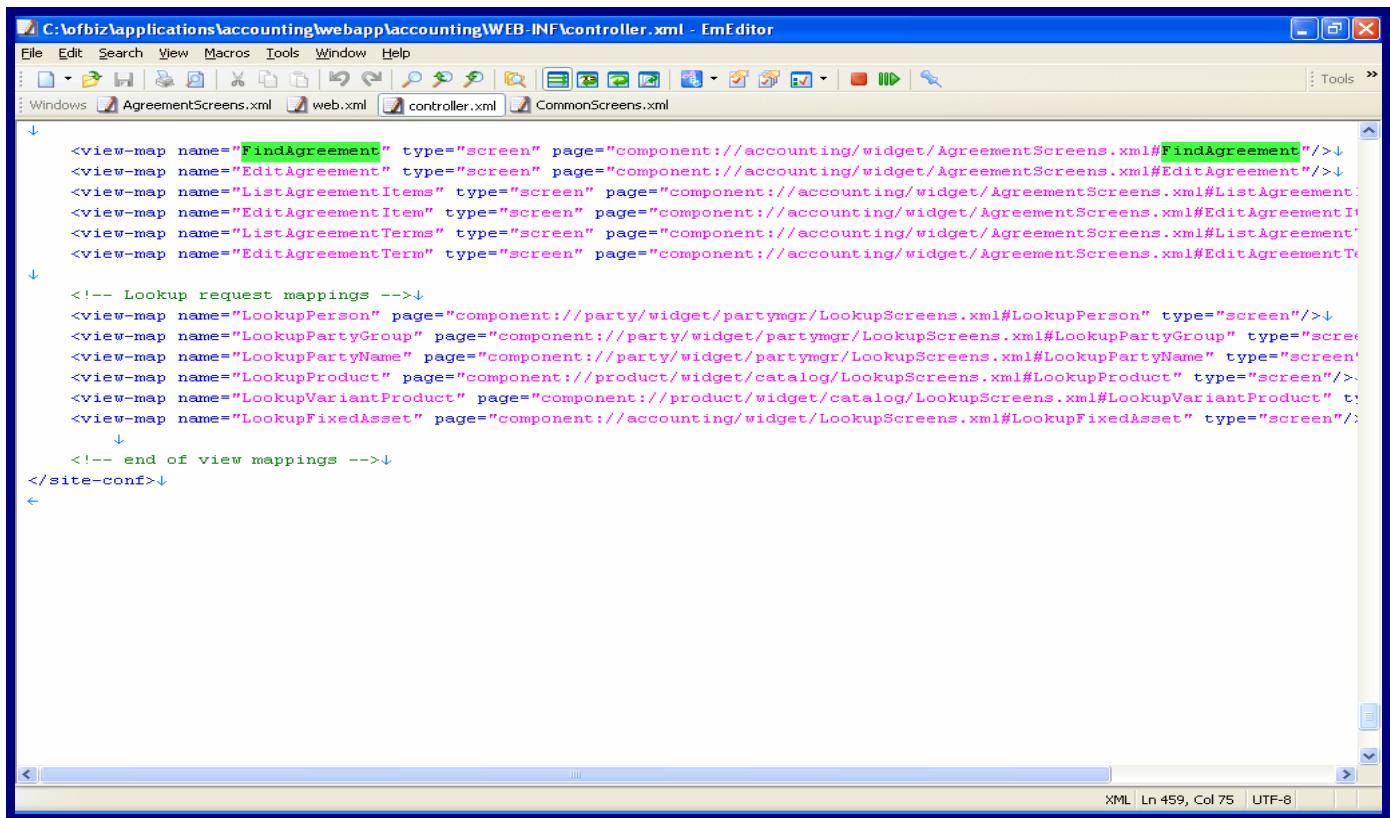
The screenshot shows the EmEditor application window displaying the XML file C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml. The file contains several request-map entries. One entry for 'FindAgreement' is highlighted with a green background. The XML code is as follows:

```
<!-- ===== Agreement mapping =====-->
<request-map uri="FindAgreement">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="FindAgreement"/>↓
</request-map>↓
<request-map uri="EditAgreement">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
<request-map uri="ListAgreementItems">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementItems"/>↓
</request-map>↓
<request-map uri="EditAgreementItem">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementItem"/>↓
</request-map>↓
<request-map uri="ListAgreementTerms">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementTerms"/>↓
</request-map>↓
<request-map uri="EditAgreementTerm">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementTerm"/>↓
</request-map>↓
↓
<request-map uri="createAgreement">↓
    <security https="true" auth="true"/>↓
    <event type="service" path="" invoke="createAgreement"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
```

Figure 34

you will find that FindAgreement is of type view, and when the request succeeded , you will be forwarded to the view whose value is : FindAgreement.

2) Still inside the controller, go down at the end of the file, and look for the view-map whose name is “FindAgreement” .



```
C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml - EmEditor
File Edit Search View Macros Tools Window Help
Windows AgreementScreens.xml web.xml controller.xml CommonScreens.xml
Tools >
↓
<view-map name="FindAgreement" type="screen" page="component://accounting/widget/AgreementScreens.xml#FindAgreement"/>↓
<view-map name="EditAgreement" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreement"/>↓
<view-map name="ListAgreementItems" type="screen" page="component://accounting/widget/AgreementScreens.xml#ListAgreementItems"/>↓
<view-map name="EditAgreementItem" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreementItem"/>↓
<view-map name="ListAgreementTerms" type="screen" page="component://accounting/widget/AgreementScreens.xml#ListAgreementTerms"/>↓
<view-map name="EditAgreementTerm" type="screen" page="component://accounting/widget/AgreementScreens.xml#EditAgreementTerm"/>↓
<!-- Lookup request mappings --&gt;↓
&lt;view-map name="LookupPerson" page="component://party/widget/partymgr/LookupScreens.xml#LookupPerson" type="screen"/&gt;↓
&lt;view-map name="LookupPartyGroup" page="component://party/widget/partymgr/LookupScreens.xml#LookupPartyGroup" type="screen"/&gt;↓
&lt;view-map name="LookupPartyName" page="component://party/widget/partymgr/LookupScreens.xml#LookupPartyName" type="screen"/&gt;↓
&lt;view-map name="LookupProduct" page="component://product/widget/catalog/LookupScreens.xml#LookupProduct" type="screen"/&gt;↓
&lt;view-map name="LookupVariantProduct" page="component://product/widget/catalog/LookupScreens.xml#LookupVariantProduct" type="screen"/&gt;↓
&lt;view-map name="LookupFixedAsset" page="component://accounting/widget/LookupScreens.xml#LookupFixedAsset" type="screen"/&gt;↓
<!-- end of view mappings --&gt;↓
&lt;/site-conf&gt;↓
←</pre>
```

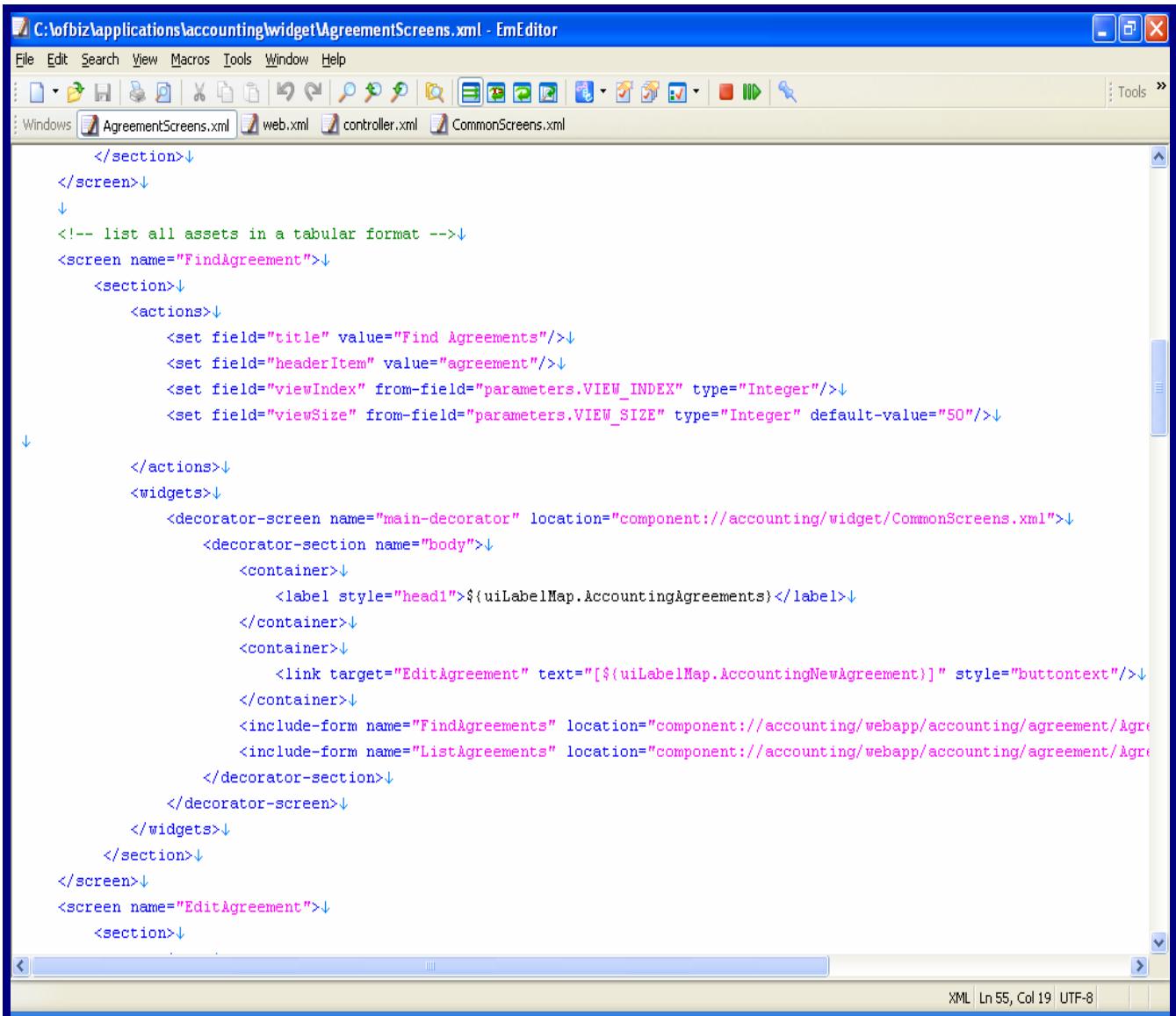
Figure 35

Now we would see that the FindAgreement view and is located in the AgreementScreens.xml file, in the #FindAgreement screen.

Note :

The screen files, for example AgreementScreens.xml file, contain many screens inside it. So to determine which screen is the one to be rendered , it is stated in the controller after the symbol ‘#’ . Thus, in our case we know it is the screen FindAgreement inside the AgreementScreens.xml

3) We would follow the path of the screen , provided in the controller, again the path appears in the header of each Figure.



The screenshot shows the EmEditor application window displaying XML code for a screen named "FindAgreement". The code includes sections for actions, widgets, and include-forms. It also references a "main-decorator" and other screens like "EditAgreement" and "ListAgreements". The XML editor interface is visible with toolbars and status bars at the top and bottom respectively.

```
</section>↓
</screen>↓
↓
<!-- list all assets in a tabular format -->↓
<screen name="FindAgreement">↓
    <section>↓
        <actions>↓
            <set field="title" value="Find Agreements"/>↓
            <set field="headerItem" value="agreement"/>↓
            <set field="viewIndex" from-field="parameters.VIEW_INDEX" type="Integer"/>↓
            <set field="viewSize" from-field="parameters.VIEW_SIZE" type="Integer" default-value="50"/>↓
        ↓
        </actions>↓
        <widgets>↓
            <decorator-screen name="main-decorator" location="component://accounting/widget/CommonScreens.xml">↓
                <decorator-section name="body">↓
                    <container>↓
                        <label style="head1">${uiLabelMap.AccountingAgreements}</label>↓
                    </container>↓
                    <container>↓
                        <link target="EditAgreement" text="[$(uiLabelMap.AccountingNewAgreement)]" style="buttonText"/>↓
                    </container>↓
                    <include-form name="FindAgreements" location="component://accounting/webapp/accounting/agreement/AgreementListForm" />↓
                    <include-form name="ListAgreements" location="component://accounting/webapp/accounting/agreement/AgreementListForm" />↓
                </decorator-section>↓
            </decorator-screen>↓
        </widgets>↓
    </section>↓
</screen>↓
<screen name="EditAgreement">↓
    <section>↓
```

Figure 36

Notice that the FindAgreement screen uses the “main-decorator”, so all together form the page. Thus we will take each part in the page. To know more about the main-decorator, you could refer to the HelloWorld tutorials in [here](#) .

Now here is the FindAgreement page :

The screenshot shows the 'OFBiz: Accounting Manager: Find Agreements' page in Microsoft Internet Explorer. The title bar reads 'OFBiz: Accounting Manager: Find Agreements - Microsoft Internet Explorer'. The menu bar includes File, Edit, View, Favorites, Tools, and Help. The toolbar contains standard icons for Back, Forward, Stop, Refresh, Home, Search, Favorites, Media, and Print. The address bar shows the URL 'https://localhost:8443/accounting/control/FindAgreement'. The top right corner displays a welcome message for 'THE ADMINISTRATOR!' dated '2005-08-29 17:53:36.015' and language settings for 'English (United States)'. The main content area has a header 'Accounting Manager Application' with tabs for Main, Agreements, Billing Accounts, Invoices, Payments, Chart of Accounts, Fixed Assets, Manual Tx, Logout, and a 'Logout' link. Below this is a section titled 'Agreements' with a sub-section '[Create Agreement]'. It features several search fields: 'Agreement Id' (with operators Equals, Begins With, Contains, Is Empty, Ignore Case), 'Product Id', 'Party Id From', 'Party Id To', 'Agreement Type Id' (dropdown), 'From Date' (with operators Equals, Same Day, Greater Than From Day Start, Greater Than, Less Than, Up To Day, Up Thru Day, Is Empty), and a 'Find' button. A table lists agreements with columns: Edit, Product Id, Party Id From, Party Id To, Role Type Id To, Agreement Type Id, From Date, Thru Date, and Description. The table contains the following data:

Edit	Product Id	Party Id From	Party Id To	Role Type Id To	Agreement Type Id	From Date	Thru Date	Description
10000	GZ-1000			Person		2005-08-14 10:56:08.554		[Cancel]
10043						2005-08-23 11:27:19.265		[Cancel]
AGR_TEST		Company	DemoSupplier	Supplier	Purchase			Agreement for DemoSupplier [Cancel]
1000		Company	BigSupplier	Supplier	Purchase			Purchasing Agreement with BigSupplier [Cancel]
1001		Company	EuroSupplier	Supplier	Purchase			Purchasing Agreement with EuroSupplier-Milan [Cancel]
1002		Company	EuroSupplier	Supplier	Purchase			Purchasing Agreement with [Cancel]

Figure 37

The page is divided into many parts :

This is the application bar, and it contains all the OFBiz applications, the first one is the accounting application.



Figure 38

**This is the name of the Manager Application. This comes from the appheader.ftl file**



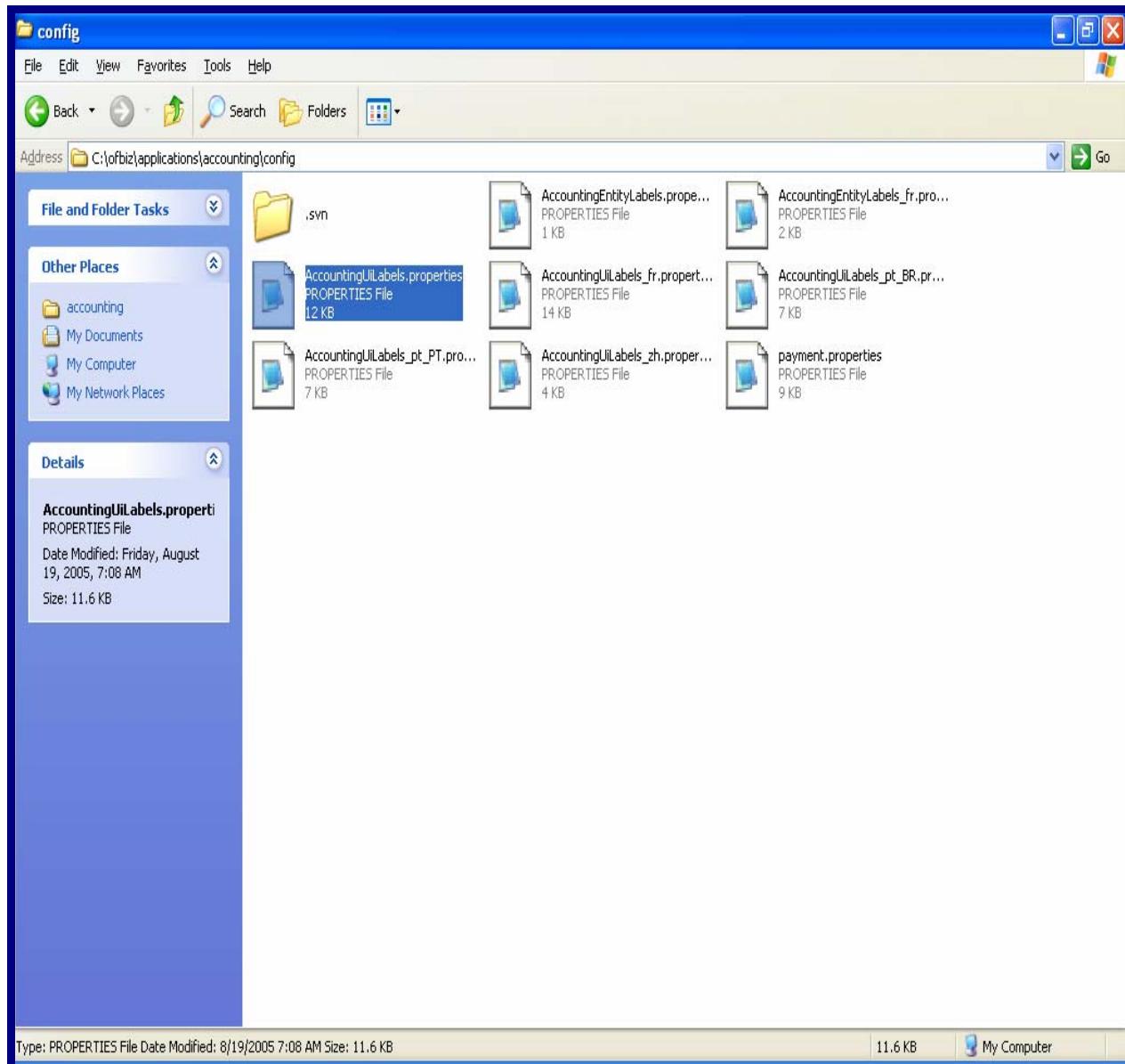
**Figure 39**

The screenshot shows the EmEditor application interface with the file "appheader.ftl" open. The code in the editor is as follows:

```
<%#if (requestAttributes.security)?exists><#assign security = requestAttributes.security></#if>↓
<%#if (requestAttributes.userLogin)?exists><#assign userLogin = requestAttributes.userLogin></#if>↓
<%#if (requestAttributes.checkLoginUrl)?exists><#assign checkLoginUrl = requestAttributes.checkLoginUrl></#if>↓
↓
<%#assign unselectedLeftClassName = "headerButtonLeft">↓
<%#assign unselectedRightClassName = "headerButtonRight">↓
<%#assign selectedLeftClassMap = {page.headerItem?default("void") : "headerButtonLeftSelected"}>↓
<%#assign selectedRightClassMap = {page.headerItem?default("void") : "headerButtonRightSelected"}>↓
↓
<div class="apptitle">${uiLabelMap.AccountingManagerApplication}</div>↓
<div class="row">↓
  <div class="col"><a href=<%@ofbizUrl>/main<@ofbizUrl>" class="${selectedLeftClassMap.main?default(unselectedLeftClassName)}>
    <div class="col"><a href=<%@ofbizUrl>/FindAgreement<@ofbizUrl>" class="${selectedLeftClassMap.agreement?default(unselectedLeftClassName)}>
      <div class="col"><a href=<%@ofbizUrl>/FindBillingAccount<@ofbizUrl>" class="${selectedLeftClassMap.billingaccount?default(unselectedLeftClassName)}>
        <div class="col"><a href=<%@ofbizUrl>/findInvoices<@ofbizUrl>" class="${selectedLeftClassMap.invoices?default(unselectedLeftClassName)}>
          <div class="col"><a href=<%@ofbizUrl>/findPayment<@ofbizUrl>" class="${selectedLeftClassMap.payments?default(unselectedLeftClassName)}>
            <div class="col"><a href=<%@ofbizUrl>/FindGlobalGIAccount<@ofbizUrl>" class="${selectedLeftClassMap.chartoffaccounts?default(unselectedLeftClassName)}>
              <div class="col"><a href=<%@ofbizUrl>/ListFixedAssets<@ofbizUrl>" class="${selectedLeftClassMap.ListFixedAssets?default(unselectedLeftClassName)}>
                ↓
                <%#if security.hasEntityPermission("MANUAL", "_PAYMENT", session)>↓
                  <div class="col"><a href=<%@ofbizUrl>/manualETx<@ofbizUrl>" class="${selectedLeftClassMap.manualTx?default(unselectedLeftClassName)}>
                </#if>↓
                <%#if userLogin?has_content>↓
                  <div class="col-right"><a href=<%@ofbizUrl>/logout<@ofbizUrl>" class="${selectedRightClassMap.logout?default(unselectedRightClassName)}>
                <%#else>↓
                  <div class="col-right"><a href='<%@ofbizUrl>${checkLoginUrl?if_exists}<@ofbizUrl>' class='${selectedRightClassMap.login?default(unselectedRightClassName)}>
                </#if>↓
                <div class="col-fill">&ampnbsp</div>↓
  </div>↓
  ↓
```

**Figure 40**

Note that it uses the uiLabelMap.AccountingManagerApplication, i.e., it is reading the name “Accounting Manager Application” from the uiLabelMap. This is located in the config directory : C:\ofbiz\applications\accounting\config



**Figure 41**

The screenshot shows the EmEditor application window with the file 'AccountingUILabels.properties' open. The window title is 'C:\ofbiz\applications\accounting\config\AccountingUILabels.properties - EmEditor'. The menu bar includes File, Edit, Search, View, Macros, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, Find, and Copy. Below the toolbar is a tab bar with several files: Windows, AgreementScreens.xml, web.xml, controller.xml, CommonScreens.xml, AccountingUILabels.properties (which is the active tab), appheader.ftl, and appheader.ftl. The main text area displays a series of key-value pairs representing UI labels:

```
AccountingInvoiceRoles=Roles↓
AccountingInvoiceStatus=Status↓
AccountingInvoiceTotal=Invoice Total↓
AccountingInvoices=Invoices↓
AccountingInvoicesFound=Invoices Found↓
AccountingInvoicesMenu=Invoices↓
AccountingLastNameCard=Last Name on Card↓
AccountingLiabilities=Liabilities↓
AccountingLineTotal=Line Total↓
AccountingLookupInvoices=Lookup Invoices↓
AccountingLookupPayment=Lookup Payment↓
AccountingMainMenu=Main↓
Accounting MainPage=Accounting Manager Main Page↓
Accounting MainPage Note=Welcome to the Accounting Manager!↓
AccountingManagerApplication=Accounting Manager Application↓
AccountingManualJournalEntry=Manual Journal Entry↓
AccountingManualPostTrans=Manual Transactions Posting↓
AccountingManualTxMenu=Manual Tx↓
AccountingMethod=Method↓
AccountingMiddleNameCard=Middle Name on Card↓
AccountingName=Name↓
AccountingNameAccount=Name on Account↓
AccountingNetIncome=Net Income↓
AccountingNewAccount>New Account↓
AccountingNewAgreement>Create Agreement↓
AccountingNewAgreementItem>Create Agreement Item↓
AccountingNewAgreementTerm>Create Agreement Term↓
AccountingNewFixedAsset>New Fixed Asset↓
AccountingNewFixedAssetStdCost>New Standard Cost↓
AccountingNewPayment>New Payment↓
AccountingNoAccount>No Account↓
```

At the bottom right of the editor window, it says 'Text | Ln 133, Col 32 | Arabic (Windows)'. There is also a scroll bar on the right side of the editor window.

Figure 42

Note that this is needed because different languages would use different user interface, so this would also be found in French, German, ..etc .

---

This is also from the appheader.ftl file.



```

<div class="row">↓
<div class="col"><a href=<@ofbizUrl>/main<@ofbizUrl>" class="${selectedLeftClassMap.main?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/FindAgreement<@ofbizUrl>" class="${selectedLeftClassMap.agreement?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/FindBillingAccount<@ofbizUrl>" class="${selectedLeftClassMap.billingaccount?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/findInvoices<@ofbizUrl>" class="${selectedLeftClassMap.invoices?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/findPayment<@ofbizUrl>" class="${selectedLeftClassMap.payments?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/FindGlobalAccount<@ofbizUrl>" class="${selectedLeftClassMap.chartoffaccounts?default(unselectedClassName)}>
<div class="col"><a href=<@ofbizUrl>/ListFixedAssets<@ofbizUrl>" class="${selectedLeftClassMap.ListFixedAssets?default(unselectedClassName)}>
</div>↓
<#if security.hasEntityPermission("MANUAL", "_PAYMENT", session)>↓
<div class="col"><a href=<@ofbizUrl>/manualETX<@ofbizUrl>" class="${selectedLeftClassMap.manualTx?default(unselectedClassName)}>
</#if>↓
<#if userLogin?has_content>↓
<div class="col-right"><a href=<@ofbizUrl>/logout<@ofbizUrl>" class="${selectedRightClassMap.logout?default(unselectedClassName)}>
</div>↓
<#else>↓
<div class="col-right"><a href='<@ofbizUrl>${checkLoginUrl?if_exists}<@ofbizUrl>' class='${selectedRightClassMap.login?default(unselectedClassName)}>
</#if>↓
<div class="col-fill">&ampnbsp</div>↓
</div>↓

```

**Figure 43**

---

After that , the rest of the page is coming from the <widget> section of the FindAgreement screen.

```

<widgets>↓
<decorator-screen name="main-decorator" location="component://accounting/widget/CommonScreens.xml">↓
<decorator-section name="body">↓
<container>↓
<label style="head1">${uiLabelMap.AccountingAgreements}</label>↓
</container>↓
<container>↓
<link target="EditAgreement" text="[$(uiLabelMap.AccountingNewAgreement)]" style="buttoncontext"/>↓
</container>↓
<include-form name="FindAgreements" location="component://accounting/webapp/accounting/agreement/AgreementForm">↓
<include-form name="ListAgreements" location="component://accounting/webapp/accounting/agreement/AgreementListForm">↓
</decorator-section>↓
</decorator-screen>↓
</widgets>↓

```

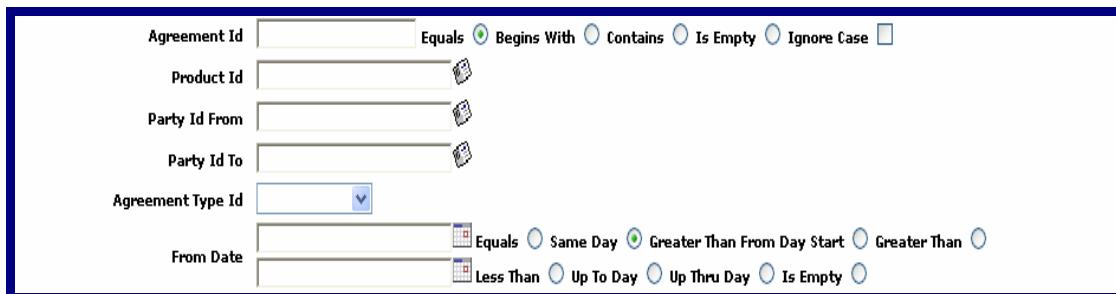
**Figure 44**

Agreements  
[Create Agreement]

This text “Agreements” is coming from the <container> tag, the uiLabelMap.AccountignAgreements provide us with the word “Agreement” with different languages.

The [Create Agreement] comes from the container that has a link to the EditAgreement page.

---



The screenshot shows a search interface for agreements. It includes fields for 'Agreement Id' (with operators: Equals, Begins With, Contains, Is Empty, Ignore Case), 'Product Id' (with a magnifying glass icon), 'Party Id From' (with a magnifying glass icon), 'Party Id To' (with a magnifying glass icon), 'Agreement Type Id' (with a dropdown menu), and 'From Date' (with operators: Equals, Same Day, Greater Than From Day Start, Greater Than, Less Than, Up To Day, Up Thru Day, Is Empty). The entire interface is enclosed in a blue border.

Figure 45

This part comes from the included form, FindAgreements :

```
<include-form name="FindAgreements" location="component://accounting/webapp/accounting/agreement/AgreementForms
```

Figure 46

---

while :

Edit	Product Id	Party Id From	Party Id To	Role Type Id To	Agreement Type Id	From Date	Thru Date	Description	
10000	GZ-1000			Person		2005-08-14 10:56:08.554			[Cancel]
10043						2005-08-23 11:27:19.265			[Cancel]
AGR_TEST	Company	DemoSupplier	Supplier	Purchase				Agreement for DemoSupplier	[Cancel]
1000	Company	BigSupplier	Supplier	Purchase				Purchasing Agreement with BigSupplier	[Cancel]
1001	Company	EuroSupplier	Supplier	Purchase				Purchasing Agreement with EuroSupplier-Milan	[Cancel]
1002	Company	EuroSupplier	Supplier	Purchase				Purchasing Agreement with EuroSupplier-New York	[Cancel]
10030						2005-08-17 19:04:34.875			[Cancel]

This part comes from the included form : ListAgreements:

```
<include-form name="ListAgreements" location="component://accounting/webapp/accounting/agreement/AgreementForms.x
```

Figure 47

---

Now we would have a look at the Forms , that are located in the directory :

C:\ofbiz\applications\accounting\webapp\accounting\agreement  
\AgreementForms.xml

Here is the FindAgreements form :

```
<forms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↓
    xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/widget-form.xsd">↓
↓
<form name="FindAgreements" default-title-style="tableheadtext"↓
    default-tooltip-style="tabletext" default-widget-style="inputBox"↓
    target="FindAgreement" title="" type="single">↓
<auto-fields-entity entity-name="Agreement" default-field-type="find"/>↓
<field name="productId" title="${uiLabelMap.AccountingProductId}">↓
    <lookup target-form-name="LookupProduct"/>↓
</field>↓
<field name="agreementTypeId" title="${uiLabelMap.AccountingAgreementTypeId}">↓
    <drop-down allow-empty="true">↓
        <entity-options description="${description}" entity-name="AgreementType" key-field-name="agreementTypeId"/>↓
    </drop-down>↓
</field>↓
<field name="partyIdFrom" title="${uiLabelMap.AccountingPartyIdFrom}">↓
    <lookup target-form-name="LookupPartyName"/>↓
</field>↓
<field name="partyIdTo" title="${uiLabelMap.AccountingPartyIdTo}">↓
    <lookup target-form-name="LookupPartyName"/>↓
</field>↓
<field name="roleIdFrom" title="${uiLabelMap.AccountingRoleIdFrom}"><hidden/></field>↓
<field name="roleIdTo" title="${uiLabelMap.AccountingRoleIdTo}"><hidden/></field>↓
<field name="agreementDate" title="${uiLabelMap.AccountingAgreementDate}"><hidden/></field>↓
<field name="description" title="${uiLabelMap.AccountingDescription}"><hidden/></field>↓
<field name="textData" title="${uiLabelMap.AccountingTextData}"><hidden/></field>↓
<field name="thruDate" title="${uiLabelMap.AccountingThruDate}"><hidden/></field>↓
<field name="submitButton" title="${uiLabelMap.CommonFind}" widget-style="standardSubmit">↓
    <submit button-type="button"/>↓
</field>↓
</form>↓
```

Figure 48

Notice that a lot of tags are used, for the time being, you need just to know the most important ones that serve your application.

target="FindAgreement" title="" type="single">↓

Figure 49

Shows to which page is the next target after serving the form.

```
<auto-fields-entity entity-name="Agreement" default-field-type="find"/>↓
```

Figure 50

This tag will by default, read all the fields of the Agreement entity, and display them in the same format based on the default-field-type , unless the field is explicitly specified to have a different feature as we would see.

Note that the “find” field type is the default, and it is among four different field types , find , edit, display and hidden, try changing the “find” to any of those and see what changes would occur, don’t worry it is an open source project!

Here is the Agreement Entity and its fields :

```
<entity entity-name="Agreement"↓
  package-name="org.ofbiz.party.agreement"↓
  title="Agreement Entity">↓
<field name="agreementId" type="id-ne"></field>↓
<field name="productId" type="id"></field>↓
<field name="partyIdFrom" type="id"></field>↓
<field name="partyIdTo" type="id"></field>↓
<field name="roleTypeIdFrom" type="id"></field>↓
<field name="roleTypeIdTo" type="id"></field>↓
<field name="agreementTypeId" type="id"></field>↓
<field name="agreementDate" type="date-time"></field>↓
<field name="fromDate" type="date-time"></field>↓
<field name="thruDate" type="date-time"></field>↓
<field name="description" type="description"></field>↓
<field name="textData" type="long-varchar"></field>↓
```

Figure 51

Unless a field is explicitly specified, all the field will appear with a “find” filed type.

---

```
<field name="productId" title="${uiLabelMap.AccountingProductId}">↓
  <lookup target-form-name="LookupProduct"/>↓
</field>↓
```

Figure 52

Here is the productId field , it is explicitly specified to be labeled with the label AccountingProductId . You could also just write the name like title=”Product Id” . However they would use to read from the config files, to support different language as mentioned earlier.

It also uses another defined tag called lookup , to allows to see the product information to choose among them.

```

<field name="agreementTypeId" title="${uiLabelMap.AccountingAgreementTypeId}">↓
    <drop-down allow-empty="true">↓
        <entity-options description="${description}" entity-name="AgreementType" key-field-name="agreementTypeId"/>↓
    </drop-down>↓
</field>↓

```

**Figure 53**

The agreementTypeId field of the agreement entity is explicitly defined to have a label , using the title=”...”

Also, it is to be shows as a drop-down menu .

The <drop-down> has an attribute called allow-empty = “true” which means it can be empty, else if “false” then it has to have one of the types.

<entity-options> tells that the filed “agreementTypeId” will have the values from the AgreementType entity , and it will match it with the field “agreementTypeId of the AgreementType entity.

What will be shown to the users is the “description” field of the AgreementType entity that describes a particular agreementTypeId.

---

```

<field name="roleTypeIdFrom" title="${uiLabelMap.AccountingRoleIdFrom}"><hidden/></field>↓
<field name="roleTypeIdTo" title="${uiLabelMap.AccountingRoleIdTo}"><hidden/></field>↓
<field name="agreementDate" title="${uiLabelMap.AccountingAgreementDate}"><hidden/></field>↓
<field name="description" title="${uiLabelMap.AccountingDescription}"><hidden/></field>↓
<field name="textData" title="${uiLabelMap.AccountingTextData}"><hidden/></field>↓
<field name="thruDate" title="${uiLabelMap.AccountingThruDate}"><hidden/></field>↓

```

**Figure 54**

This list of fields are specified to be hidden, so clearly they will not appear in the page with other filesds.

---

Finally,

```

<field name="submitButton" title="${uiLabelMap.CommonFind}" widget-style="standardSubmit">↓
    <submit button-type="button"/>↓
</field>↓

```

**Figure 55**

The submit button to submit the form .

If you followed the ListAgreement form, you would notice it is similar to the findAgreement , so I won’t discuss it here .I would discuss now the EditAgreement page/form.

---

## Edit Agreement :

The screenshot shows the 'Find Agreements' page of the OFBiz Accounting Manager. At the top, there's a navigation bar with links like File, Edit, View, Favorites, Tools, Help, Back, Forward, Stop, Home, Search, Favorites, Media, Print, Copy, Paste, and SnagIt. Below that is a toolbar with icons for Back, Forward, Stop, Home, Search, Favorites, Media, Print, Copy, Paste, and SnagIt. The main menu has tabs for Accounting, Catalog, Content, Contract, Example, Facility, Manufacturing, Marketing, Order, Party, Shark, WebTools, and WorkEffort. The sub-menu for Accounting includes Main, Agreements, Billing Accounts, Invoices, Payments, Chart of Accounts, Fixed Assets, and Manual Tx. On the right, there's a Logout link. The main content area is titled 'Agreements' and features a 'Create Agreement' link which is circled in green. Below it are search fields for Agreement Id, Product Id, Party Id From, Party Id To, and Agreement Type Id. There are also date search fields for From Date and Thru Date, each with a dropdown menu showing comparison operators like Equals, Begins With, Contains, Is Empty, Ignore Case, Same Day, Greater Than From Day Start, Greater Than, Less Than, Up To Day, Up Thru Day, and Is Empty. A 'Find' button is located below these search fields. A table lists several agreements with columns for Edit, Product Id, Party Id From, Party Id To, Role Type Id To, Agreement Type Id, From Date, Thru Date, and Description. Each row has a '[cancel]' link in the last column. The table rows are: 10000 (GZ-1000, Person, 2005-08-14 10:56:08.554), 10043 (2005-08-23 11:27:19.265), AGR\_TEST (Company, DemoSupplier, Supplier, Purchase, Agreement for DemoSupplier), 1000 (Company, BigSupplier, Supplier, Purchase, Purchasing Agreement with BigSupplier), 1001 (Company, EuroSupplier, Supplier, Purchase, Purchasing Agreement with EuroSupplier-Milan), 1002 (Company, EuroSupplier, Supplier, Purchase, Purchasing Agreement with EuroSupplier-New York), and 10030 (2005-08-17 19:04:34.875). The bottom status bar shows the URL https://localhost:8443/accounting/control/EditAgreement and a Local intranet icon.

Figure 56

Again, and exactly as we did with the FindAgreement ,if we want to press on the [create Agreement] , we would notice a request to control/EditAgreement, and again, and if we followed the controller it will guide us to the EditAgreement screen in the AgreementScreens.xml file.

```
</widgets>↓
</section>↓
</screen>↓
<screen name="EditAgreement">↓
  <section>↓
    <actions>↓
      <set field="title" value="Edit Agreement"/>↓
      <set field="titleProperty" value="PageTitleEditAgreement"/>↓
      <set field="headerItem" value="agreement"/>↓
      <set field="tabButtonItem" value="EditAgreement"/>↓
      ↓
      <set field="agreementId" from-field="parameters.agreementId"/>↓
      <entity-one entity-name="Agreement" value-name="agreement"/>↓
    </actions>↓
    <widgets>↓
      <decorator-screen name="CommonAgreementDecorator">↓
        <decorator-section name="body">↓
          <container>↓
            <link target="EditAgreement" text="${uiLabelMap.AccountingNewAgreement}" style="buttoncontext"/>↓
          </container>↓
          <include-form name="EditAgreement" location="component://accounting/webapp/accounting/agreement/EditAgreementForm" />↓
        </decorator-section>↓
      </decorator-screen>↓
    </widgets>↓
  </section>↓
</screen>↓
<screen name="ListAgreementItems">↓
  <section>↓
    <actions>↓
      <set field="title" value="List Agreement Items"/>↓
      <set field="titleProperty" value="PageTitleListAgreementItems"/>↓

```

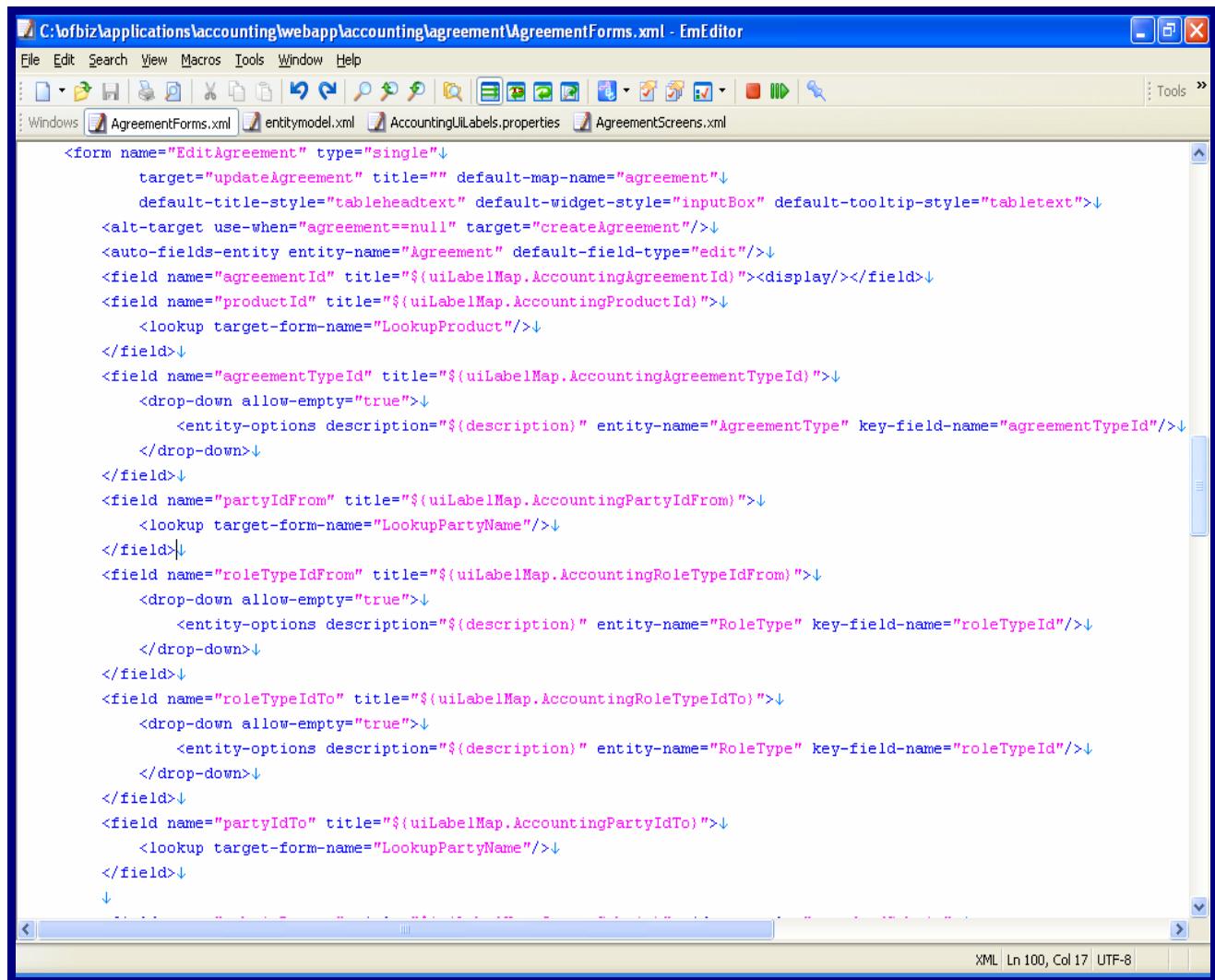
11,061 bytes, 212 lines. XML | Ln 1, Col 1 | UTF-8

Figure 57

Figure 57 shows the Edit Agreement screen , again similar to the FindAgreement screen.

It is also including the EditAgreement form, so whenever a request to this page is rendered, the form will be included in the page.

Here is the EditAgreement form :



The screenshot shows the EmEditor application window with the file 'C:\ofbiz\applications\accounting\webapp\accounting\agreement\AgreementForms.xml' open. The code displays an XML configuration for a single-form named 'EditAgreement'. It includes fields for agreementId, productId, agreementTypeId, partyIdFrom, roleTypeIdFrom, and roleTypeIdTo, each with specific titles and descriptions. The 'agreementTypeId' field uses a dropdown menu with entity-options for 'AgreementType'. The 'partyIdFrom' and 'roleTypeIdFrom' fields also use dropdown menus with entity-options for 'RoleType'. The 'roleTypeIdTo' field uses a dropdown menu with entity-options for 'RoleType'. The 'target' attribute for most fields is set to 'updateAgreement' with a title of '' and a default map name of 'agreement'. An 'alt-target' block handles the case where 'agreement' is null, pointing to 'createAgreement'.

```
<form name="EditAgreement" type="single">
    target="updateAgreement" title="" default-map-name="agreement">
        default-title-style="tableheadtext" default-widget-style="inputBox" default-tooltip-style="tabletext">
    <alt-target use-when="agreement==null" target="createAgreement"/>
    <auto-fields-entity entity-name="Agreement" default-field-type="edit"/>
    <field name="agreementId" title="${uiLabelMap.AccountingAgreementId}"><display/></field>
    <field name="productId" title="${uiLabelMap.AccountingProductId}">
        <lookup target-form-name="LookupProduct"/>
    </field>
    <field name="agreementTypeId" title="${uiLabelMap.AccountingAgreementTypeId}">
        <drop-down allow-empty="true">
            <entity-options description="${description}" entity-name="AgreementType" key-field-name="agreementTypeId"/>
        </drop-down>
    </field>
    <field name="partyIdFrom" title="${uiLabelMap.AccountingPartyIdFrom}">
        <lookup target-form-name="LookupPartyName"/>
    </field>
    <field name="roleTypeIdFrom" title="${uiLabelMap.AccountingRoleTypeIdFrom}">
        <drop-down allow-empty="true">
            <entity-options description="${description}" entity-name="RoleType" key-field-name="roleTypeId"/>
        </drop-down>
    </field>
    <field name="roleTypeIdTo" title="${uiLabelMap.AccountingRoleTypeIdTo}">
        <drop-down allow-empty="true">
            <entity-options description="${description}" entity-name="RoleType" key-field-name="roleTypeId"/>
        </drop-down>
    </field>
    <field name="partyIdTo" title="${uiLabelMap.AccountingPartyIdTo}">
        <lookup target-form-name="LookupPartyName"/>
    </field>

```

Figure 58

```
target="updateAgreement" title="" default-map-name="agreement">
```

Figure 59

It says that its next target should be updateAgreement, but what is updateAgreement? Where is it ? the Controller would “always” answer .

But it also says :

```
<alt-target use-when="agreement==null" target="createAgreement"/>
```

Figure 60

It is clear now that if the agreement we are dealing with is “new” ,i.e., `=null` , so our target will be “`createAgreement`” , else our target is “`updateAgreement`” .

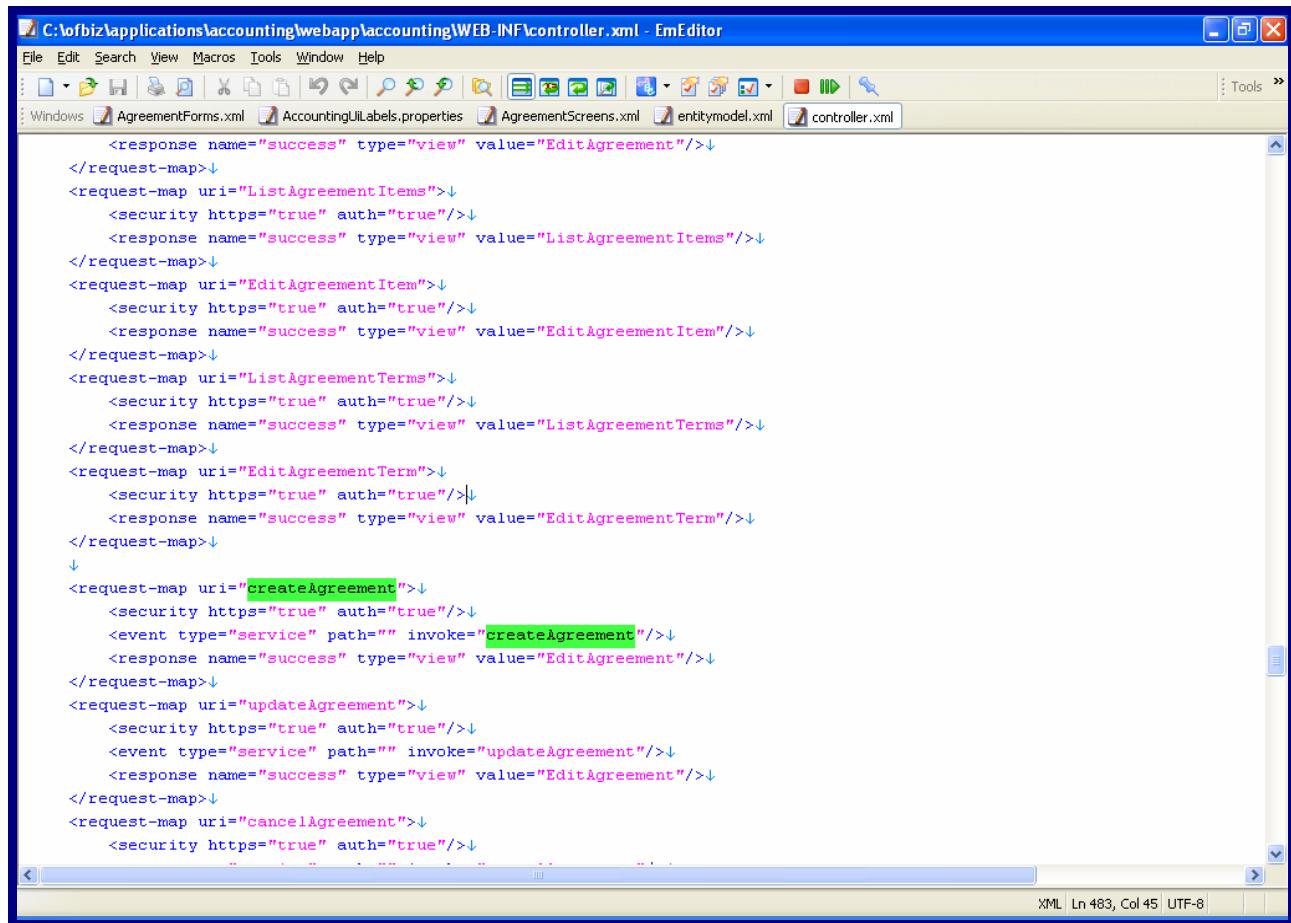
The rest of this form is a normal declaration for the fileds , as we saw earlier in the `FindAgreement` form.

Let us now follow the `createAgreement` and the `updateAgreement`.

1) Assume we are creating a new agreement, that is we pressed on the button [create Agreement] in Figure 56 , and started entering the fields of the `EditAgreement` .

Thus, it will find that the condition “`agreement == null`” returns true .

So, the request goes to the controller , for `createAgreement`.



```
C:\ofbiz\applications\accounting\webapp\accounting\WEB-INF\controller.xml - EmEditor
File Edit Search View Macros Tools Window Help
Windows AgreementForms.xml AccountingUILabels.properties AgreementScreens.xml entitymodel.xml controller.xml
<response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
<request-map uri="ListAgreementItems">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementItems"/>↓
</request-map>↓
<request-map uri="EditAgreementItem">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementItem"/>↓
</request-map>↓
<request-map uri="ListAgreementTerms">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="ListAgreementTerms"/>↓
</request-map>↓
<request-map uri="EditAgreementTerm">↓
    <security https="true" auth="true"/>↓
    <response name="success" type="view" value="EditAgreementTerm"/>↓
</request-map>↓
↓
<request-map uri="createAgreement">↓
    <security https="true" auth="true"/>↓
    <event type="service" path="" invoke="createAgreement"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
<request-map uri="updateAgreement">↓
    <security https="true" auth="true"/>↓
    <event type="service" path="" invoke="updateAgreement"/>↓
    <response name="success" type="view" value="EditAgreement"/>↓
</request-map>↓
<request-map uri="cancelAgreement">↓
    <security https="true" auth="true"/>↓
    ...
```

Figure 61

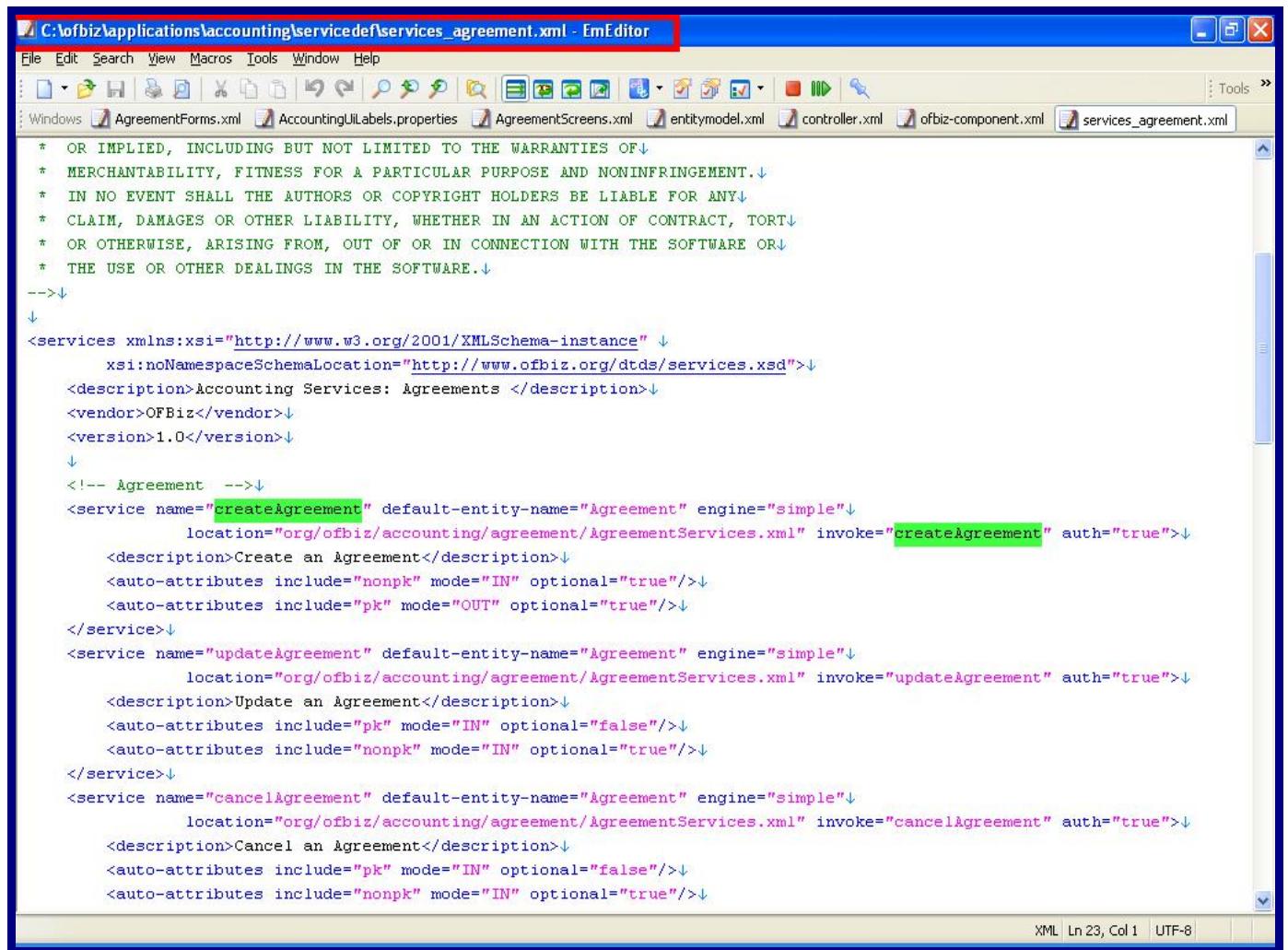
We found the `createAgreement` , but it is of type service! “Not a view”.

and it invokes a service called `createAgreement` , and on success , i.e., if we didn't have any errors or problems, it would be redirected to the `EditAgreement` page.

```
<event type="service" path="" invoke="createAgreement"/>↓
```

Figure 62

Now , and by default, the controller will try to search for the `createAgreement` service , in the `servicedef` directory. It will then find the definition for this service” `createAgreement`” in the `service_agreement.xml` file .



The screenshot shows the `C:\ofbiz\applications\accounting\servicedef\services_agreement.xml` file open in the EmEditor XML editor. The file contains XML code defining services for accounting agreements. A specific line of code is highlighted in green: `invoke="createAgreement" auth="true">`. This indicates that the `createAgreement` service will be invoked with authentication enabled.

```
* OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.↓
* IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY↓
* CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT↓
* OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR↓
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.↓
-->↓
↓
<services xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↓
    xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/services.xsd">↓
<description>Accounting Services: Agreements </description>↓
<vendor>OFBiz</vendor>↓
<version>1.0</version>↓
↓
<!-- Agreement -->↓
<service name="createAgreement" default-entity-name="Agreement" engine="simple"↓
    location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="createAgreement" auth="true">↓
    <description>Create an Agreement</description>↓
    <auto-attributes include="nonpk" mode="IN" optional="true"/>↓
    <auto-attributes include="pk" mode="OUT" optional="true"/>↓
</service>↓
<service name="updateAgreement" default-entity-name="Agreement" engine="simple"↓
    location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="updateAgreement" auth="true">↓
    <description>Update an Agreement</description>↓
    <auto-attributes include="pk" mode="IN" optional="false"/>↓
    <auto-attributes include="nonpk" mode="IN" optional="true"/>↓
</service>↓
<service name="cancelAgreement" default-entity-name="Agreement" engine="simple"↓
    location="org/ofbiz/accounting/agreement/AgreementServices.xml" invoke="cancelAgreement" auth="true">↓
    <description>Cancel an Agreement</description>↓
    <auto-attributes include="pk" mode="IN" optional="false"/>↓
    <auto-attributes include="nonpk" mode="IN" optional="true"/>↓

```

Figure 63

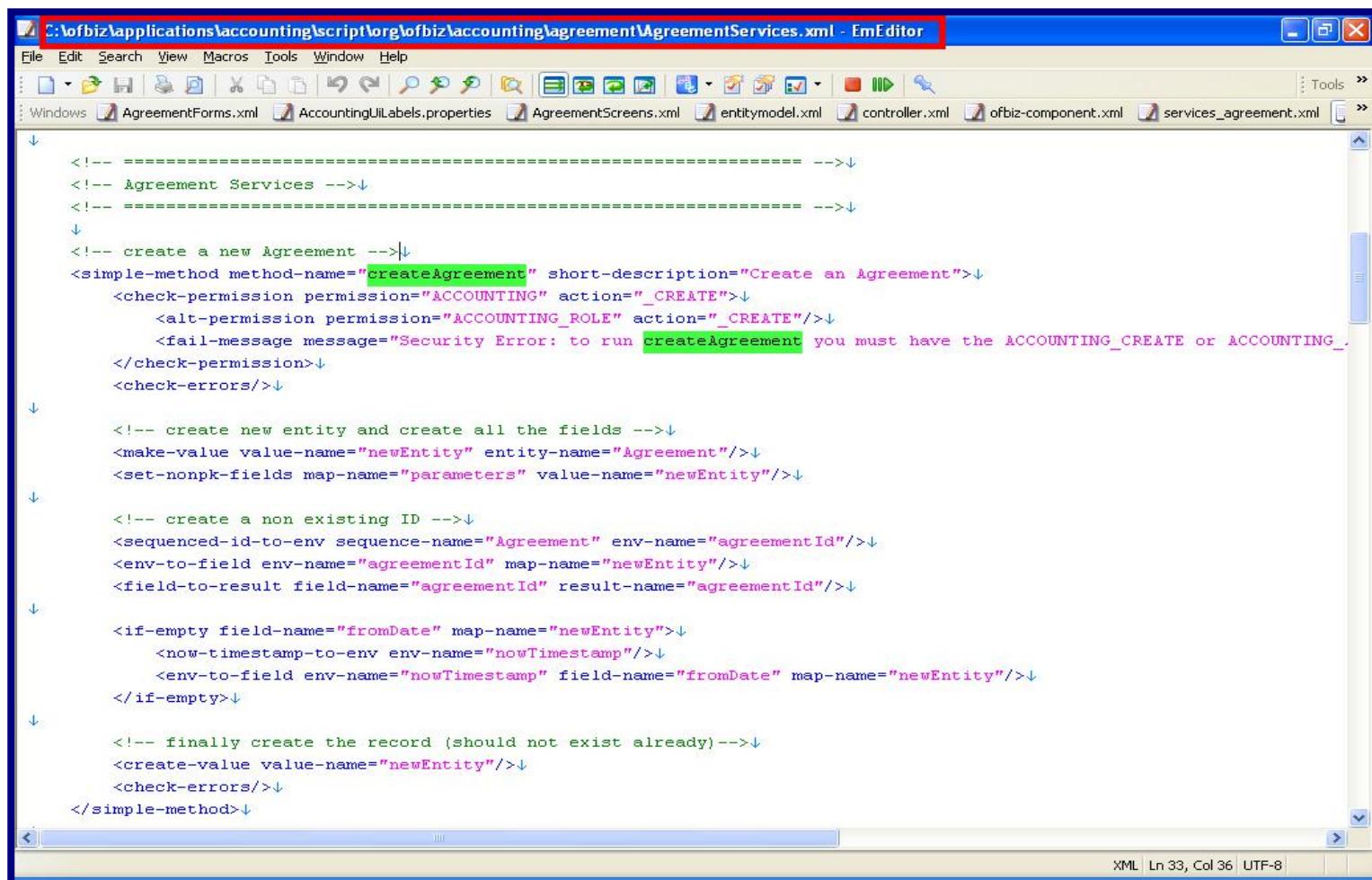
Here it is clear, the `createAgreement` service will be requested It will be servicing the `Agreement` entity as mentioned in the `default-entity-name` .  
It is of type simple, it will invoke a simple method called `createAgreement`, and its location is provided.

The <auto-attributes> will govern the mode and existence of the non-pk and pk attributes of the entity Agreement.

You can follow the again refer to the service engine document on the OFBiz website.

Then we would follow the path , to see the implementation of the createAgreement simple method.

And finally here is the createAgreement service implemented with the OFBiz mini-language.



The screenshot shows the EmEditor application window displaying an XML file named 'AgreementServices.xml'. The file contains the implementation of the 'createAgreement' simple method. The code includes permission checks, value assignments, and field mappings. The XML is color-coded for readability, with green for tags and pink for attributes and values. The status bar at the bottom right indicates 'XML | Ln 33, Col 36 | UTF-8'.

```
<!-- ===== -->
<!-- Agreement Services -->
<!-- ===== -->
<!-- create a new Agreement -->
<simple-method method-name="createAgreement" short-description="Create an Agreement">
    <check-permission permission="ACCOUNTING" action="_CREATE">
        <alt-permission permission="ACCOUNTING_ROLE" action="_CREATE"/>
        <fail-message message="Security Error: to run createAgreement you must have the ACCOUNTING_CREATE or ACCOUNTING_ROLE permission" />
    </check-permission>
    <check-errors/>

    <!-- create new entity and create all the fields -->
    <make-value value-name="newEntity" entity-name="Agreement"/>
    <set-nonpk-fields map-name="parameters" value-name="newEntity"/>

    <!-- create a non existing ID -->
    <sequenced-id-to-env sequence-name="Agreement" env-name="agreementId"/>
    <env-to-field env-name="agreementId" map-name="newEntity"/>
    <field-to-result field-name="agreementId" result-name="agreementId"/>

    <if-empty field-name="fromDate" map-name="newEntity">
        <now-timestamp-to-env env-name="nowTimestamp"/>
        <env-to-field env-name="nowTimestamp" field-name="fromDate" map-name="newEntity"/>
    </if-empty>

    <!-- finally create the record (should not exist already) -->
    <create-value value-name="newEntity"/>
    <check-errors/>
</simple-method>
```

Figure 64

```

<!-- create new entity and create all the fields -->↓
<make-value value-name="newEntity" entity-name="Agreement"/>↓
<set-nonpk-fields map-name="parameters" value-name="newEntity"/>↓

```

Figure 65

As shown,

Make-value means we want to create a new row in the database of type “Agreement”. Parameters “fields” are sent/sent/get using Maps. So, now our map that will contain the fields of the new created entity is called . “newEntity” .

This “newEntity” map is taking the values of all the non-pk from another map called “parameters” . The parameters map holds the values/parameters of the form, “i.e., from the parameters entered by the user in the form “.

```

<!-- create a non existing ID -->↓
<sequenced-id-to-env sequence-name="Agreement" env-name="agreementId"/>↓
<env-to-field env-name="agreementId" map-name="newEntity"/>↓
<field-to-result field-name="agreementId" result-name="agreementId"/>↓

```

Figure 66

Here is creating the sequence for the agreementId field which is the primary key of the Agreement entity, then it is filling the agreementId in the “newEntity” map.

```

<if-empty field-name="fromDate" map-name="newEntity">↓
  <now-timestamp-to-env env-name="nowTimestamp"/>↓
    <env-to-field env-name="nowTimestamp" field-name="fromDate" map-name="newEntity"/>↓
</if-empty>↓

```

Figure 67

The filed fromDate should not be null “you do not expect an agreement not to have the date taken” . Thus, if the user didn’t fill it, it will be filled with the current time value, using the “nowTimesamp” env-name .

```

<!-- finally create the record (should not exist already)-->↓
<create-value value-name="newEntity"/>↓
<check-errors/>↓
</simple-method>↓

```

Figure 68

And finally, the new row is created in the database.

---

Here is the createAgreement :

The screenshot shows a Microsoft Internet Explorer window with the following details:

- Title Bar:** OFBiz: Accounting Manager: PageTitleEditAgreement - Microsoft Internet Explorer
- Toolbar:** File, Edit, View, Favorites, Tools, Help
- Address Bar:** Address: https://localhost:8443/accounting/control/EditAgreement
- Header:** OPEN FOR BUSINESS OFBiz.org, Welcome THE ADMINISTRATOR! 2005-08-30 17:09:26.468, English, Set
- Menu Bar:** Accounting, Catalog, Content, Contract, Example, Facility, Manufacturing, Marketing, Order, Party, Shark, WebTools, WorkEffort
- Submenu Bar:** Main, Agreements, Billing Accounts, Invoices, Payments, Chart of Accounts, Fixed Assets, Manual Tx, Logout
- Content Area:**
  - Form title: [ID:] [Create Agreement]
  - Form fields:
    - Agreement Id
    - Product Id
    - Party Id From
    - Party Id To
    - Role Type Id From
    - Role Type Id To
    - Agreement Type Id
    - Agreement Date
    - From Date
    - Thru Date
    - Description
    - Text Data
  - Submit button
- Validation Logos:** W3C CSS and W3C XHTML 1.0
- Status Bar:** Local intranet

Figure 69

And here after submitting the button “note that we saw that button in at the end of the “EditAgreement” form.

The Following Occurred:

- The action was performed successfully.

**Agreement** **Agreement Items** **Agreement Terms**

ID: 10050  
[create Agreement]

Agreement Id	10050
Product Id	
Party Id From	
Party Id To	
Role Type Id From	Requesting Party
Role Type Id To	
Agreement Type Id	
Agreement Date	
From Date	2005-08-30 17:11:32.421
Thru Date	
Description	
Text Data	

Submit

W3C CSS W3C XHTML 1.0

Copyright (c) 2001-2005 The Open For Business Project - www.ofbiz.org  
Powered By OFBiz

Figure 70

Note the control/createAgreement request , because the agreement== null was true, since the user hasn't created the agreement yet.

Now, and after we have created a new row in the database , in the table Agreement , whose ID is 10050 . Now a click on the submit button, will check and find that agreement == null is not a true anymore, since the agreement will read the values of the fields from the “parameters” map . Thus it will request the controller > updateAgreement is a service> looks for it in the servicedef > finds out its type and location > looks for its implementation > perform the service >data stored in the database > gets back to the page as shown below.

OFBiz: Accounting Manager: PageTitleEditAgreement - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address: https://localhost:8443/accounting/control/updateAgreement

Accounting Manager Application

Main Agreements Billing Accounts Invoices Payments Chart of Accounts Fixed Assets Manual Tx Logout

The Following Occurred:

. The action was performed successfully.

Agreement Agreement Items Agreement Terms

[ID:10050] [Create Agreement]

Agreement Id: 10050  
Product Id:   
Party Id From:   
Party Id To:   
Role Type Id From: Requesting Party   
Role Type Id To:   
Agreement Type Id:   
Agreement Date:   
From Date: 2005-08-30 17:11:32.421   
Thru Date:   
Description:   
Text Data:

Submit

W3C CSS W3C XHTML 1.0

Done Local intranet

Figure 71

Notice that : these

Agreement Agreement Items Agreement Terms

[ID:10050] [Create Agreement]

Agreement Id: 10050  
Product Id:   
Party Id From:

Figure 72

were not there in the createAgreement Figure,Figure 69, so they must be attached to a particular Agreement , in other words, there should be an agreement , for them to appear.

So we would check the AgreementScreen again, we would find it is using main-decorator pattern,

```

<screen name="EditAgreement">↓
  <section>↓
    <actions>↓
      <set field="title" value="Edit Agreement"/>↓
      <set field="titleProperty" value="PageTitleEditAgreement"/>↓
      <set field="headerItem" value="agreement"/>↓
      <set field="tabButtonItem" value="EditAgreement"/>↓
    ↓
      <set field="agreementId" from-field="parameters.agreementId"/>↓
      <entity-one entity-name="Agreement" value-name="agreement"/>↓
    </actions>↓
  <widgets>↓
    <decorator-screen name="CommonAgreementDecorator">↓
      <decorator-section name="body">↓
        <container>↓
          <link target="EditAgreement" text="${uiLabelMap.AccountingNewAgreement}" style="buttontext" />↓
        </container>↓
        <include-form name="EditAgreement" location="component://accounting/webapp/accounting/agreement/EditAgreement.ftl" />↓
      </decorator-section>↓
    </decorator-screen>↓
  </widgets>↓

```

Figure 73

and having a look into the main-decorator pattern , we would see it is including an ftl file called AgreementTabBar.ftl .

```

C:\ofbiz\applications\accounting\widget\AgreementScreens.xml - EmEditor
File Edit Search View Macros Tools Window Help
File Edit Search View Macros Tools Window Help
Windows AgreementForms.xml AccountingUILabels.properties AgreementScreens.xml entitymodel.xml controller.xml ofbiz-component.xml services_agreement.xml
HERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
SE OR OTHER DEALINGS IN THE SOFTWARE.↓

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↓
 xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/widget-screen.xsd">↓
<screen name="CommonAgreementDecorator">↓
  <section>↓
    <widgets>↓
      <decorator-screen name="main-decorator" location="component://accounting/widget/CommonScreens.xml">↓
        <decorator-section name="body">↓
          <section>↓
            <!-- do check for ACCOUNTING, _VIEW permission -->↓
            <condition>↓
              <if-has-permission permission="ACCOUNTING" action="_VIEW"/>↓
            </condition>↓
          <widgets>↓
            <platform-specific>↓
              <html><html-template location="component://accounting/webapp/accounting/agreement/AgreementTabBar.ftl" />↓
            </platform-specific>↓
            <container>↓
              <label style="head1">${uiLabelMap.CommonId}: ${agreement.agreementId} ${agreement.description}</label>↓
            </container>↓
          </widgets>↓
          <fail-widgets>↓
            <label style="head3">${uiLabelMap.AccountingViewPermissionError}</label>↓
          </fail-widgets>↓
        </decorator-section>↓
      </decorator-screen>↓
    </widgets>↓
  </section>↓
</screen>↓

```

Figure 74

and inside the AgreementTabBar.ftl file :

C:\ofbiz\applications\accounting\webapp\accounting\agreement\AgreementTabBar.ftl - EmEditor

```

File Edit Search View Macros Tools Window Help
Windows AgreementForms.xml AccountingJILabels.properties AgreementScreens.xml entitymodel.xml controller.xml ofbiz-component.xml services_agreementTabBar.ftl
*↓
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS ↓
* OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF ↓
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. ↓
* IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY ↓
* CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT ↓
* OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR ↓
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.↓
*↓
*@author David E. Jones (jonesde@ofbiz.org)↓
*@version $Rev: 5462 $↓
*@since 2.2↓
-->↓
↓
<#assign unselectedClassName = "tabButton">↓
<#assign selectedClassMap = (page.tabButtonItem?default("void") : "tabButtonSelected")>↓
↓
<if agreement?has_content>↓
    <div class='tabContainer'>↓
        <a href=<@ofbizUrl>/EditAgreement?agreementId=${agreement.agreementId}</@ofbizUrl>" class="${selectedClassMap}<@ofbizUrl>">
        <a href=<@ofbizUrl>/ListAgreementItems?agreementId=${agreement.agreementId}</@ofbizUrl>" class="${selectedClassMap}<@ofbizUrl>">
        <a href=<@ofbizUrl>/ListAgreementTerms?agreementId=${agreement.agreementId}</@ofbizUrl>" class="${selectedClassMap}<@ofbizUrl>">
    </div>↓
</if>↓
+↓

```

Figure 75

It is clear that these would appear only when there is an agreement because it is checking if the agreement has content ,i.e., it is not null, and it shows them” the button we have seen” only when this condition satisfies.

As for the ID : 10050 That had a green circle in Figure 72 , it also came from the commonAgreementDecorator :

C:\ofbiz\applications\accounting\widget\AgreementScreens.xml - EmEditor

```

File Edit Search View Macros Tools Window Help
Windows controller.xml web.xml fieldTypeoracle.xml AgreementScreens.xml
mlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↓
i:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/widget-screen.xsd"↓
n name="CommonAgreementDecorator">↓
action>↓
<widgets>↓
    <decorator-screen name="main-decorator" location="component://accounting/widget/CommonScreens.xml">↓
        <decorator-section name="body">↓
            <section>↓
                <!-- do check for ACCOUNTING, _VIEW permission -->↓
                <condition>↓
                    <if-has-permission permission="ACCOUNTING" action="_VIEW"/>↓
                </condition>↓
                <widgets>↓
                    <platform-specific>↓
                        <html><html-template location="component://accounting/webapp/accounting/agreement/AgreementTabBar.ftl" />
                    </platform-specific>↓
                    <container>↓
                        <label style="head1">[$(uiLabelMap.CommonId):${(agreement.agreementId)}] ${agreement.description}</label>
                    </container>↓
                </widgets>↓
            </section>↓
            <decorator-section-include name="body"/>↓
        </decorator-section>↓
    </decorator-screen>↓
</widgets>↓
<fail-widgets>↓
    <label style="head3">$(uiLabelMap.AccountingViewPermissionError)</label>↓
</fail-widgets>↓
</section>↓
</decorator-section>↓
</decorator-screen>↓
</widgets>↓
section>↓

```

Figure 76

Finally , we would see this Agreement in our list.

The screenshot shows a Microsoft Internet Explorer window titled "OFBiz: Accounting Manager: Find Agreements - Microsoft Internet Explorer". The address bar shows the URL <https://localhost:8443/accounting/control/FindAgreement>. The main content area is titled "Accounting Manager Application" and "Agreements". It features a search form with fields for "Agreement Id", "Product Id", "Party Id From", "Party Id To", "Agreement Type Id", and "From Date". Below the form is a table listing agreements. The table has columns: Edit, Product Id, Party Id From, Party Id To, Role Type Id To, Agreement Type Id, From Date, Thru Date, and Description. One row in the table, corresponding to the agreement with ID 10050, has its "Edit" link circled in green. The table data is as follows:

Edit	Product Id	Party Id From	Party Id To	Role Type Id To	Agreement Type Id	From Date	Thru Date	Description
10000	GZ-1000			Person		2005-08-14 10:56:08.554		[Cancel]
10043						2005-08-23 11:27:19.265		[Cancel]
10050						2005-08-30 17:11:32.421		[Cancel]
AGR_TEST		Company	DemoSupplier	Supplier	Purchase			Agreement for DemoSupplier [Cancel]
1000		Company	BigSupplier	Supplier	Purchase			Purchasing Agreement with BigSupplier [Cancel]
1001		Company	EuroSupplier	Supplier	Purchase			Purchasing Agreement with EuroSupplier-Milan [Cancel]
1002		Company	EuroSupplier	Supplier	Purchase			Purchasing Agreement with EuroSupplier-New York [Cancel]
10030						2005-08-17 19:04:34.875		[Cancel]

Figure 77

## **Summary :**

OFBiz uses the three-tiers architecture in its model and a controller “main-servlet” to control and forward the requests to the application, whether this request is for a particular service or page. For building any application you need to build up these layers.

1) Data layer : represents the database, your stored data.

→ Design your database tables

→ Build the data layer in the entitydef folder:

→ entitygroup.xml file to define the tables name.

→ entitymodel.xml to implement the defined tables, their pk's, field types and relationships.

2) Business Logic Layer: represents what services applied on the database.

→ Decide what services you need.

→ Build this layer:

→ servicedef folder : define all the services ,their types, the methods they invoke, their inputs and outputs..etc.

→ Implement the services:

→ script folder : if the service can be implemented with xml.

→ src folder : if the service to be implemented with java.

3) Presentation layer : display pages “user-interface”

→ Decide what pages you want, what do you want to display to the user.

→ Build this layer:

→ widget folder : contains the screens that represent the application pages.

→ webapp/”appfolder” folder : contains the forms that might be included in some screens.

4) Controller :

→ It contains all the URLs related to the application, it receives the requests and forwards the requests to their location. It also defines the types of the requests, the handlers for the different events,...etc.

→ contains the path for all the screens.

→ Located in : webapp/WEB-INF/controller.xml .

→ When ever you add any new service or screen, you need to include it in the controller.

[1] Taken from <http://opensourcestrategies.com/ofbiz>

[1]\* Taken by word from <http://opensourcestrategies.com/ofbiz>

**THE END**