Open OTC Viewer

Desarrollo de Aplicaciones

Autor: Luis Ricardo Luzondo Oyón
Consultor: Gregorio Robles Martinez
Tutor externo: Rubén Jimeno Corella
Fecha: XXXXXXXXXX

Copyright (C) 2015 Luis Ricardo Luzondo Oyón

Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".

Resumen del proyecto

El proyecto **Open OTC Viewer** consiste en el desarrollo de una aplicación web de gestión de operaciones financieras de tipo Interest Rate Swap (IRS). Este tipo de producto derivado financiero es uno de los más usados y difundidos dentro de los mercados OTC. Open OTC Viewer permite la visualización de operaciones de tipo IRS, se trata de una aplicación desarrollada sobre Java EE que junto a una base de datos Mysql y un servidor web permitirá al usuario disponer de la información de su cartera. El producto final se distribuye bajo licencia **GNU General Public License**.

Índice

Introducción

Motivación

El sector de los mercados de inversión está dominado por una serie de empresas que ofrecen soluciones muy completas y sofisticadas a unos precios muy elevados. Esto provoca que sólo las grandes empresas de inversión puedan acceder a este tipo de soluciones. Siempre se ha tratado de un sector muy ocultista, apenas se encuentra información en la red, no existen comunidades Open Source dedicadas al este sector, Open OTC Viewer rompe con la tendencia.

Si bien Open OTC Viewer no se puede comparar con las grandes soluciones como puede ser Murex o Calypso, ya que se tratan de un software muy sofisticado y completo, Open OTC Viewer se presenta como un complemento a estas soluciones que puede ser usado tanto por grandes corporaciones como por pequeños inversores para abaratar la inversión.

Propósito del proyecto

El propósito del proyecto es proporcionar a la comunidad de pequeños inversores una plataforma online de gestión de operaciones financieras OTC[i] de tipo Interest Rate Swap[ii].

Open OTC Viewer se presenta como una solución Open Source dentro de un sector tradicionalmente ocultista, por lo proporciona a inversores otra opción de gestión que no conlleve una gran inversión en forma de licencia de software privativo.

Open OTC Viewer trabaja con ficheros FpML[iii], convirtiéndolos a formato MxML e importando en la base de datos para su presentación desde una interfaz web. Al trabajar con estándares derivados de XML como son FpML y MxML, proporcionamos a Open OTC Viewer una compatibilidad con el resto de soluciones disponibles en el mercado.

Objetivo del proyecto

El objetivo es crear una plataforma online de gestión de operaciones Interest Rate Swaps Open Source que pueda ser fácilmente gestionada por cualquier inversor.

Open OTC Viewer pretende crear en torno al proyecto una comunidad de usuarios Open Source participativa que permita hacer crecer el software con calidad y seguridad. El papel de la comunidad tiene un peso importante dentro del proyecto, sólo una comunidad “sana” podrá proporcionar al proyecto un ciclo de vida continuo, en el que se vayan publicando nuevos módulos, nuevas funcionalidades, para poco a poco ir asemejándose más a las grandes soluciones privativas del sector.

Ya que Open OTC Viewer se presenta como software Open Source, para el desarrollo del producto se van a usar tecnologías afines, como son Mysql[iv], Glassfish[v], Egit[vi], Eclipse[vii] y FpML.

Motivación personal

Desde que inicié mi carrera profesional en Accenture me introdujeron en el mundo de la banca de inversión, en concreto con el programa para gestión de tesorería Murex. Desde el primer momento descubrí que es un sector y en concreto un aplicativo bastante complejo e interesante porque implica desarrollar conocimientos técnicos pero también funcionales acerca de mercados financieros y en concreto operaciones OTC (Over the counter).

Fue a través de un proyecto concreto que conocí el lenguaje FpML y sus usos dentro del sector. Se trata del estándar basado en XML más difundido y usado para representar operaciones financieras. Por este motivo, elegí un proyecto que me permitiera aprender más sobre FpML y sobre el lenguaje basado en XML usado para el mismo fin pero creado por la empresa Murex[viii], MxML. La solución fue un conversor de FpML a MxML usando lenguaje de transformación XSL conjuntamente con Java. Con el fin de incluir alguna tecnología más, y por el interés de recordar la programación en Java EE (la cual tenía un poco olvidada) decidí incluir un módulo web que permitiera visualizar la operativa contratada por un usuario concreto.

Estructura de la memoria

El presente documento tiene como propósito principal el desarrollo de las diferentes fases que forman el desarrollo del producto final.

Se hará referencia a la planificación temporal y económica del ciclo de vida del software, así como las conclusiones y las referencias bibliográficas consultadas.

El documento está dividido en capítulos que describen toda la información recogida en el proyecto desarrollado (fases del proyecto, recopilación de requisitos, tecnologías usadas, modelo de negocio del producto, etc).

Por último, encontraremos un anexo con toda la información externa relativa al proyecto.

Estado del arte

El avance tecnológico ha producido un cambio en la forma de inversión, actualmente se puede invertir a cualquier hora del día en diferentes países con una simple conexión a Internet. El sistema de inversiones ha cambiado radicalmente con la entrada de las nuevas tecnologías, gracias a sistemas en tiempo real, que proporcionan información casi al instante de las variaciones de los distintos índices, divisas, etc.

Esto ha supuesto un gran incremento de los inversores, ya que este mercado se ha abierto a todo el mundo que disponga de una conexión a Internet, provocando un importante auge del número de transacciones realizadas a lo largo de un día.

Open OTC Viewer se adapta perfectamente a esta nueva situación, y proporciona una diferenciación al tratarse de software Open Source. Cualquier usuario de Open OTC Viewer podrá ubicar su servidor en un dominio y hacerlo accesible a lo largo del mundo con un coste reducido. De esta manera se convierte en un sistema online con gran disponibilidad, permitiendo su uso en diferentes geografías pero trabajando sobre un mismo ambiente.

Dentro de los mercados financieros derivados, los Interest Rate Swaps (en adelante Swap) son uno de los productos más difundidos. Fueron introducidos en el mercado en 1981 cuando IBM y el Banco Mundial contrataron el primer Swap.

La inclusión de trabajo con FpML, que es el estándar más difundido actualmente para la representación de operaciones financieras derivadas, proporciona a Open OTC Viewer la tecnología Open Source con más proyección de futuro dentro de los entandares XML para derivados financieros.

Open OTC Viewer al tratarse de una aplicación web, podrá ser consultada desde cualquier dispositivo con acceso a Internet. De esta manera abrimos el producto a las nuevas tecnologías, dispositivos móviles bajo cualquier sistema operativo, tablets, terminales Windows, GNU/Linux, Apple. Cualquier tecnología es compatible con Open OTC Viewer ya que su publicación en la web mediante HTML, lenguaje compatible con todo tipo de tecnologías.

Herramientas usadas para el desarrollo

A continuación mostramos las herramientas usadas para el desarrollo de Open OTC Viewer.

Eclipse: Versión 3.8.1. Se distribuye bajo licencia “Eclipse Public License[ix]”. Eclipse se ha utilizado como IDE para el desarrollo de Open OTC Viewer.

Módulos instalados en Eclipse:

- Eclipse Java Development Tools. Versión 3.8.1. Permite programar en Java.
- Eclipse Java EE Developer Tools. Versión 3.3.2. Permite programación Java enfocada a la web (uso de sesiones, servlets, jps, etc).
- Eclipse EGIT: Versión 1.3.0. EGIT es un software de control de versiones, está basado en JGIT, que a su vez es una versión Java de GIT. Se integra en Eclipse, permite la publicación del código fuente para hacerlo accesible a la comunidad de usuarios. Se distribuye bajo licencia “Eclipse Public License”.
- Conector mysql-connector-java-5.1.6-bin.jar: Una vez importado en Eclipse en el workspace del proyecto, nos permite realizar conexiones a una base de datos MySQL externa.
- GlassFish: Versión 3.1.2. Se distribuye bajo las licencias: “License Common Development and Distribution License” y “GNU General Public License”. Se trata de un servidor web que se integra en Eclipse permitiendo la ejecución de código Java enfocado a la web.

MySQL: MySQL Server 5.5 y MySQL Workbench 6.0. MySQL Server proporciona un gestor de base de datos completo y Open Source. Ha sido desarrollado por Oracle y se distribuye

bajo licencia GPL (Versión 2).

Ubuntu: Sistema operativo usado como base para la instalación del software arriba mencionado. Versión Ubuntu 14.04 LTS. Se trata de una de las distribuciones GNU/Linux más difundidas, está desarrollado por Canonical y se distribuye bajo licencia GPL.

Como podemos observar, todo el software usado es compatible con los requisitos que hemos mencionado en puntos anteriores.

Estudio viabilidad

En este apartado se definirán los diferentes aspectos de Open OTC Viewer y las herramientas usadas para su creación.

Definición y alcance del proyecto

El proyecto consiste en la creación de una pieza programada en Java que hace uso del lenguaje de transformación para transformar ficheros FpML en MxML. Los ficheros FpML y MxML contienen todos los campos necesarios para definir una operación de tipo Interest Rate Swap, estos valores serán cargados en una base de datos MySQL. Esta base de datos contendrá varias tablas que ayudan a la transformación, y otras dedicadas a la gestión de la interfaz web. La versión web que proporciona Open OTC Viewer permitirá a los diferentes usuarios del aplicativo la visualización mediante el uso de un navegador web.

El proyecto se enmarca dentro del sector financiero, como sabemos, este sector está en pleno auge desde varias décadas atrás, lo que ha permitido la creación de estándares que ayuden y faciliten la negociación, uno de ellos es el FpML. Dentro de la banca de inversión en mercados no regulados (OTC), los productos Interest Rate Swap son los más difundidos.

Los ficheros FpML y MxML que contienen la información de la operación son archivos de texto extensos y difíciles de leer para los usuarios, es por esto que Open OTC Viewer proporciona la interfaz web. A través de ella, los usuarios podrán visualizar en formato tabla los datos importantes de su operativa contratada.

Open OTC Viewer está pensado para instituciones financieras que ya hagan uso de alguna aplicación de gestión de tesorería, como puede ser Murex o Calypso[x]. Ambas herramientas permiten una gestión eficaz de todo lo relativo a la tesorería de una institución, pero tienen un alto coste de licencia, instalación y mantenimiento. Open OTC Viewer proporciona una herramienta de gestión complementaria a las anteriores y compatible con ellas.

Ya que se trata de una aplicación que se apoya tanto en bases de datos como en servidores web, será necesario disponer de soporte técnico para dichas tecnologías.

Open OTC Viewer podrá mantener en la base de datos las operaciones swap de varios brokers o traders. La base de datos asocia las operaciones al broker que las contrata, por lo que podrá ser usado por uno o varios usuarios al mismo tiempo, cada uno visualizando únicamente su operativa. Una de las grandes ventajas de Open OTC Viewer es que está deslocalizado, es decir, se podrá acceder desde cualquier lugar del mundo que disponga de conexión a Internet. Al tratarse de peticiones web se puede securizar de manera sencilla y barata.

Definición de requisitos

En el siguiente apartado describimos los diferentes requisitos que Open OTC Viewer debe cumplir.

Requisitos técnicos:

- El aplicativo debe ser accesible desde cualquier navegador y desde cualquier plataforma.
- El sistema gestor de la base de datos debe permitir el almacenamiento y el acceso desde el aplicativo web. Además de lo anterior, el administrador de la base datos debe tener permisos para modificar la base de datos. Se buscará una solución Open Source como gestor de base de datos. Para el desarrollo se ha usado MySQL.
- El servidor web donde se ubique el aplicativo debe ser seguro y tener gran disponibilidad para atender las peticiones de los usuarios. Se buscará una solución Open Source como servidor web. Para el desarrollo se ha usado GlassFish Server.
- Seguridad. Control de usuarios de acceso a la web. Protección de los servidores de base de datos. En un principio se usará el protocolo HTTP para realizar las peticiones web, pero se abre la posibilidad se usar el protocolo HTTPS para cifrar la comunicación y permitir el uso de certificados digitales. También se abre la posibilidad de usar conexiones VPN.
- Control de versiones. Open OTC Viewer busca crear una gran comunidad de usuarios que apoyen en el desarrollo. Para ello, es fundamental la transparencia, para ello haremos uso de un repositorio de código Open Source como puede ser Git.
- Sistema operativo. Tanto el servidor web como la base de datos funcionarán bajo un sistema operativo de licencia open source como puede ser cualquier versión de GNU/Linux. Para el desarrollo se ha utilizado una versión de Ubuntu Server.
- Desarrollo. Para el desarrollo del aplicativo usaremos el IDE Eclipse, el cual es Open Source. El software de control de versiones utilizado es Egit, el cual está basado en Git, se trata de un software Open Source que se integra con el IDE Eclipse.

Requisitos económicos:

- Se buscarán siempre las soluciones/tecnologías Open Source con el fin de abaratar presupuestos.
- El servidor web deberá ser accesible mediante una dirección web. Para ello, si no se dispone de una infraestructura de red, se deberá contratar un dominio. Se buscará la opción más acorde y barata a las necesidades del cliente. Normalmente el contrato con la compañía proveedora de servicios se acordará según el número de usuarios que harán uso de la web.

Requisitos operativos:

- La interfaz web debe ser simple e intuitiva. Debe permitir la visualización de al menos las partes más importantes que definen un swap.
- Será necesario la contratación de un administrador de sistemas para la gestión de los servidores y la seguridad.

- La administración de la base de datos y del servidor web debe ser accesible por el administrador en cualquier momento y desde cualquier lugar.

Requisitos legales:

- Se usarán licencias Open Source para el desarrollo del aplicativo. Siempre cumpliendo con las especificaciones que definen las distintas licencias del software utilizado.
- El programa resultante debe tener una licencia Open Source compatible con las tecnologías usadas. La licencia de Open OTC será open source con clausulas para impedir que otros hagan beneficio económico con el programa.

Estudio y valoración de alternativas tecnológicas

Según hemos visto en el apartado anterior, en los requisitos siempre buscamos las soluciones Open Source, tanto para las tecnologías usadas como para el producto final.

Como alternativas posibles encontramos los siguientes:

- Infraestructura con licencia privativa:
  - S.O. → Microsoft Windows Server
  - Web Server → ISS
  - SQL Server → Microsoft SQL Server, Sybase, Oracle Database.
- Infraestructura con licencia libre:
  - S.O. → Ubuntu Server
  - Web Server → Apache
  - SQL Server → Mysql

La aplicación resultante en cualquier caso debe ser Open Source.

Las dos opciones de infraestructura cumplen los requisitos técnicos en cuanto a carga de trabajo, posibilidad de gestión, seguridad, mantenimiento y desarrollo. Sin embargo, como hemos dicho antes, buscaremos siempre las opciones Open Source disponibles. La opción de infraestructura Open Source se presenta como la mejor opción. Contando que el sistema operativo basado en GNU/Linux son los más difundidos en lo que a servidores se refiere, no se ve la necesidad en ningún caso de optar por la solución privativa. Lo mismo ocurre con el servidor SQL y sobre todo con el servidor web Apache, que el servidor web más difundido en Internet.

Si bien es cierto que en términos legales, la elección de la infraestructura no es determinante en la elección de la licencia Open Source en la aplicación resultante.

Riesgos de la infraestructura privativa:

- Pérdida del soporte del sistema operativo escogido. Obligación de realizar una actualización.
- Dependencia del fabricante a la hora de realizar parches de seguridad. Lentitud a la

hora de solventar problemas de seguridad.

Los riesgos anteriores se podrían mitigar mediante contratos con el fabricante que nos protejan de los riesgos anteriores, suponiendo un incremento en el gasto inicial.

Riesgos de la infraestructura Open Source:

- Pérdida del interés de la comunidad en la continuidad de alguno de los proyectos utilizados.
- Falta de soporte a la hora de realizar parches o actualizaciones necesarias.

Los riesgos anteriores se podrían mitigar mediante la contratación de empresas que ofrezcan servicios de soporte. Por otro lado, las tecnologías usadas tienen una comunidad de usuarios fortísima, y una serie de empresas del sector privado interesadas en la continuidad de los diferentes proyectos, por lo que presupone difícil que se cumpla alguno de los riesgos anteriores.

Con los datos anteriores, se ha decidido optar por la solución Open Source, tanto en la infraestructura usada como en el producto final.

Planificación de las actividades

Las diferentes etapas del desarrollo de Open OTC Viewer se pueden observar en el siguiente diagrama de Gantt. Se comenzó el desarrollo en Abril de 2015, finalizando en Octubre del mismo año.

Se ha seguido el sistema tradicional de acabar una tarea para comenzar con la siguiente. Sin embargo, para la fase de pruebas, las unitarias se realizan en paralelo al desarrollo, es decir, se van realizando pruebas según se desarrolla, para finalizar con unas pruebas integradas de todos los aspectos que forman Open OTC Viewer.

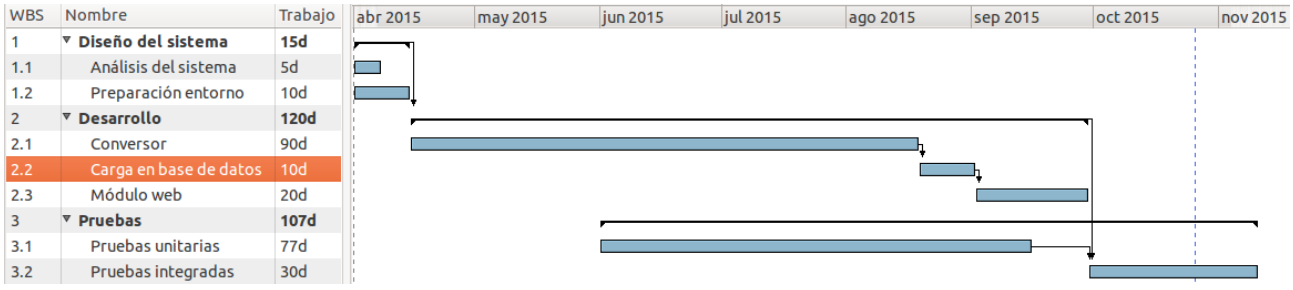


Ilustración 1: Diagrama de Gant - Fases de Open OTC Viewer

Fundamentos teóricos

En el apartado de fundamentos teóricos vamos a incluir la información relativa a los ficheros origen FpML, su contenido, su significado y su equivalente en MxML.

Toda la información se ha obtenido de la página oficial del proyecto FpML.

Definición de FpML: “Financial products Markup Langage”. Es un estándar Open Source basado en XML dedicado al procesamiento electrónico de derivados OTC. Establece un protocolo para el intercambio de información financiera.

¿Qué es un Interest Rate Swap?. Un Interest Rate Swap o permuta financiera es un contrato entre dos partes en el que se acuerda el intercambio de flujos de capital en fechas futuras en base a un subyacente. Normalmente se referencia a algún índice bursátil. El swap más común contendrá dos “patas”, una flotante y otra fija. La pata flotante se referencia con un índice bursátil, y la pata fija se presenta como un portcentaje sobre el subyacente. Los swaps son usados en los siguientes escenarios:

- Cambiar bienes o recursos futuros.
- Especulación.
- Cobertura.

FpML

A continuación mostramos las etiquetas XML que conforman una operación de tipo Interest Rate Swap en un fichero FpML:

- Definición del esquema:
  <FpML xmlns="http://www.fpml.org/FpML-5/confirmation"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  fpmlVersion="5-3"
  xsi:type="DataDocument">
- En FpML, el nodo que nos indica el tipo de producto que representa viene definido en los siguientes campos:
  <primaryAssetClass assetClassScheme="http://www.fpml.org/coding-scheme/asset-class-simple">**InterestRate**</primaryAssetClass>
  <productType productTypeScheme="http://www.fpml.org/coding-scheme/product-taxonomy">**InterestRate:IRSwap:FixedFloat**</productType>
- Un swap, como hemos dicho, es un contrato entre dos partes, es decir, existe una parte A y una parte B. Estos valores se expresan de la siguiente manera:
  <payerPartyReference href="party1"/>
  <receiverPartyReference href="party2"/>
  Una vez transformados, estos valores serán el portfolio y la contrapartida.
- Como hemos dicho antes, el caso más habitual es tener una pata fija y otra flotante. Las etiquetas FpML para especificar la flotante y la fija son las siguientes:
  - <swapStream id="fixedLeg1"> → Pata fija.

- <swapStream id="floatingLeg2"> → Pata flotante.

- Para la pata fija, los siguientes nodos XML especifican toda la información relativa a la pata fija del swap, todos ellos dentro de la etiqueta <swapStream id="fixedLeg1">.

  - **effectiveDate**: Define la fecha en la que la pata fija del swap se hace efectiva (la pata fija y flotante pueden comenzar en diferentes fechas). Se puede dar el caso que la fecha sea en festivo (en festivos los mercados OTC están cerrados, es decir, no hay negociación, por lo que ninguna fecha de pago/cobro/inicio/fin podrá ser festivo). Es por este motivo que se añade la etiqueta **unadjustedDate,** la cual indica la fecha que se debe ajustar en caso de ser necesario**.** Las etiquetas **dateAdjustments** y **businessDayConvention** se utilizan para especificar que hacer con la fecha en caso de festivo.

    NOTA: El valor de los nodos son códigos usados por los usuarios, no son importantes de cara a la transformación, simplemente tenemos que conocer que nodo FpML corresponde a que nodo MxML, por lo que el contenido no se va a explicar en este proyecto. Estos códigos los entienden las aplicaciones de tesorería, por lo que al introducir un fichero con estos códigos, los aplicativos cargan la operación aplicando las modificaciones indicadas en los nodos FpML o MxML.

    Por último, la etiqueta **adjustedDate** indica la fecha definitiva de inicio una vez aplicado el ajuste NONE (en este caso NONE significa que no hay que hacer nada).

```
<effectiveDate>
        <unadjustedDate>2013-09-10</unadjustedDate>
        <dateAdjustments>
                <businessDayConvention>NONE</businessDayConvention>
        </dateAdjustments>
        <adjustedDate>2013-09-10</adjustedDate>
</effectiveDate>
```

  - **terminationDate**: Define la fecha en la que la pata fija finalizada, es decir, el día que acaba el contrato para la pata fija. Las etiquetas son las mismas que para el **effectiveDate,** pero con una diferencia, en este caso el valor de **businessDayConvention** es MODFOLLOWING, este código indica que la fecha se debe mover en caso de caer en festivo. Cuando una fecha se debe mover se aplica el nodo **businessCenter** que indica el calendario que se debe tener en cuenta a la hora de mover la fecha.

```
<terminationDate>
        <unadjustedDate>2015-09-10</unadjustedDate>
        <dateAdjustments>
        <businessDayConvention>MODFOLLOWING</businessDayConvention>
        <businessCenters>
                <businessCenter>CATO</businessCenter>
        </businessCenters>
        </dateAdjustments>
```

```
        <adjustedDate>2015-09-10</adjustedDate>
</terminationDate>
```

- **calculationPeriodDatesAdjustments**: Si alguno de los nodos anteriores caen en día no laborable, se aplica el modificador **businessDayConvention** sobre el calendario **businessCenter.**

```
<calculationPeriodDatesAdjustments>
        <businessDayConvention>MODFOLLOWING</businessDayConvention>
        <businessCenters>
                <businessCenter>CATO</businessCenter>
        </businessCenters>
</calculationPeriodDatesAdjustments>
```

- **calculationPeriodFrequency**: Nos indica la frecuencia de cobros/pagos de la pata fija. La etiqueta **periodMultiplier** es un multiplicador, que unido a la etiqueta **period** nos indica cada cuanto tiempo se cobra/paga. En el ejemplo, cada 6 meses. Por último, la etiqueta **rollConvention** indica el día exacto en el que se cobra/paga, en este caso es 10, eso quiere decir que cada 6 meses, en el décimo día se cobrará/pagará.

```
<calculationPeriodFrequency>
        <periodMultiplier>6</periodMultiplier>
        <period>M</period>
        <rollConvention>10</rollConvention>
</calculationPeriodFrequency>
```

- El siguiente nodo especifica las fechas de pago **<paymentDates id="paymentDates1">**, es usado junto al nodo anterior que especifica como calcular las fechas de pago/cobro, la siguiente etiqueta referencia a dicho nodo **<calculationPeriodDatesReference href="fixedCalcPeriodDates1"/>.**

```
<paymentDates id="paymentDates1">
        <calculationPeriodDatesReference href="fixedCalcPeriodDates1"/>
        <paymentFrequency>
                <periodMultiplier>6</periodMultiplier>
                <period>M</period>
        </paymentFrequency>
        <paymentDatesAdjustments>
        <businessDayConvention>MODFOLLOWING</businessDayConvention>
        <businessCenters>
                <businessCenter>CATO</businessCenter>
        </businessCenters>
        </paymentDatesAdjustments>
</paymentDates>
```

- **calculationPeriodAmount:** Este nodo especifica los parámetros del subyacente.

Como etiquetas importantes encontramos **initialValue**, especifica la cantidad, la divisa de la cantidad anterior se detalla en el nodo **currency.** El nodo **fixedRateSchedule**, solo presente en la pata fija, especifica el porcentaje del subyacente que se paga/cobra cuando se cumplen los periodos especificados arriba. Por último, la etiqueta **dayCountFraction** indica la fracción de conteo de días.

```
<calculationPeriodAmount>
        <calculation>
                <notionalSchedule>
                        <notionalStepSchedule>
                                <initialValue>100000000</initialValue>
                                <currency>CAD</currency>
                        </notionalStepSchedule>
                </notionalSchedule>
                <fixedRateSchedule>
                        <initialValue>0.016409</initialValue>
                </fixedRateSchedule>
                <dayCountFraction>ACT/365.FIXED</dayCountFraction>
        </calculation>
</calculationPeriodAmount>
```

La agrupación de todos los nodos anteriores definen toda la información necesaria de la que está formada la pata fija.

- Pasamos a definir la otra parte del swap, la pata flotante. Toda ella dentro de nodo <swapStream id="floatingLeg2">

  - Los nodos effectiveDate, terminationDate, calculationPeriodDatesAdjustments, calculationPeriodFrequency y paymentDates son similares a los de la pata fija, especifican la misma información de la pata flotante.
  - Sin embargo, en la pata flotante, se especifica un nuevo nodo <resetDates id="resetDates2">. Esta etiqueta se utiliza en las patas flotantes para indicar el día en el que se toma el valor del índice de la pata flotante. Por ejemplo, si tenemos una pata referenciada al índice EURIBOR 3M, este valor cambia cada día, es por esto que la pata flotante debe especificar el día concreto en el que se tomará el valor del índice para realizar los pagos/cobros correspondientes. Las fechas en las que se toma el valor del índice se llaman fijaciones. Las etiquetas hijas de resetDates se han explicado en los puntos anteriores.
  - En la etiqueta calculationPeriodAmount se dan diferencias entre la pata fija y la flotante. En la flotante, el nodo que especificaba el porcentaje del subyacente a cobrar/pagar es sustituido por:

    ```
    <floatingRateCalculation>
        <floatingRateIndex>CAD-BA-CDOR</floatingRateIndex>
        <indexTenor>
                <periodMultiplier>3</periodMultiplier>
                <period>M</period>
        </indexTenor>
    </floatingRateCalculation>
    ```

Como hemos dicho, la pata flotante suele estar referenciada a un índice, en este caso está expresado en el nodo floatingRateIndex. Este tipo de índices tienen asociado un periodo, que en este caso viene informado en los nodos periodMultiplier y period.
Por último, encontramos el nodo compoundingMethod (puede estar informado en cualquiera de las patas), este valor se utiliza para indicar que los beneficios se acumulan, se trata de una variación de los contratos swap utilizados por los inversores.
También podemos encontrar en algunos casos que para la pata flotante se especifica un valor dentro del nodo <spreadSchedule> y su subnodo <initialValue>. El valor contenido en dicho nodo se utiliza para calcular (suma o resta) el valor de los cobros/pagos en relación con el valor del índice.

MxML

A continuación mostramos el contenido de los ficheros MxML una vez traducidos los ficheros FpML por la herramienta de transformación de Open OTC Viewer. La estructura de etiquetas es diferente a los FpML pero contienen la misma información. Los ficheros resultantes MxML podrían ser cargados directamente mediante el módulo de importación de operaciones del aplicativo Murex.

- La etiqueta <MxML> es el nodo padre de los ficheros.
- En MxML tenemos la etiqueta <events> que indica el tipo de acción que se va a realizar al insertar el fichero.

```
<events>
	<mainEvent>
		<action>insertion</action>
		<object href="#trade">
			<objectNature>trade</objectNature>
		</object>
		<inputBy>
			<userName>REALTIME</userName>
			<group>MXML</group>
		</inputBy>
	</mainEvent>
</events>
```

Se define el tipo de acción (<action>insertion</action>), que en este caso indica que el fichero MxML contiene una operación (<objectNature>trade</objectNature>) para insertar. Por último, se definen el usuario (<userName>REALTIME</userName>) y el grupo (<group>MXML</group>) que ha insertado la operación. Estos últimos valores de usuario y grupo se usan en Murex para organizar e identificar los diferentes grupos y usuarios que han contratado la operativa.

- El siguiente nodo <trades> engloba toda la información de la operación.
- Dentro de <trades> encontramos el nodo <parties> que se utiliza para identificar las partes de una operación. En este apartado solamente se definen una parte A y otra B, una será la que paga y otra la que cobra, se referencian en otro nodo posterior del

fichero MxML.

```
<parties>
        <party id="party1">
                <partyName>BBVA</partyName>
                <partyFullName>BBVA</partyFullName>
        </party>
        <party id="party2">
                <partyName>CONTRAPARTIDA</partyName>
        </party>
</parties>
```

- El siguiente nodo es utilizado para indicar el portfolio. Los portfolios son utilizados en Murex como sistema de organización. En este nodo únicamente se referencia el portfolio y se le da un valor.

```
<portfolios>
        <portfolio id="portfolio">
                <portfolioLabel>Portfolio Prueba</portfolioLabel>
        </portfolio>
</portfolios>
```

- En Murex se define una operación con su cabecera y su cuerpo. La cabecera es común a las dos patas (fija y flotante), su información se encuentra dentro del nodo <tradeHeader>. A continuación mostramos los nodos hijos:

  - <tradeDate>20130910</tradeDate>: Identifica la fecha de contratación.
  - <tradeCategory>: Este nodo es usado para definir el tipo de operación. Como hemos dicho antes, trabajamos con operaciones de tipo swap. En Murex, se definen los diferentes tipos de productos por familia, grupo y tipo. En nuestro caso, un swap tiene familia “IRD”, grupo “IRS”, y tipo vacío.
    También encontramos el nodo <tradeDestination>. Murex permite la contratación de operativa “externa” e “interna”. Es decir, si la contrapartida es una entidad externa, o en caso contrario, la contrapartida pertenece a la misma entidad.

```
<tradeCategory>
        <tradeDestination>external</tradeDestination>
        <tradeFamily>IRD</tradeFamily>
        <tradeGroup>IRS</tradeGroup>
</tradeCategory>
```

  - El siguiente nodo es <tradeViews>. En el se continúan definiendo valores de la cabecera de la operación. Está dividido en dos nodos, que identifican las dos partes de la operación. Cada una de las partes se definen dentro de los nodos <tradeView>:

```
<tradeViews>
        <tradeView>
                <partyReference href="#party1">
                <portfolioReference href="#portfolio"/>
```

```
                    </partyReference>
                        <userName>REALTIME</userName>
                        <entity>EntidadPortfolio</entity>
                    <tradeId>
                        <tradeGlobalId>LCH00005835481</tradeGlobalId>
                    </tradeId>
                    <comments>
                        <comment index="0">LCH00005835481</comment>
                    </comments>
                </tradeView>
                <tradeView>
                    <partyReference href="#party2"/>
                </tradeView>
            </tradeViews>
```

En este ejemplo, la primera parte de la operación se referencia a la party1 definida anteriormente mediante el nodo <partyReference href="#party1">, y se asocia al valor #portfolio definido en un nodo anterior. Es decir, en este ejemplo, la parte1 será el porfolio.
El nodo <entity> se utiliza en Murex como otro nivel de organización. Los porfolios en Murex forman parte de una entidad. Open OTC Viewer asocia portfolios con su entidad mediante una tabla en base de datos.
<tradeId> se utiliza como identificador de la operación.
<comments> es utilizado por los brokers y traders para escribir comentarios que afectan a la operación. Murex permite hasta 4 comentarios, por este motivo se especifica el nodo <comment index="0"> con un índice, que podrá tomar los valores [0-3].
Por último, se referencia la parte2 a la contrapartida mediante <partyReference href="#party2"/>.

- Una vez definida la cabecera de la operación, pasamos al cuerpo del swap. El cuerpo se detalla en el nodo <tradeBody> y su nodo hijo <interestRateSwap>, en el se especifica toda la información sobre la pata fija y la pata flotante.

  - El nodo <templateLabel> especifica el "generador" de la pata flotante. Los generadores se utilizan para que el aplicativo calcule (según unos valores definidos dentro del generador) las fijaciones y los diferentes flujos, su cantidad a pagar/cobrar y sus fechas.
  - En MxML, las dos patas de la operación se definen dentro de la etiqueta <stream>, como nodos hijos encontramos (son comunes tanto a la pata fija como a la flotante):

    - <effectiveDate>: Día que se hace efectiva la operación. No tiene por que ser la misma que encontrábamos en el nodo <tradeDate>.
    - <maturity>: Día en el que la operación vence.
    - <capital>: Subyacente de la operación. Se define en el subnodo <initialCapitalAmount>.
    - El siguiente nodo especifica la información de la pata 1 y 2 respectivamente.
      - <phase> y <leg>: Definen la fase y la pata. En Murex, una operación

puede tener varias fases, en cada una de ellas se definen dos patas. Lo habitual es tener una única fase, pero se podrían definir más.

- El siguiente nodo <streamSchedules> se utilizada para que Murex sea capaz de calcular las fechas de pagos/cobros. Los nodos donde se encuentran los valores a aplicar los encontramos en <scheduleGeneratorLabel> y <dateShifterLabel>.
- El nodo <paymentBusinessCenters> indica el calendario a aplicar a la hora de calcular fechas.
- En MxML, para cada una de las patas, se especifica la parte que paga mediante el nodo <payerPartyReference href="#party2"/> y la que recibe <receiverPartyReference href="#party1"/>. En el nodo que expresa la siguiente pata las partes que pagan y reciben son en dirección opuesta.
- En la pata fija encontraremos el nodo <fixedRateStream> y el subnodo <fixedRate> que especifica el porcentaje del subyacente a cobrar/pagar en la pata fija. Sin embargo, en la pata flotante encontramos el nodo <spreadSchedule> y el subnodo <initialValue> que indican el porcentaje a sumar/restar al valor del índice en las fechas de cobros/pagos.

Análisis del sistema

En el siguiente apartado pasamos a definir las especificaciones detalladas de Open OTC Viewer.

Definición del sistema

Los objetivos que Open OTC Viewer deberá cumplir son los siguientes:

- Transformador FpML a MxML:
  - El transformador tiene como origen ficheros FpML generados desde un aplicativo de gestión de tesorería como puede ser Calypso o desde otro tipo de organizaciones como puede ser la London Clearing House[xi] o similares. Estos ficheros contienen toda la información necesaria para definir una operación de tipo IRD IRS (Swap).
  - Las transformaciones se realizan en base a un fichero de transformaciones dtd[xii] y a unas plantillas que definen los orígenes FpML y sus correspondientes destinos MxML.
  - El resultado son ficheros con formato MxML. Estos ficheros podrían ser cargados en el aplicativo Murex a través de sus propios módulos de importación. No aplica en este proyecto la inserción de operativa en el aplicativo Murex ya que se trata de una aplicación de código propietario.
- Carga en la base de datos:
  - Con el fin de que los usuarios puedan visualizar su operativa desde cualquier dispositivo con conexión a Internet, los ficheros MxML transformados son recorridos por Open OTC Viewer y cargados en una base de datos de operaciones.
  - La base de datos y sus correspondientes tablas deben estar creadas bajo unas definiciones preestablecidas de longitudes y nombres de campos.
  - Open OTC Viewer trabaja con bases de datos MySQL. El administrador del sistema será el encargado de configurar la base de datos.
- Visualización web:
  - Open OTC Viewer permite la visualización de la operativa de cada trader vía web. Para ello se debe disponer de un servidor Web accesible desde el exterior.
  - El administrador del sistema será el encargado de la gestión del servidor web.

- Open OTC Viewer será accesible mediante usuario y contraseña. Será el administrador del sistema el encargado de dar de alta los diferentes usuarios en la tabla de usuarios de la base de datos.

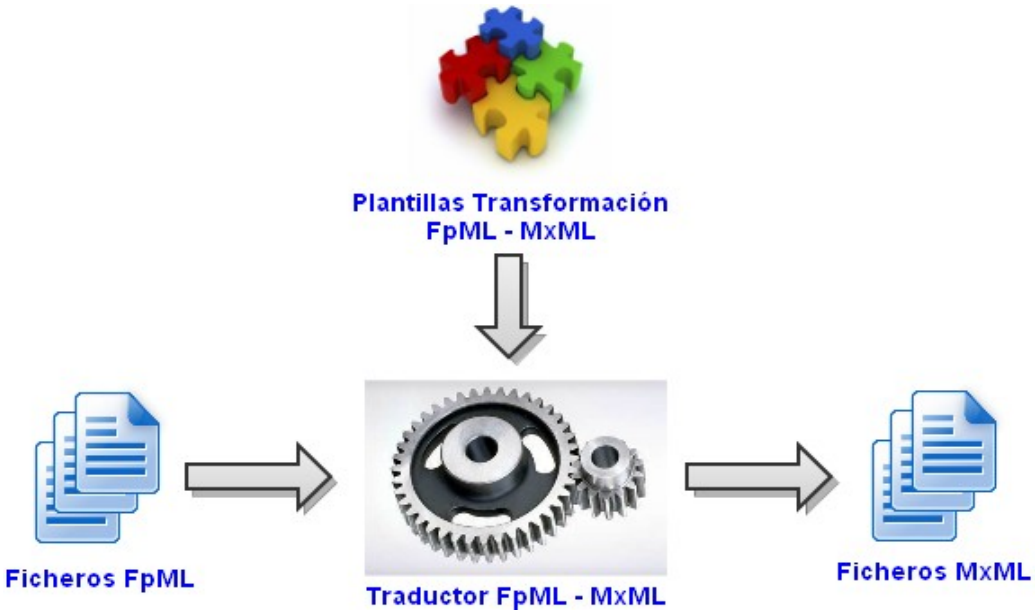
- El servidor web debe ser accesible desde el exterior. Será necesario la configuración de sistemas de seguridad.

- En caso de que algún parámetro de alguna operación se modifique, se deberá contactar con el administrador de la base de datos para que realice los cambios especificados por el trader.

Infraestructura y programación

Como hemos comentado en puntos anteriores, Open OTC Viewer se ha programado con lenguaje Java para las transformaciones de ficheros y Java EE y JSP[xiii] para la parte web.

Las transformaciones la realiza la pieza Java en base a unas plantillas (ficheros xml) que definen la estructura de los ficheros MxML. Estas plantillas se apoyan en un fichero dtd que explicaremos más adelante para realizar las transformaciones y validaciones. Todo el conjunto de programación Java, sumado a las plantillas y al código dtd forman el módulo de transformación.



La parte web, ha sido programada con lenguaje Java EE. Java EE tiene un API muy difundido que permite realizar diferentes funciones enfocadas a la programación web, como puede ser el uso de servlets, JavaBeans, JSP, etc. Open OTC Viewer consta de varias páginas web programadas en JSP. JSP nos proporciona un lenguaje que puede ser leído por servidores web y permiten la creación de webs dinámicas.

Open OTC Viewer utiliza servlets. Son los encargados de redirigir la ejecución a los diferentes módulos de los que consta Open OTC Viewer. Esto lo consigue por medio de las peticiones GET y POST que realiza el protocolo HTTP.

La base de datos se ha creado con MySQL. Java permite la conexión a la base de datos usando conectores. En nuestro caso se ha usado el conector “mysql-connector-java-5.1.6-bin.jar”. Este conector permite a Java realizar conexiones a la base de datos externa, pudiendo realizar peticiones a la base de datos (select, update, drop, truncate, etc).

El código web (JSP) debe ser interpretado por un servidor web. Para la programación del aplicativo se ha usado el servidor GlassFish, el cual es fácilmente integrado con el IDE Ecplipse. Sin embargo, se recomienda el uso del servidor Apache2[xiv] para la puesta en producción, ya que Apache2 es el servidor web por excelencia, por su robustez, fiabilidad y gracias a una comunidad de usuarios muy involucrada con el proyecto.

Por último, el servidor web y la base de datos se han ubicado en un servidor GNU/Linux. Para el desarrollo se ha utilizado Ubuntu 14.04.

El conjunto de las tecnologías arriba descritas y la programación Java forman el conjunto de Open OTC Viewer.

Otro punto importante del proyecto Open OTC Viewer es su filosofía Open Source. El objetivo será crear una comunidad que permita el crecimiento de la herramienta. Un punto clave en este apartado es la publicación del código fuente. Como hemos dicho anteriormente se ha usado el software EGIT. Está basado en GIT, se trata de un software de control de versiones que ha sido desarrollado por el equipo de Eclipse. Permite la inclusión de GIT en el propio IDE Ecplise. Al igual que el servidor web, se podría haber usado un software externo, pero Eclipse proporciona un entorno donde GlassFish y EGIT se pueden integrar en Ecplise y así facilitar el desarrollo.

Diseño del sistema

En este apartado pasamos a definir en detalle Open OTC Viewer. Nos centraremos en el módulo de transformación, ya que es el más extenso de Open OTC Viewer.

Módulo de transformación

El módulo de transformación es la parte de Open OTC Viewer más extensa. En este apartado vamos a definir detalladamente la programación de este módulo.

La traducción se realizará siguiendo una plantilla de documento MxML, la cual se completará con los datos de la operación FpML siguiendo unas reglas de transformación preestablecidas para cada tipo de producto. De esta manera no será necesario realizar cambios en el código del componente cuando se modifique la traducción de un tipo de producto o la creación de la traducción de un tipo de producto adicional.

Los ficheros MxML, aún siendo del mismo tipo de producto, pueden tener una estructura diferente dependiendo de los datos de la operación.

El proceso de traducción construirá el fichero MxML utilizando distintas plantillas que representan partes concretas del MxML. Dichas plantillas se añadirán o no al fichero MxML según los datos de la operación. La creación del fichero MxML es una construcción mediante piezas.

MxML Template

Las plantillas MxML contienen la definición de una operación en formato MxML con campos vacíos y con campos fijos dependiendo del tipo de operación.

Se deben encontrar en el directorio de plantillas determinado en la configuración, y deben seguir la siguiente nomenclatura:

[nombre_plantilla]_template.xml

Estas plantillas MxML siguen la definición de tipo de documento del formato MxML:

mxml.dtd

MxML Transform

Los ficheros de transformación MxML definen la forma en la que se completan las plantillas MxML con los datos de la operación.

Se deben encontrar en el directorio de plantillas determinado en la configuración, y deben seguir la siguiente nomenclatura:

[nombre_plantilla]_transform.xml

Los ficheros de transformación de FpML a MxML siguen la definición de tipo de documento:

transform-fpml-mxml.dtd

A partir del nodo transform-fpml-mxml.

Transformación

La transformación de una operación se inicia utilizando la plantilla y la transformación correspondientes al tipo de operación.

Para poder transformar una operación debe existir como mínimo una plantilla y una transformación con el nombre del tipo de operación. Por ejemplo:

Tipo de Operación:	IRS
Plantilla:	IRS_template.xml
Transformación:	IRS_transform.xml

Definición de una transformación

Los ficheros de transformación de FpML a MxML siguen la definición de tipo de documento:

transform-fpml-mxml.dtd

A continuación mostramos el contenido del fichero dtd:

```
<!ELEMENT transform-fpml-mxml ( transformacion* ) >
<!ELEMENT transformacion ( condicion? , origen , destino+ ) >
 <!ELEMENT condicion ( operador | existe | not | and | or ) >
 <!ELEMENT not ( operador | existe | not | and | or ) >
 <!ELEMENT and ( operador | existe | not | and | or )* >
 <!ELEMENT or ( operador | existe | not | and | or )* >
 <!ELEMENT operador ( origen+ ) >
<!ATTLIST operador id ( igual | mayor | menor ) #REQUIRED >
 <!ELEMENT existe ( origen ) >
 <!ELEMENT origen EMPTY >
<!ATTLIST origen tipo ( propiedad | xpath | nodo | variable | funcion | template | constante )
#REQUIRED >
<!ATTLIST origen id CDATA #IMPLIED >
<!ATTLIST origen cardinalFpml CDATA #IMPLIED >
```

```
<!ATTLIST origen valorDefecto CDATA #IMPLIED >
<!ATTLIST origen tratamiento NMTOKEN #IMPLIED >
<!ATTLIST origen recursivoPor CDATA #IMPLIED >
 <!ELEMENT destino EMPTY >
<!ATTLIST destino tipo ( variable | xpath | eliminar | xpathlist ) #REQUIRED >
<!ATTLIST destino id CDATA #REQUIRED >
<!ATTLIST destino tratamiento NMTOKEN #IMPLIED >
 <!ELEMENT productos ( producto+ ) >
 <!ELEMENT producto ( condicion, intercambiarPatas? ) >
<!ATTLIST producto id CDATA #REQUIRED >
 <!ELEMENT intercambiarPatas ( condicion, origen, origen ) >
 <!ELEMENT caracteristicas ( caracteristica+ ) >
<!ATTLIST caracteristicas id CDATA #REQUIRED >
 <!ELEMENT caracteristica ( condicion? , origen+ ) >
<!ATTLIST caracteristica id CDATA #REQUIRED >
<!ATTLIST caracteristica comun ( true | false ) #IMPLIED >
```

A continuación se explica el contenido del fichero dtd y sus implicaciones.

transform-fpml-mxml.dtd

transform-fpml-mxml es el nodo principal del documento de transformación.

Estructura

- <transform-fpml-mxml>
  Subnodos:
  - <transformacion>

Subnodos:

Subnodo	Descripción	Cardinalidad
`<transformacion>`	Nodo que define la transformación de un dato	[1 .. N]

El nodo <transformacion> define la transformación de un dato. Indica el origen del que se obtendrá el valor a transformar y los distintos destinos a los que se asignará el valor obtenido. Puede definir además una condición que habilitará o no la transformación.

Estructura

- <transformacion>
  Subnodos:
  - <condicion>
  - <origen>
  - <destino>

Subnodos:

Subnodo	Descripción	Cardinalidad
<condicion>	Nodo que define la condición que habilita la transformación.	[0 .. 1]
<origen>	Nodo que define el origen del dato que se transforma.	[1 .. 1]
<destino>	Nodo que define el destino al que se asignará el dato obtenido.	[1 .. N]

El nodo <condicion> define la condición que habilita la transformación según el resultado de la evaluación de sus operadores condicionales. Debe contener solo uno de sus posibles subnodos.

Estructura

- <condicion>
  Subnodos:
  - <operador>
  - <existe>
  - <not>
  - <and>
  - <or>

Subnodos:

Subnodo	Descripción	Cardinalidad
<operador>	Nodo que define una comparación de dos valores según un operador (igual, mayor, menor)	[0 .. 1]
<existe>	Nodo que define la comprobación de la existencia de un valor.	[0 .. 1]
<not>	Nodo que define el operador condicional de negación.	[0 .. 1]
<and>	Nodo que define el operador condicional 'Y'.	[0 .. 1]
<or>	Nodo que define el operador condicional 'O'.	[0 .. 1]

El nodo <operador> define un operador condicional que realiza la comparación de dos valores.

Atributos:

Atributo	Descripción	Obligatorio
id	Indica el identificador del operador de comparación que se va a aplicar a los datos.	X

Estructura:

- <operador>
  Atributos:
  - id
    Subnodos:
    <origen>

Subnodos:

Subnodo	Descripción	Cardinalidad
`<origen>`	Nodo que define el origen del que se obtendrá el valor para la comparación.	[1 .. 2]

El nodo <existe> define un operador condicional que determina si el origen indicado tiene valor.

Estructura:

- <existe>
  Subnodos:
- <origen>

Subnodos:

Subnodo	Descripción	Cardinalidad
`<origen>`	Nodo que define el origen del que se obtendrá el valor a comprobar.	[1 .. 1]

El nodo <not> define un operador condicional que realiza la negación de la condición que contiene. Debe contener solo uno de sus posibles subnodos.

Estructura:

- <not>
  Subnodos:
  - <operador>
  - <existe>
  - <not>
  - <and>
  - <or>

Subnodos:

Subnodo	Descripción	Cardinalidad
<operador>	Nodo que define la comparación de dos valores.	[0 .. 1]
<existe>	Nodo que define la comprobación de la existencia de un valor.	[0 .. 1]
<not>	Nodo que define el operador condicional de negación.	[0 .. 1]
<and>	Nodo que define el operador condicional 'Y'.	[0 .. 1]
<or>	Nodo que define el operador condicional 'O'.	[0 .. 1]

El nodo <and> define un operador condicional que realiza la conjunción (condición 'Y') de las condiciones que contiene. Debe contener al menos uno de sus posibles subnodos.

Estructura:

- <and>
  Subnodos:
  - <operador>
  - <existe>
  - <not>
  - <and>
  - <or>

Subnodos:

Subnodo	Descripción	Cardinalidad
<operador>	Nodo que define la comparación de dos valores.	[0 .. N]
<existe>	Nodo que define la comprobación de la existencia de un valor.	[0 .. N]
<not>	Nodo que define el operador condicional de negación.	[0 .. N]
<and>	Nodo que define el operador condicional 'Y'.	[0 .. N]
<or>	Nodo que define el operador condicional 'O'.	[0 .. N]

El nodo <or> define un operador condicional que realiza la disyunción (condición 'O') de las condiciones que contiene. Debe contener al menos uno de sus posibles subnodos.

Estructura:

- <or>
  Subnodos:
  - <operador>

- <existe>
- <not>
- <and>
- <or>

Subnodos:

Subnodo	Descripción	Cardinalidad
`<operadot>`	Nodo que define la comparación de dos valores.	[0 .. N]
`<existe>`	Nodo que define la comprobación de la existencia de un valor.	[0 .. N]
`<not>`	Nodo que define el operador condicional de negación.	[0 .. N]
`<and>`	Nodo que define el operador condicional 'Y'.	[0 .. N]
`<or>`	Nodo que define el operador condicional 'O'.	[0 .. N]

El nodo <origen> define el origen del que se obtendrá un dato para la transformación.

Estructura:

- <origen>
  Atributos:
  - tipo
  - id
  - cardinalidadFpML
  - valorDefecto
  - tratamiento
  - recursivoPor

Atributos:

Atributo	Descripción	Obligatorio
`tipo`	Es el tipo de origen de dato. Determina como se obtiene.	X
`id`	Indica el identificador del dato, según el tipo de origen. Es obligatorio salvo con tipo "fpml".	X
`cardinalidadFpML`	Es opcional y solo se usa con tipo de origen "template". Indica que para la transformación de la plantilla se usará la cardinalidad indicada en todos sus orígenes de datos.	
`valorDefecto`	Indica el valor en caso de que el tipo de origen sea "constante"	
`tratamiento`	Indica que se aplicará una función al dato antes de utilizarlo.	
`recursivoPor`	Indica que se utilizará esa transformación recursivamente a todos los nodos que correspondan con la expresión XPath indicada.	

Tipos de origen

- propiedad: Obtiene el dato del fichero de propiedades.
- xpath: Obtiene el texto de un nodo FpML indicado por una expresión XPath.
- nodo: Obtiene un nodo FpML indicado por una expresión XPath.
- funcion: Obtiene el dato de la ejecución de una función.
- variable: Obtiene el dato de una variable del proceso.
- template: El dato será una plantilla MxML transformada. Puede ir acompañado del atributo cardinalidadFpML que indica la cardinalidad que se usará en las expresiones XPath de la transformación de la plantilla.
- constante: El dato es un valor literal indicado en el atributo valorDefecto.

El nodo <destino> define el destino al que se asignará un dato en la transformación.

Estructura:

- <destino>
  Atributos:
  - tipo
  - id
  - tratamiento

Atributos:

Atributo	Descripción	Obligatorio
`tipo`	Es el tipo de destino de dato.	X
`id`	Indica el identificador del dato, según el tipo de origen.	X
`tratamiento`	Indica que se aplicará una función al dato antes de asignarlo al destino.	

Tipos de destino:

- xpath: El dato se asignará a un nodo MxML indicado por una expresión XPath.
- variable: El dato se asignará a una variable del proceso.

Propiedad

Las plantillas de traducción de operaciones pueden tener reglas de transformación que asignen a campos de la plantilla datos obtenidos de propiedades de la herramienta.

En la regla de transformación aparece como origen de tipo propiedad y el valor del atributo id será el nombre de la propiedad de la que se obtendrá el valor.

Ejemplo:

<!-- userName -->

```
<transformacion>
        <origen tipo="propiedad" id="userName"/>
        <destino tipo="xpath”  id="/trades/trade/tradeHeader/tradeViews/tradeView[1]/userName"/>
</transformacion>
```

XPath

Las plantillas de traducción de operaciones pueden tener reglas de transformación que asignen a campos de la plantilla datos obtenidos de un nodo del fichero xml de la operación traducida.

En la regla de transformación aparece como origen de tipo xpath y el valor del atributo id será la expresión XPath del nodo del que se obtendrá el valor.

Las expresiones XPath también pueden identificar un destino en una regla de transformación.

Ejemplo:

```
<transformacion>
        <origen tipo="xpath"
id="/FpML/trade/swap/swapStream/calculationPeriodDates/terminationDate/dateAdjustments/busin
essCenters/businessCenter"/>
        <destino tipo="xpath"
id="/floatingRateStreamTemplate/resetBusinessCenters/businessCenterLabel"/>
</transformacion>
```

Variable

Las plantillas de traducción de operaciones pueden tener reglas de transformación que asignen a campos de la plantilla datos obtenidos de variables mantenidas por el proceso de traducción.

En la regla de transformación aparece como origen de tipo variable y el valor del atributo id será el nombre de la variable de la que se obtendrá el valor.

Las variables también pueden ser destino en una regla de transformación para asignar o modificar su valor.

Ejemplo:

```
<!-- portfolio, portfolioReference, entity -->
<transformacion>
```

```
        <origen tipo="variable" id="portfolio"/>
        <destino tipo="xpath" id="/trades/trade/portfolios/portfolio/portfolioLabel"/>
        <destino tipo="xpath" id="/trades/trade/tradeHeader/tradeViews/tradeView[1]/entity"
tratamiento="entityPortfolio"/>
</transformacion>
```

Las variables se mantienen durante todo el proceso de traducción.

Funciones de obtención de datos

Las plantillas de traducción de operaciones pueden tener reglas de transformación que asignen a campos de la plantilla datos obtenidos de la ejecución de una función.

En la regla de transformación aparece como origen de tipo funcion y el valor del atributo id será el nombre de la función a ejecutar.

Las funciones de obtención de datos no tienen parámetros de entrada y devuelven un objeto de la clase String.

Ejemplo:

```
<!-- systemDate -->
<transformacion>
        <origen tipo="funcion" id="getSystemDate"/>
        <destino tipo="xpath" id="/trades/trade/tradeInputConditions/systemDate"/>
</transformacion>
```

Funciones de tratamiento de datos

Las plantillas de traducción de operaciones pueden tener reglas de transformación que traten los datos que se asignan a los campos de la plantilla..

En la regla de transformación aparece el nombre de la función en el atributo tratamiento de un origen o destino de la transformación.

Las funciones de tratamiento de datos tienen un parámetro de entrada de tipo Object y devuelven un objeto de la clase String.

Ejemplo:

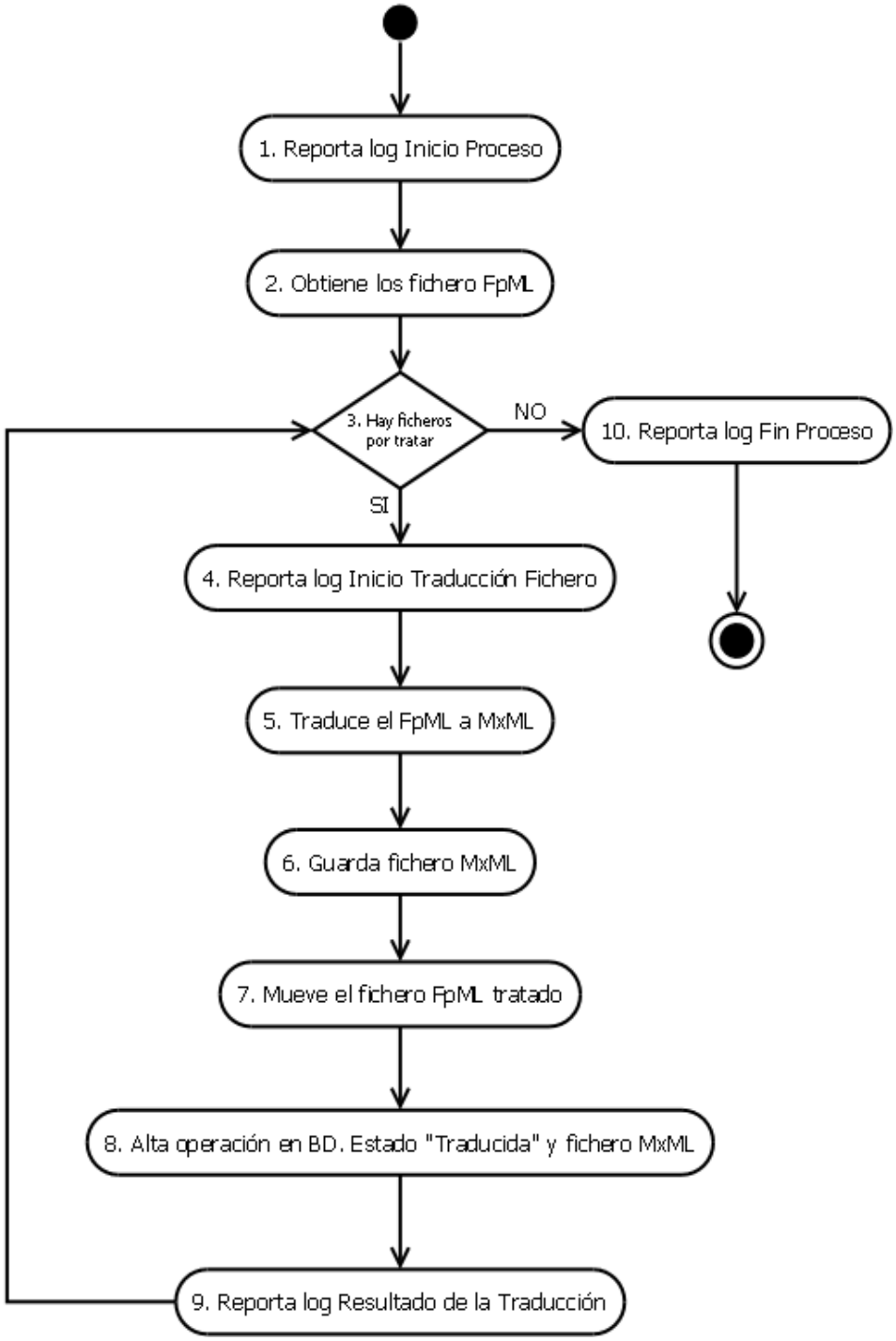
```
<!-- effectiveDate -->
<transformacion>
        <origen tipo="xpath"
id="/FpML/trade/swap/swapStream/calculationPeriodDates/effectiveDate/unadjustedDate"/>
        <destino tipo="xpath" id="/stream/effectiveDate" tratamiento="formatoFechaMxML"/>
```

</transformacion>

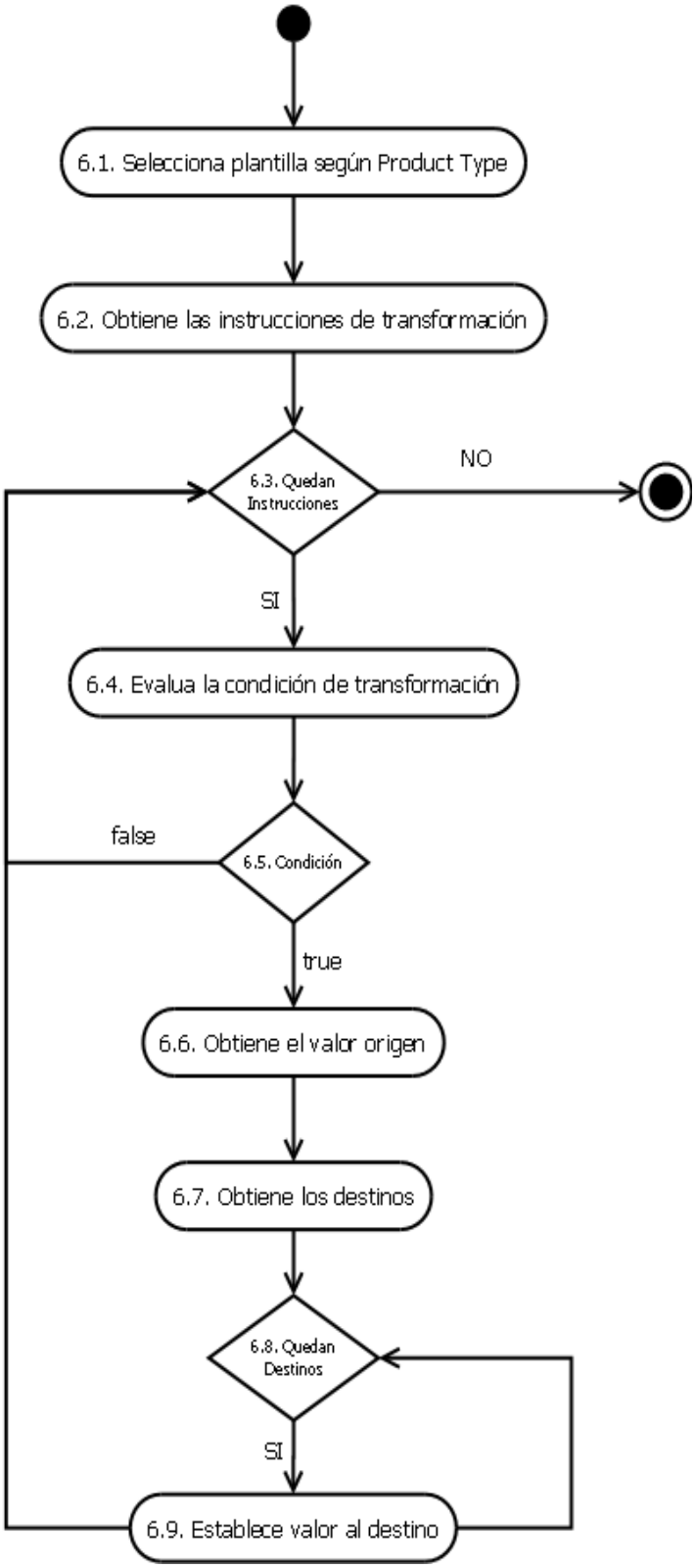
Diagrama de actividad

A continuación mostramos el diagrama de actividad de los diferentes aspectos del transformador.

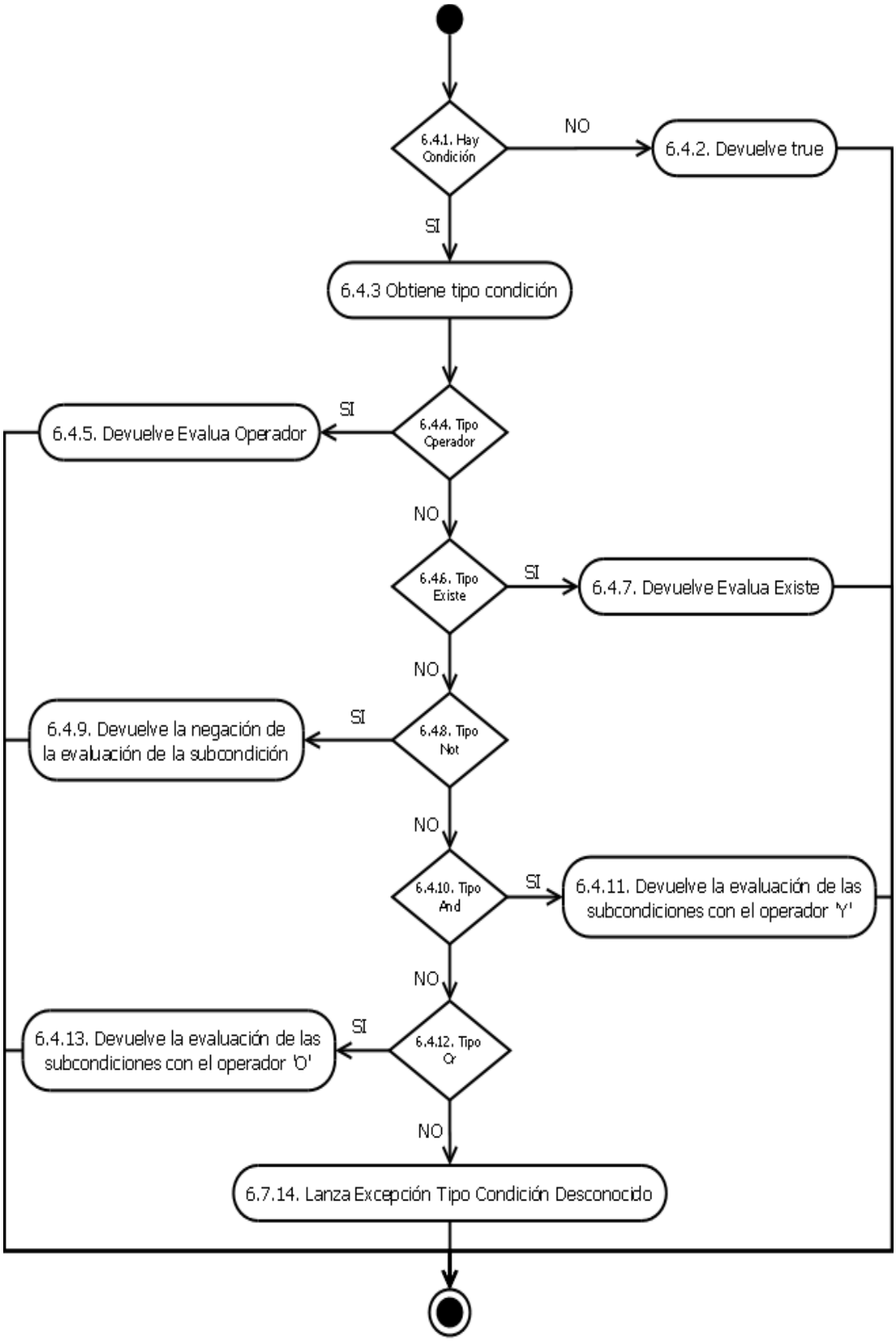
Proceso de traducción



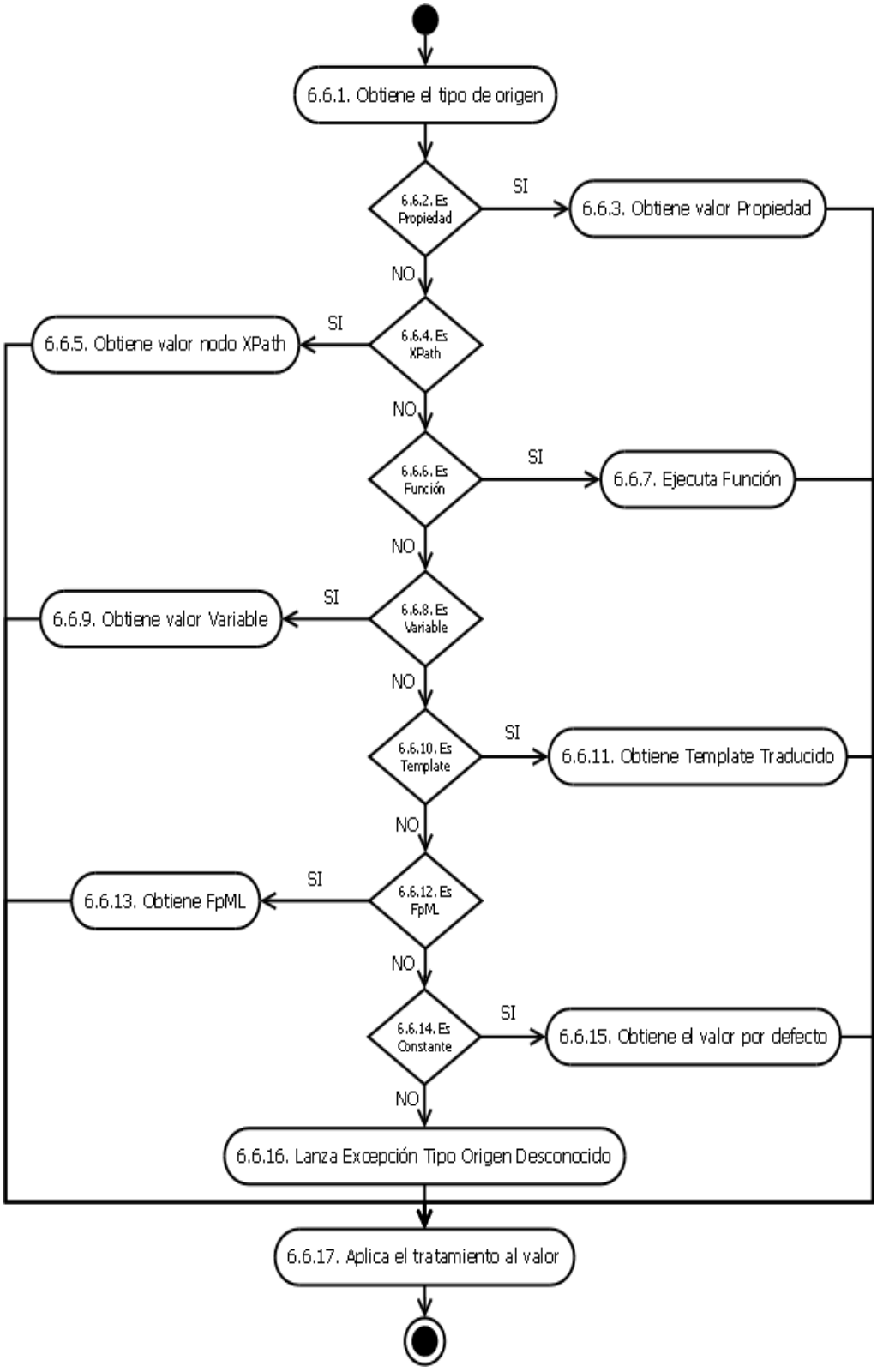
Traducción de una operación



Evaluación de una condición de transformación



Obtención del valor origen



Diseño base de datos

comentado en puntos anteriores, Open OTC Viewer tranforma los ficheros FpML en MxML y se cargan en la base de datos MySQL para hacerlas accesibles por el módulo web.

Se ha definido en MySQL el esquema TFM que contiene dos tablas, una dedicada a las operaciones y otra usada para los usuarios.

Tabla operaciones

En la tabla “operaciones” se cargarán los datos más importantes que forman una operación (fechas inicio y fin, business centers, convention dates, generado, etc.), siempre irán asociados al broker que contrató la operación (campo “broker”). Esta asociación operación – broker tiene como objetivo que varios brokers utilicen la herramienta Open OTC Viewer de manera independiente, el módulo web mostrará únicamente las operaciones del broker logueado.

La definición de la tabla de operaciones es la siguiente:

```
CREATE TABLE `operaciones` (
  `USER` varchar(45) DEFAULT NULL,
  `GRUPO` varchar(45) DEFAULT NULL,
  `PORTFOLIO` varchar(45) DEFAULT NULL,
  `CONTRAPARTIDA` varchar(45) DEFAULT NULL,
  `TRADENUMBER` varchar(45) CHARACTER SET dec8 NOT NULL DEFAULT '',
  `TRN_FAMLIY` varchar(45) DEFAULT NULL,
  `TRN_GRP` varchar(45) DEFAULT NULL,
  `ENT_PORTFOLIO` varchar(45) DEFAULT NULL,
  `BROKER` varchar(45) DEFAULT NULL,
  `START_DATE_1` varchar(45) DEFAULT NULL,
  `MATURITY_1` varchar(45) DEFAULT NULL,
  `CAPITAL_1` varchar(45) DEFAULT NULL,
  `PHASE_1` varchar(5) DEFAULT NULL,
  `LEG_1` varchar(5) CHARACTER SET big5 COLLATE big5_bin DEFAULT NULL,
  `SCH_GENLABEL_1` varchar(45) DEFAULT NULL,
  `SHIFTER_1` varchar(45) DEFAULT NULL,
  `BUS_CENTERLABEL_1` varchar(45) DEFAULT NULL,
  `RATECONV_LABEL_1` varchar(45) DEFAULT NULL,
  `START_DATE_2` varchar(45) DEFAULT NULL,
  `MATURITY_2` varchar(45) DEFAULT NULL,
  `CAPITAL_2` varchar(45) DEFAULT NULL,
  `PHASE_2` varchar(5) DEFAULT NULL,
  `LEG_2` varchar(5) DEFAULT NULL,
  `SCH_GENLABEL_2` varchar(45) DEFAULT NULL,
  `SHIFTER_2` varchar(45) DEFAULT NULL,
  `BUS_CENTERLABEL_2` varchar(45) DEFAULT NULL,
  `RATECONV_LABEL_2` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`TRADENUMBER`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Como podemos ver, la tabla “operaciones” contiene los valores que definen cualquier operación FpML. Se ha creado una primary key en el campo “tradenumber” que no permitirá cargar una misma operación duplicada en el sistema.

Tabla usuarios

La tabla de usuarios, también ubicada en el esquema TFM, permitirá dar de alta usuarios y sus respectivas contraseñas.

La tabla se define de la siguiente manera:

```
CREATE TABLE `usuarios` (
  `id` int(5) NOT NULL,
  `nombre` varchar(45) DEFAULT NULL,
  `pass` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Al igual que en la tabla de operaciones, se ha creado una clave primaria (campo id).

El encargado de introducir los usuarios en la tabla será el administrador web. Tenemos el campo “pass” que guardará la contraseña del usuario, este campo debe estar cifrado por motivos de seguridad. Para ello el administrador web dará de alta los usuarios usando un cifrado AES, por lo que el módulo web cifrará la contraseña que introduce el usuario y la comparará con el campo “pass” de la tabla de usuarios. Con este sistema nos aseguramos que si algún atacante está realizando algún ataque man-in-the-middle, la contraseña no vaya en texto plano.

Se darán de alta los usuarios de la siguiente manera:

```
INSERT INTO usuarios VALUES(2,'luis',AES_ENCRYPT('pass','luis'));
```

La función AES_ENCRYPT realiza el cifrado del campo “pass”. Desde el módulo web de Open OTC Viewer se realizará la consulta a la base de datos cifrando el contenido del campo contraseña en la sección de login, por lo que la validación usuario – contraseña se realiza con el campo “pass” cifrado.

```
pSt = connection.prepareStatement("select id from TFM.usuarios where nombre = ? and pass =
AES_ENCRYPT ('pass', ?)");
```

Si el resultado anterior da resultado, la validación de usuario se habrá realizado correctamente.

Módulo web

El módulo web de Open OTC Viewer permite a los usuarios disponer en cualquier dispositivo con conexión a Internet sus operaciones contratadas en Open OTC Viewer.

A continuación veremos como se ha diseñado el módulo web de Open OTC Viewer.

Servlets

Los Servlet[xv] son clases java que responden dinámicamente a request o peticiones hechas por un cliente y construyen una respuesta al mismo. El servlet se ejecuta en un contenedor web, el cual es responsable del ciclo de vida del servlet.

Los servlets en Open OTC Viewer derivan de clase Java javax.servlet.http.HttpServlet, por lo que debemos definir los métodos doGet() o doPost(), que recibirán las petición HTTP GET y HTTP POST respectivamente, estos métodos llamarán al método doProcess() que es el encargado de redirigir la ejecución al dispatcher apropiado.

Para que Java EE sea capaz de trabajar con servlets, se deben definir en el fichero web.xml de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>OTC Viewer</display-name>
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <description></description>
    <display-name>FrontServlet</display-name>
    <servlet-name>FrontServlet</servlet-name>
    <servlet-class>servlet.FrontServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FrontServlet</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>20</session-timeout>
  </session-config>
</web-app>
```

Sombreado en negrita vemos la definición del servlet para Java EE. Básicamente todas las peticiones HTTP que contengan el literal *.do serán tratadas por la clase FrontServlet. Por ejemplo, el formulario de ingreso de usuario y contraseña se realiza en la petición login.do (<form action="login.do" method="post">). En la clase FrontServlet, dentro del método init se realizará la asignación al controlador adecuado (controlers.put("/login.do", new LoginControler());).

Sesiones

Una de las funciones que nos proporciona Java EE es el uso de sesiones. Las sesiones son muy útiles en webs dinámicas, ya que podemos crear, borrar o mantener variables durante el ciclo de vida del usuario dentro de Open OTC Viewer.

La interface HttpSession es la que proporciona los métodos necesarios para mantener sesiones. El objeto se obtiene de la interfaz HttpServletRequest con:

```
public HttpSession getSession(boolean);
public HttpSession getSession();
```

Se pueden asociar pares nombre/valor a la sesión:

Añadir valor: setAttribute(String, Object)
Obtener valor: getAttribute(String)

Open OTC Viewer usa sesiones para mantener el valor del usuario logueado y las operaciones que tiene asociadas dicho usuario.

Ficheros JSP

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

JSP puede ser visto como una abstracción de alto nivel de los servlets Java. Las JavaServer Pages son traducidas a servlets en tiempo real; cada servlet es guardado en caché y reusado hasta que la JSP original es modificada.

Open OTC Viewer contiene ficheros JSP que realizan funciones de log in y log out, y para la visualización de las operaciones de un determinado usuario.

En Eclipse estos ficheros se ubican dentro de la sección WEB-INF.

Desarrollo

A continuación se detallan las acciones realizadas en el desarrollo de Open OTC Viewer.

Preparación del entorno

El primer paso en el desarrollo fue la preparación del entorno, es decir, la instalación de todas las herramientas descritas en la fase de diseño.

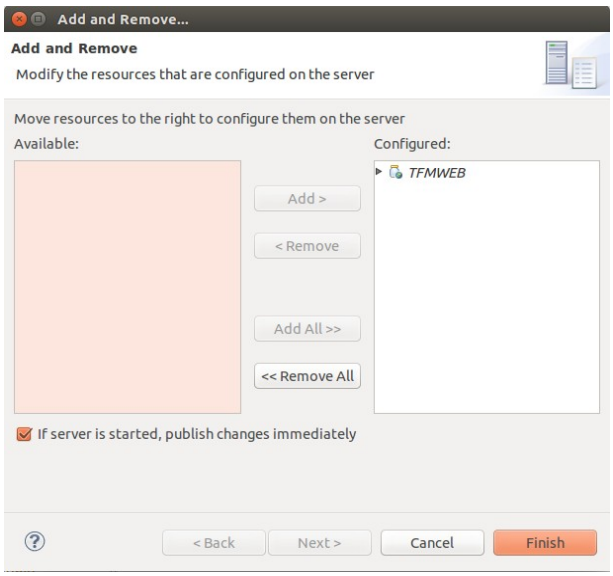
1. Sistema operativo: Ubuntu 14.04. Ya disponía de dicho S.O. instalado en mi ordenador personal.
2. IDE Eclipse: Instalación a partir del "Centro de Software de Ubuntu". Ubuntu proporciona un centro de software donde podemos encontrar los programas más difundidos, y nos proporciona un sistema de instalación de fácil utilización.
3. Instalación de módulos extra en Eclipse para programación web. Se realiza la instalación a través del propio Eclipse. Help → Install new software...
4. Instalación de EGIT (sistema de control de versiones). Se realiza la instalación desde el propio módulo de instalación de paquetes extra de Eclipse.
5. Instalación de GlassFish. Se realiza la instalación desde el propio módulo de instalación de paquetes extra de Eclipse.
6. Instalación de MySQL. Se realiza la instalación a través del "Centro de Software de Ubuntu". Se instala el software "MySQL Workbench" que facilita la configuración del servidor además de proporcionarnos un cliente de base de datos.
7. Creación en Eclipse de los proyectos TFM y TFMWEB. Eclipse se organiza en proyectos. Open OTC Viewer contendrá dos proyectos, uno para el módulo de transformación y carga en base de datos, que utiliza el lenguaje de programación Java (proyecto TFM), y otro para el módulo web, que utiliza lenguaje de programación Java EE (proyecto TFMWEB).
8. Creación de las rutas sobre las que trabajará Open OTC Viewer. Los ficheros origen y destino deberán estar ubicados en el filesystem de Ubuntu. En el $HOME del usuario de Ubuntu se han creado las siguientes subcarpetas:

```
0:luis@~/TFM$ ls
FICHEROS_PRUEBA  in_fpml  mueve.sh  out_fpml  out_mxml
0:luis@~/TFM$
```

9. El script **mueve.sh** se ocupa de mover los ficheros a su posición original con cada ejecución de Open OTC Viewer. Su contenido es el siguiente:

```
#!/bin/sh
cd /home/luis/TFM/out_fpml
mv * ../in_fpml
rm /home/luis/TFM/out_mxml/*
echo "Movimiento de ficheros finalizado"
exit 0
```

10. Configuración del proyecto TFMWEB para que trabaje con GlassFish. Desde Eclipse entramos en la configuración de GlassFish: Properties → Add and Remove...

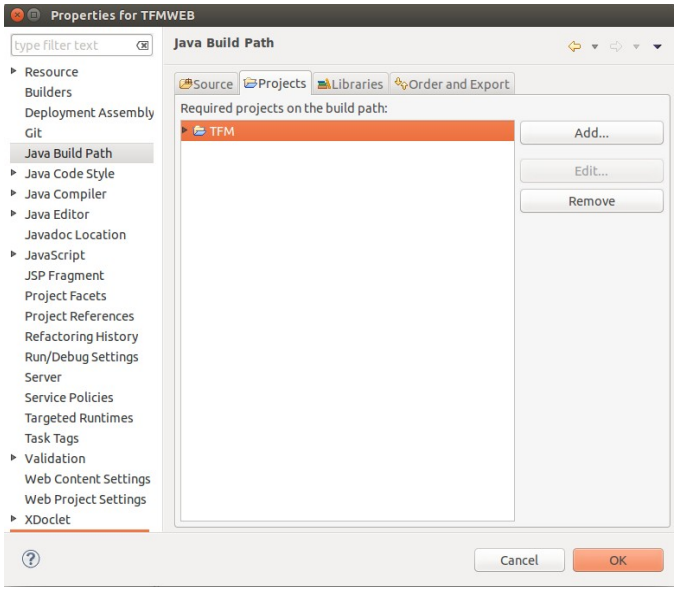


Eclipse nos muestra los posibles proyectos que pueden ser incluidos en GlassFish, en este caso nos aparece el único proyecto que se ha creado como Java EE (TFMWEB), pulsamos finalizar y de esta manera Eclipse ya es capaz de ejecutar el código de TFMWEB en GlassFish.

11. Añadir al proyecto TFM el conector a la base de datos MySQL. Descargamos desde la web de Oracle el conector **mysql-connector-java-5.1.6-bin.jar.** Desde la configuración "Configure Build Path" del proyecto TFM, añadimos el jar anterior.

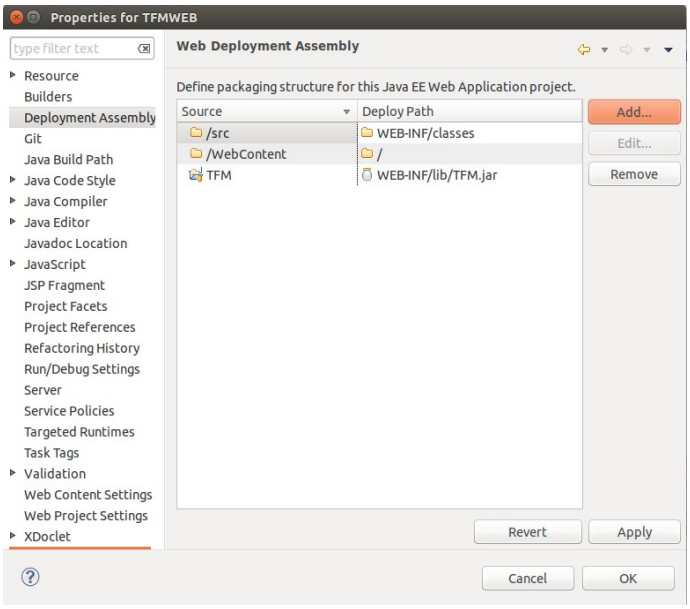
12. Los dos proyectos (TFM y TFMWEB) deben trabajar conjuntamente para el producto final, es decir, TFMWEB necesitará realizar peticiones a métodos ubicados en TFM, por lo que es necesario "unir" los proyectos de alguna manera. Eclipse permite lo anterior realizando lo siguiente: Desde la configuración del proyecto TFMWEB.

    Properties → Java Build Path → Projects → Add (Añadimos proyecto TFM)



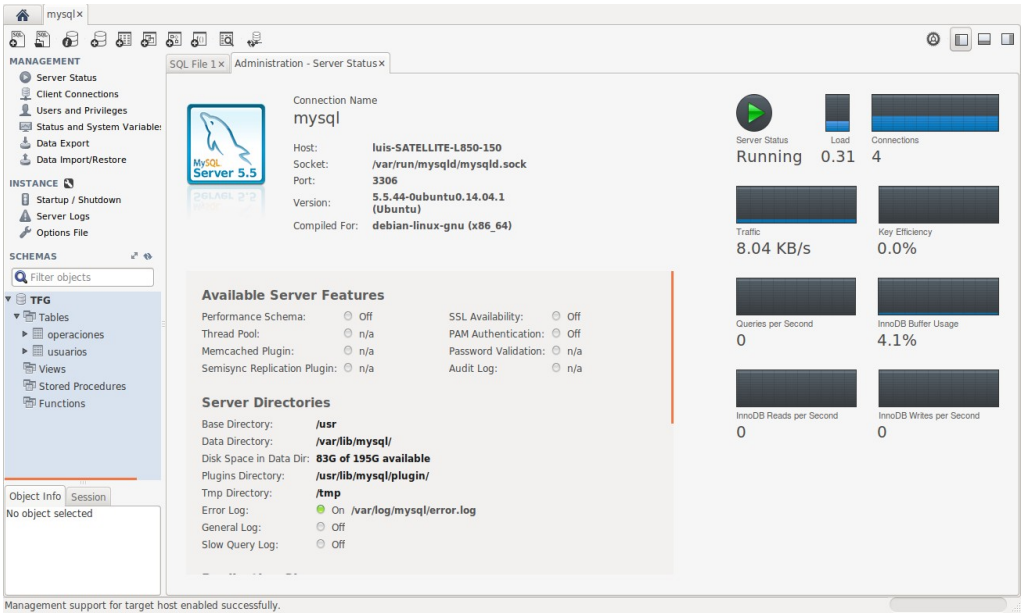
Una vez añadido el proyecto en Java Build Path.

Properties → Deployment Assembly → Add → Project (Añadimos el proyecto TFM)



Gracias a la configuración anterior, podemos realizar peticiones desde TFMWEB a métodos de TFM.

13. Creación del esquema TFM en MySQL. MySQL se organiza mediante el uso de esquemas, donde se definen las tablas “operaciones” y “usuarios” según la definición comentada en puntos anteriores.



En la imagen anterior se puede ver el servidor MySQL corriendo en el sistema Ubuntu con el esquema TFM y sus tablas.

14. Añadir proyectos TFM y TFMWEB a EGIT. Sobre ambos proyectos realizamos lo siguiente:

    Propiedades del proyecto → Team → Share project

    Realizamos el primer “Commit”:

    Propiedades del proyecto → Team → Commit

Conversor FpML a MxML

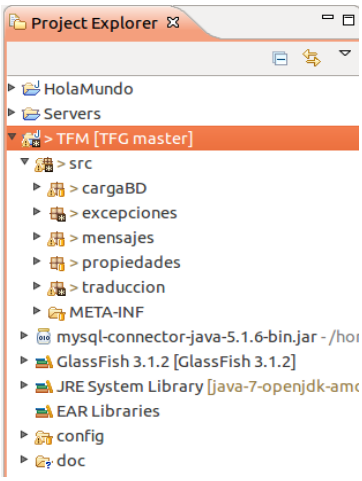
El módulo conversor de FpML a MxML es la parte más extensa de Open OTC Viewer. Se ha realizado un análisis de los parámetros origen en FpML y sus respectivos destinos en MxML. Este módulo incluye la creación del fichero dtd, las plantillas MxML de transformación que trabajan conjuntamente con el fichero dtd, y por último, la programación en lenguaje Java para realizar las funciones descritas en las plantillas.

En muchos casos, las clases utilizan valores constantes, de forma repetida y en múltiples puntos. Estos valores no deben utilizarse de forma literal y se debe proveer de un punto de acceso único a cada valor. Para ello se utilizan constantes que almacenamos en interfaces que después implementan las clases implicadas. Creamos las interfaces de constantes en el mismo paquete que la clase cuyas constantes almacena y con el nombre de la clase al que se añade el sufijo Cte.

Ciertas clases del módulo de transformación contendrán una única instancia a lo largo de la ejecución, esto lo conseguimos mediante el patrón singleton.

Otro punto importante es el uso de excepciones, se ha creado una clase TFMExcepciones que hereda de la clase Java Exception. Cada parte del módulo de transformación contiene una clase dedicada a la gestión de las posibles excepciones que puedan ocurrir, todas ellas heredan de TFMExcepciones, por lo que deberán implementar los métodos de la clase Java Exception.

La programación del módulo en Eclipse se ha dividido en paquetes. A continuación veremos los paquetes que se han creado y su función.



- **Traducción**: La clase Traduccion es la clase principal del proceso de traducción de operaciones FpML a MxML. Esta clase contiene el método main que inicia el proceso de traducción. Se encarga de obtener los ficheros FpML y llamar a los métodos que realizan su traducción. También reporta los resultados de las traducciones en el log.

  Clase Padre :
  java.lang.Object

  Interfaces :
  traduccion.TraduccionCte

  Responsabilidades :
  Escribe en el log la traza de comienzo del proceso de traducción de los ficheros FpML.
  Obtiene todos los ficheros FpML.
  Lanza la traducción de cada uno de los ficheros FpML.
  Da de alta la operación de la tabla de operaciones de la base de datos.
  Escribe en el log el resultado de la traducción de cada fichero.
  Escribe en el log el resultado final del proceso completo de traducción.
  Captura las excepciones producidas en el proceso de traducción y reporta la información.

  Paquete :
  traduccion

- **Operación**: La clase Operacion es una clase modelo que contiene la información de una operación: FpML, tipo de producto, nombre, MxML, etc.

  Clase Padre :
  java.lang.Object

  Interfaces:
  traduccion.OperacionCte

  Paquete :
  traduccion

- **OperacionControlador**: La clase OperacionControlador es la clase contiene los métodos que obtienen la información de una operación: su tipo de producto y sus características. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces:
  operacion.OperacionCte

Responsabilidades :
- Obtener el tipo de producto de la operación.
- Obtener las características de la operación.
- Obtener el nombre de una operación del fichero.

Paquete :
traduccion

- **Operacion:** La clase Condicion es la clase contiene los métodos que implementan las condiciones posibles en una transformación. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces:
  condicion.CondicionCte

  Responsabilidades:
  Evalúa condiciones de tipo operador (igual, mayor, menor).
  Evalúa condición de tipo existe.
  Evalúa condiciones lógicas (not, and, or).

  Paquete :
  traduccion.

- **TraductorDocumentos:** La clase TraductoDocumentos es la clase obtiene los distintos tipos de documentos XML que son necesarios en la traducción. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces :
  traduccion.TraductorCte

  Responsabilidades :
  Obtener los documentos XML que son necesarios en la traducción.
  Mantener una caché de documentos XML para mejorar el rendimiento de la obtención repetitiva de un mismo documento.

  Paquete:
  traduccion

- **TraductorVariables:** La clase TraductorVariables mantiene la lista de variables del proceso de traducción. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Responsabilidades :
  Obtiene el valor de una variable de la lista de variables.
  Establece el valor de una variable de la lista de variables.

  Paquete:
  traduccion

- **TraductorFunciones:** La clase TraductorFunciones contiene las funciones de obtención y tratamiento de datos que se utilizan en el proceso de traducción. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre:
  java.lang.Object

  Interfaces:
  traduccion.TraductorCte

  Responsabilidades:
  Proveer de funciones de obtención de datos para el proceso de traducción.
  Proveer de funciones de tratamiento de datos para el proceso de traducción.

  Paquete:
  traduccion

- **AccesoBaseDatos:** La clase AccesoBaseDatos contiene las funciones de acceso a base de datos. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre:
  java.lang.Object

  Interfaces :
  cargaBD.AccesoBaseDatosCte

  Responsabilidades :
  Conecta a la base de datos.

Ejecuta consultas.
Inserta y actualiza datos.

Paquete :
cargaBD.datos

- **CargaBD:** Clase encargada de la carga de operativa en la base de datos. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre:
  java.lang.Object

  Interfaces:
  cargaBD.CargaBDCte

  Responsabilidades:
  Realiza la carga en la base de datos con todos los valores que forman una operación.

  Paquete:
  cargaBD

- **XmlUtils:** La clase XmlUtils contiene las funciones de manejo de documentos XML.
  La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces :
  traduccion.XmlUtilsCte

  Responsabilidades:
  Lee documentos XML de fichero.
  Provee de acceso a nodos y atributos.
  Permite modificar los valores de nodos y atributos.
  Escribe documentos XML en fichero.

  Paquete:
  traduccion

- **Propiedades:** La clase Propiedades permite acceder a las propiedades de configuración definidas en el fichero de propiedades. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces:

propiedades.PropiedadesCte

Responsabilidades :
Obtener el valor de las propiedades

Paquete :
propiedades

- **FicheroUtils:** La clase FicherosUtils contiene las funciones de manejo de ficheros. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces :
  propiedades.FicherosUtilsCte

  Responsabilidades :
  Copiar ficheros
  Mover fichero
  Buscar ficheros
  Comprobar si un fichero existe

  Paquete:
  propiedades

- **GestorMensajes:** La clase GestorMensajes se encarga gestionar los mensajes que produce el proceso de traducción, tanto de información como de error. La clase sigue el patrón singleton que garantiza que la clase sólo tendrá una instancia y proporciona un punto de acceso global a sus funciones.

  Clase Padre :
  java.lang.Object

  Interfaces:
  mensajes.GestorMensajesCte

  Responsabilidades:
  Construye mensajes a partir de un identificador de mensaje y unas variables.
  Transformar una excepción en un mensaje explicativo para el usuario.

  Paquete:
  mensajes

- **Excepciones:** Las clases informan de los errores que se producen en la ejecución de sus funciones mediante el uso de Excepciones. Las Excepciones permiten transmitir fácilmente los errores producidos al tiempo que almacenan información sobre el motivo del error y el punto en que se produce.
  Para poder utilizar todas las funcionalidades del Gestor de Mensajes creamos una clase genérica de tipo Exception que además almacena variables para utilizar la construcción dinámica de mensajes: TFMExceptiones.
  Todas las excepciones lanzadas por el proceso de traducción e inserción heredan de esta clase.
  La clase TFMExceptiones es la excepción genérica de la que heredan todas las excepciones lanzadas por el proceso de traducción e inserción. Tiene la particularidad de que almacena variables que se utilizarán para completar el mensaje de error.

  Clase Padre:
  java.lang.Exception

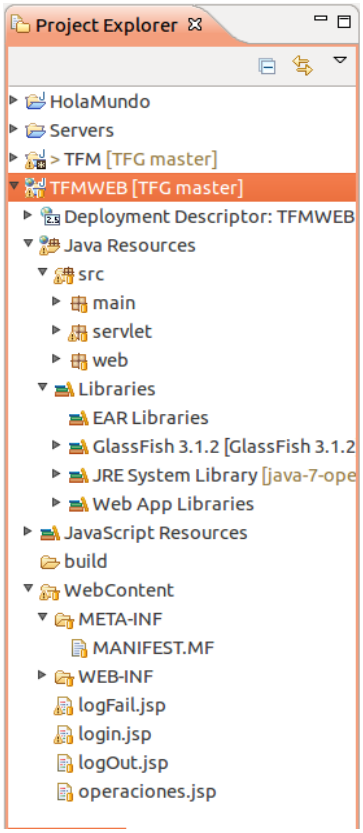
  Responsabilidades :
  Almacenar el identificador del mensaje de error.
  Almacenar las variables necesarias para completar el mensaje de error.

  Paquete:
  excepciones

Módulo web

El módulo web de Open OTC Viewer permite la visualización de la operativa contratada por un determinado usuario (informado como 6º parámetro del transformador). Como hemos comentado anteriormente, este módulo ha sido programado con Java EE, haciendo uso de servlets y páginas JSP.

Vamos a definir las diferentes clases y objetos de los que está formado este módulo.



- **Operacion**: Clase Java utilizada para identificar una operación IRS y todas las características que la definen. Contiene todos los métodos que permiten obtener y establecer todos los parámetros de una operación.

  Clase padre:
  java.lang.Object

  Responsabilidades:
  Obtiene y establece los parámetros que definen una operación.

  Paquete:
  main

- **User:** Clase utilizada para definir un usuario con acceso al módulo web. En Open OTC Viewer un usuario se define únicamente con su usuario y su contraseña.

  Clase padre:
  java.lang.Object

  Responsabilidades:
  Obtiene el usuario y la contraseña del usuario a loguear.

  Paquete:
  main

- **UserBD:** Clase utilizada para los accesos a la base de datos desde el módulo web.

  Clase padre:
  java.lang.Object

  Responsabilidades:
  Realiza búsquedas en la base de datos para loguear usuarios. Obtiene las operaciones cargadas en la base de datos para un usuario concreto.

  Paquete:
  main

- **FrontServlet:** Clase de tipo Servlet. Encargada de recibir las peticiones del usuario y redirigirlas al controlador adecuado.

  Clase padre:
  javax.servlet.http.HttpServlet

  Responsabilidades:
  Recuperar las peticiones HTTP realizadas por el usuario desde el módulo web. Cada petición es redirigida a la pieza Java encargada de gestionar dicha petición.

Paquete:
servlet

- **LoginControler:** Clase encargada de realizar el proceso de login de usuario.

  Clase padre:
  java.lang.Object

  Interfaces:
  web.IControladorWeb

  Responsabilidades:
  Realiza la validación de usuario/contraseña de usuario, carga sus operaciones en variable y redirige al JSP adecuado.

  Paquete:
  web

- **LogOutControler:** Clase encargada de realizar el logout de usuario.

  Clase padre:
  java.lang.Object

  Interfaces:
  web.IControladorWeb

  Responsabilidades:
  Mata la sesión del usuario activo y redirige al JSP adecuado.

  Paquete:
  web

JSP

El módulo web de Open OTC Viewer está formado por, además del código Java EE que hemos visto en el punto anterior, ficheros JSP que forman las páginas web que el usuario visualiza en el módulo web.

El código jsp se integra junto a HTML para formar el código web de Open OTC Viewer.

- **login.jsp**: Fichero web que muestra los campos usuario y contraseña para acceder al módulo web. Crea un formulario y redirige a la clase LoginControler mediante el uso del servlet y la petición “login.do”.

  <form action="login.do" method="post">

  Es el punto de entrada del módulo web.

- **logFail.jsp:** Fichero web ejecutado cuando la combinación usuario / contraseña no ha sido encontrada en la tabla de usuarios. Únicamente muestra un mensaje de error.

- **logOut.jsp:** Fichero ejecutado cuando el usuario realiza logout, es decir, sale del aplicativo. Muestra un mensaje de despedida.

- **operaciones.jsp:** Fichero principal del módulo web, encargado de mostrar las operaciones del usuario logueado correctamente. Recibe desde el controlador la variable “operacione” que contiene una lista con todas las operaciones contratadas por el usuario logueado.

  Realiza una validación de la variable “operaciones” por si estuviera vacía (en ese caso el usuario no tiene operativa contratada), en caso de no estarlo, realiza el recorrido de la lista mostrando los datos en formato tabla.

```
<c:choose>
<c:when test="${empty operaciones}">
</c:when>

<c:otherwise>
	<!-- Tabla con los productos disponibles obtenidos de la BBDD. -->
	<table border="0.5" width=10% class="texto" cellspacing="10" align="center">
		<tr>

			<!-- Añadir los nombres de las columnas que forman la operación -->
			<th>Trade Number</th>
			<th>Portfolio</th>
			<th>Broker</th>
			<th>Tipología</th>
			<th>Start Date 1</th>
			<th>Maturity 1</th>
			<th>Capital 1</th>
			<th>Schedule Gen. 1</th>
			<th>Shifter 1</th>
			<th>Business Center Label 1</th>
			<th>Rate Conv. Label 1</th>
			<th>Start Date 2</th>
			<th>Maturity 2</th>


			<!-- Recorrer la tabla operaciones sacando a tabla los valores de cada
operacion -->
			<c:forEach var="ope" items="${operaciones}">
				<tr>
					<!-- <td><input type="submit" name=""
value="Modifica"/></td> -->
					<td>${ope.tradeNumber}</td>
					<td>${ope.portfolio}</td>
					<td>${ope.broker}</td>
					<td>${ope.trn_grp}</td>
```

```
                                    <td>${ope.start_date_1}</td>
                                    <td>${ope.maturity_1}</td>
                                    <td>${ope.capital_1}</td>
                                    <td>${ope.sch_genlabel_1}</td>
                                    <td>${ope.shifter_1}</td>
                                    <td>${ope.bus_centerlabel_1}</td>
                                    <td>${ope.rateconv_label_1}</td>
                                    <td>${ope.start_date_2}</td>
                                    <td>${ope.maturity_2}</td>

                                </tr>
                            </c:forEach>
                        </tr>
                    </table>
                </c:otherwise>
                </c:choose>
```

Como se puede observar, utilizamos código jsp para comprobar si la variable “operaciones” está vacía y luego recorrerla mediante un for y mostrándola en formato tabla.

Hoja de estilos

Con el fin de proporcionar al módulo web de Open OTC Viewer una interfaz más amigable y agradable, se va a incluir una hoja de estilos css a los ficheros jsp.

La hoja de estilos se ha descargado desde Internet en la web http://www.freecsstemplates.org, se distribuye bajo licencia “Creative Commons Attribution 2.5” por lo que es compatible con nuestro desarrollo.

Con la siguiente nota especificamos que la hoja de estilos proviene desde un creador externo.

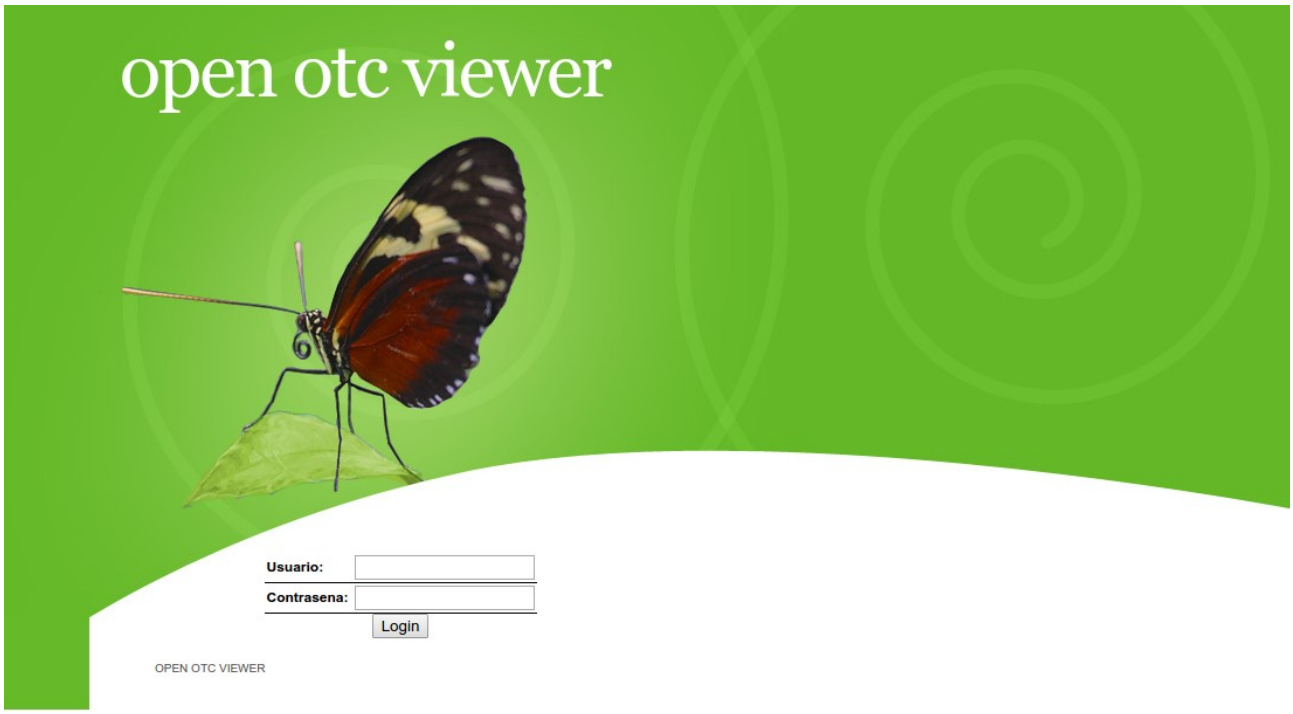
```
/*
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Commons Attribution 2.5 License
*/
```

En la hoja de estilos se definen el diseño que tendrá cada parte de los ficheros HTML, la hoja define los estilos que se usarán para cada etiqueta HTML contenida dentro de los ficheros JSP.

En los ficheros HTML se ha incluido el siguiente código para indicarle que debe usar dicha hoja de estilos:

```
<!-- Escpecifica la hoja de estilos que usara el jsp -->
<link rel="stylesheet" type="text/css" href="default.css" />
```

De esta manera conseguimos proporcionar a Open OTC Viewer de una interfaz amable y sencilla.



Documentación

La documentación de Open OTC Viewer se ha realizado mediante Javadoc[1]. Javadoc es un generador de documentación creado por SUN Microsystems para lenguaje Java. Mediante comentarios de tipo Java Doc en los ficheros .java del aplicativo, podemos generar una documentación en formato HTML completa y fácil de interpretar.

1 https://en.wikipedia.org/wiki/Javadoc

Manual de usuario

En este capítulo vamos a definir el manual de usuario una vez que los requisitos técnicos de Open OTC Viewer han sido puestos en marcha.

Parámetros del transformador

La herramienta de transformación debe recibir una serie de parámetros que determinan ciertos aspectos de la transformación. En concreto existen 6 parámetros que se deben especificar obligatoriamente.

1. Portfolio. Este valor es meramente organizativo. Lo deberá especificar el usuario propietario de la operación.
2. Contrapartida. Este valor también deberá ser especificado por el usuario propietario de la operación.
3. Ruta donde se ubicarán los ficheros FpML a traducir.
4. Ruta directorio ficheros FpML traducidos.
5. Ruta del directorio donde se depositarán los ficheros MxML traducidos.
6. Usuario que contrata la operación. Este valor es importante para el módulo web, ya que sólo se mostrarán las operaciones contratadas por el usuario logueado, el cual se informa en este parámetro.

Por ejemplo:

IRSPORTF 0001897 /home/luis/TFM/in_fpml /home/luis/TFM/out_fpml /home/luis/TFM/out_mxml ivonne

El resultado de la ejecución anterior, en caso de no producirse errores, será:

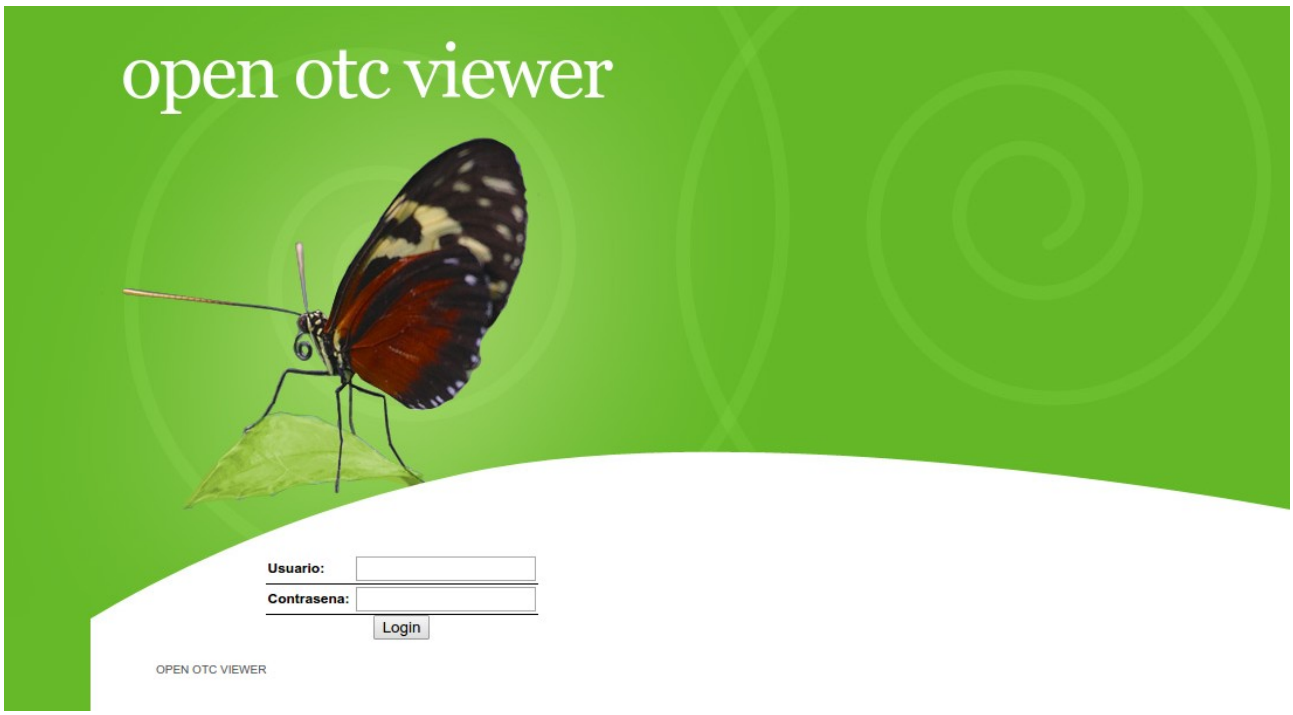
1. Los ficheros origen FpML obtenidos desde la ruta del parámetro 3º se ubicarán en el directorio especificado en el 4º parámetro.
2. Los ficheros MxML, resultado de la transformación, se ubicarán en el directorio especificado en el 5º parámetro.
3. La base de datos de operaciones se habrá rellenado con la información importante de los ficheros FpML.

Módulo web

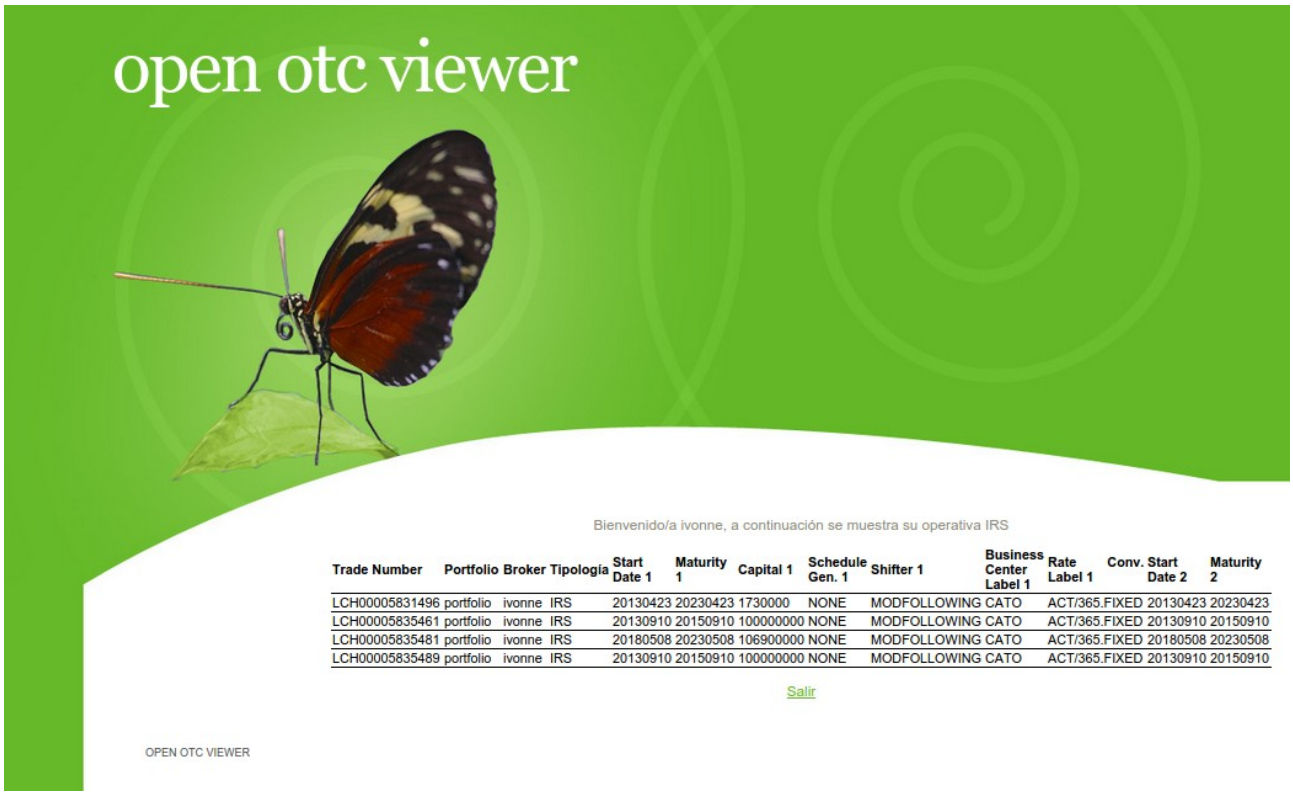
Cuando el transformador se ha ejecutado sin errores, la base de datos se ha poblado con las operaciones IRS que contenían los ficheros FpML, el siguiente paso será su visualización desde el módulo web.

Como requisito del módulo web, el administrador de la base de datos deberá dar de alta aquellos usuarios con acceso al aplicativo.

La web inicial de Open OTC viewer es la siguiente:

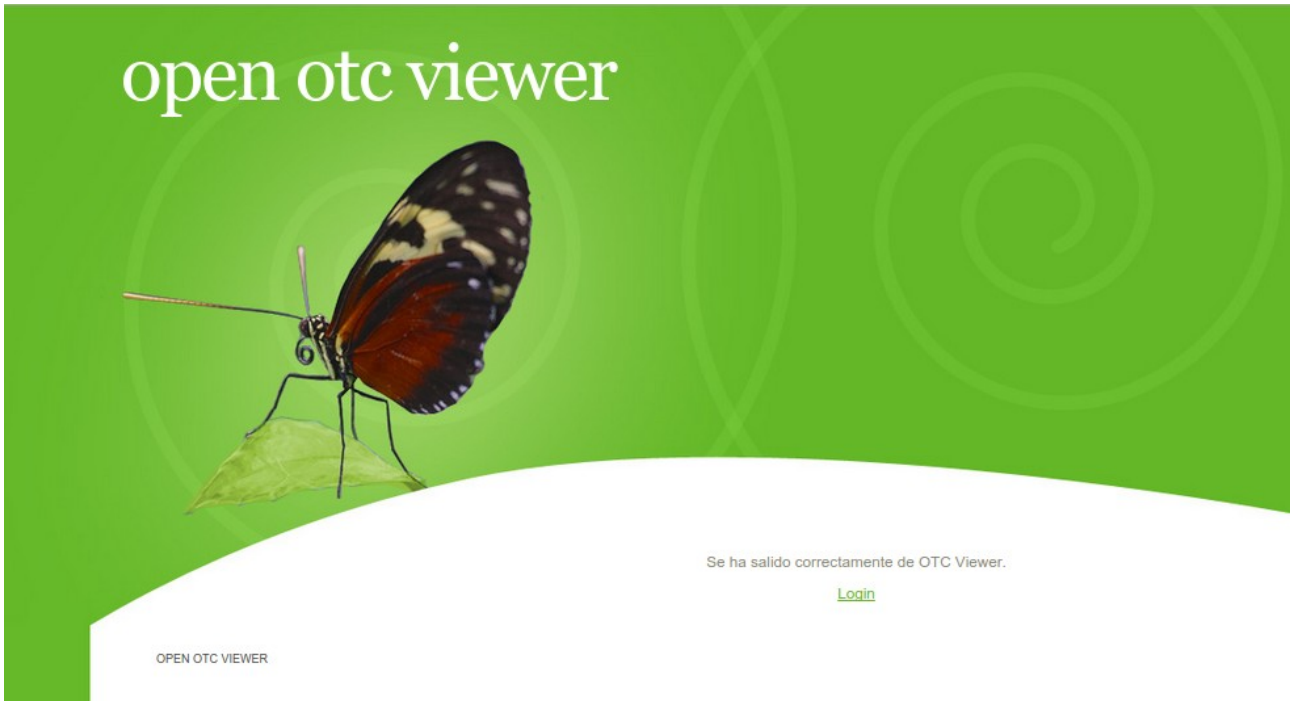


Habrá que introducir un usuario existente en la tabla de usuarios. Para el ejemplo vamos a usar el usuario “ivonne”, el cual tiene 4 operaciones contratadas:

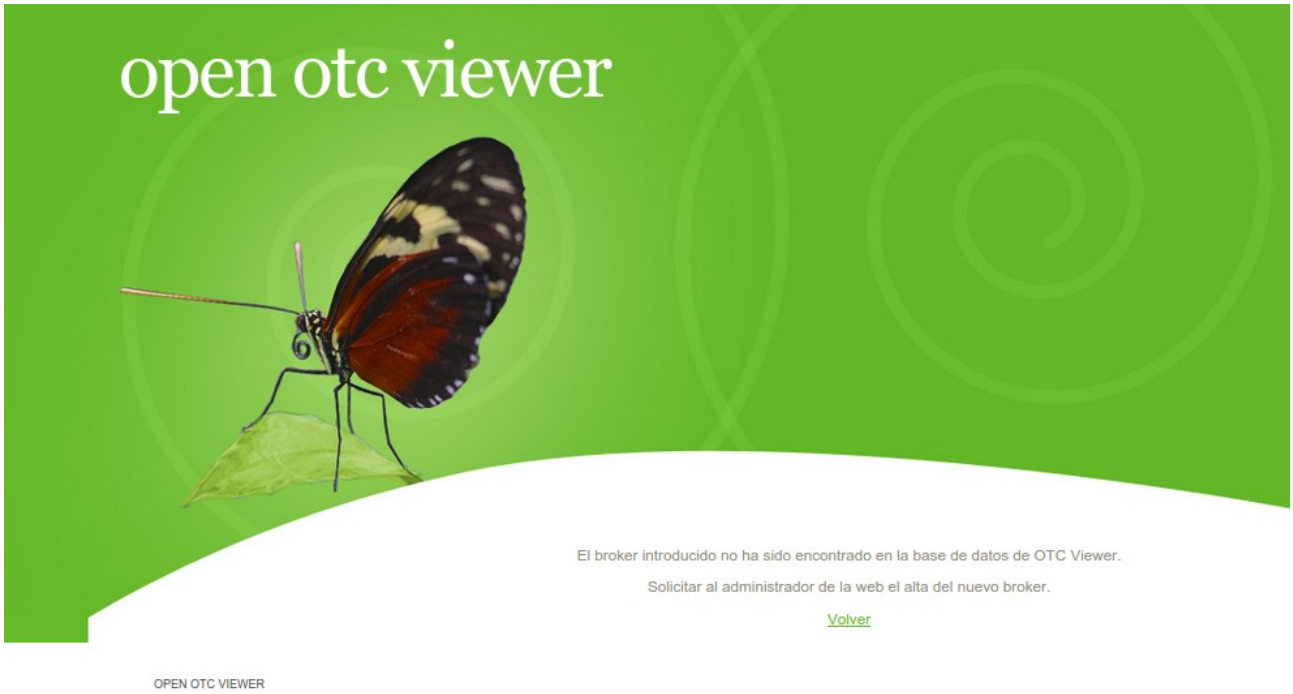


Desde esta página podrá visualizar su operativa contratada.

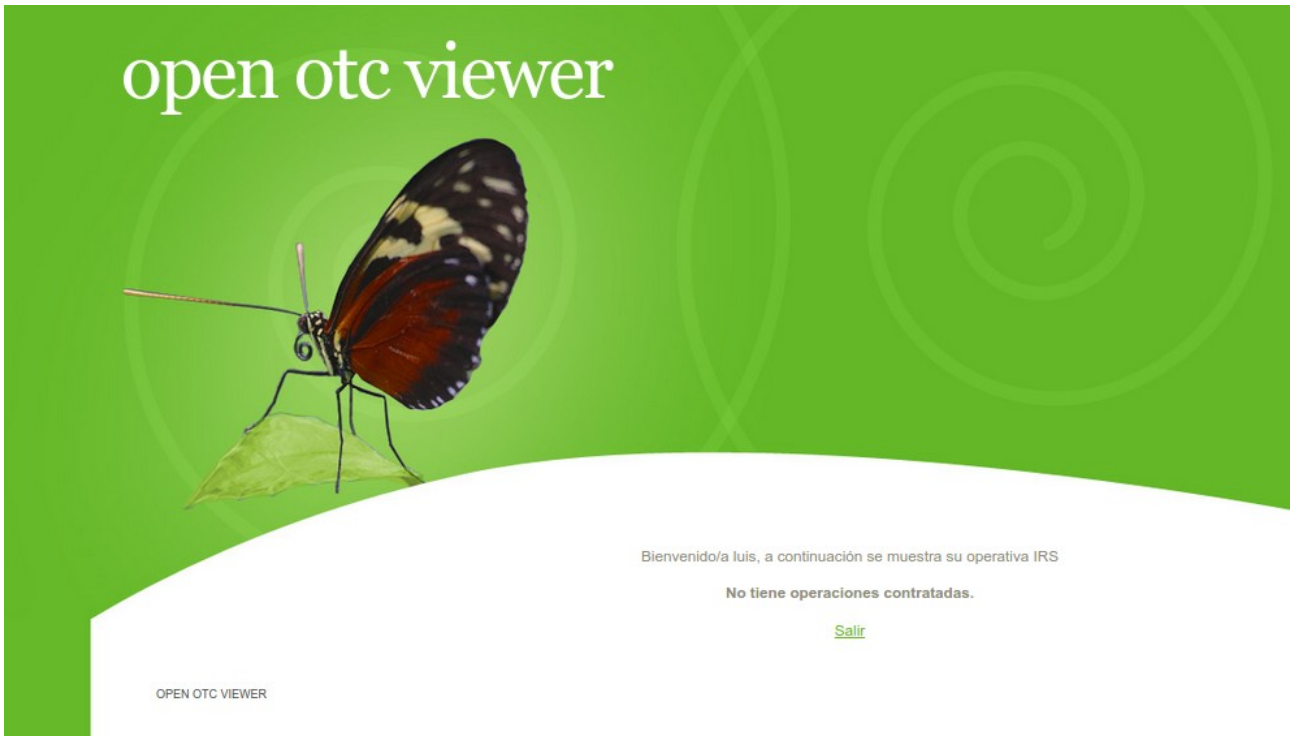
Pulsando “Salir” la sesión del usuario logueado se destruye y se muestra la siguiente web:



Se puede dar el caso de introducir un usuario inexistente en la base de datos, en ese caso Open OTC Viewer mostrará el siguiente mensaje indicando que se debe poner en contacto con el administrador de la web:



Por último, se puede dar el caso de que el usuario exista en la base de datos pero no tenga operativa contratada, en ese caso Open OTC Viewer mostrará el siguiente menaje:


open otc viewer
Bienvenido/a luis, a continuación se muestra su operativa IRS
No tiene operaciones contratadas.
Salir
OPEN OTC VIEWER

Conclusiones

El resultado del desarrollo es el software Open OTC Viewer, el cual permite la conversión de ficheros FpML (representando una operación del tipo Interest Rate Swap) a MxML (misma operación pero en formato que Murex es capaz de importar). Este módulo puede ser usado por una gran cantidad de organizaciones dedicadas a la inversión en mercados OTC. Además, se ha añadido un módulo web para visualizar la operativa transformada o contratada por un determinado usuario, lo que proporciona a Open OTC Viewer un punto más de utilidad. Uno de los objetivos principales del proyecto era crear una plataforma Open Source que rompiera un poco con la tendencia que siempre ha caracterizado a este sector, el ocultismo.

Dentro del ámbito personal, el desarrollo del proyecto me ha proporcionado conocimientos funcionales sobre operaciones del tipo Interest Rate Swap (uno de los productos más usados en mercados OTC). Por otro lado, el análisis realizado sobre los ficheros FpML, me ha permitido conocer uno de los estándares más usados dentro de la banca de inversión, lo cual me va a proporcionar en el futuro una experiencia que antes del proyecto no tenía, y que considero importante dentro del sector en el que me estoy desarrollando. Por otro lado, Open OTC Viewer está formado por varias tecnologías, en especial el lenguaje de programación XSL, muy usado en informática para la transformación de ficheros basados en XML. Actualmente existen infinidad de lenguajes basados en XML, por lo que es muy común encontrarte en la situación de tener que transformar de un formato a otro, gracias al desarrollo del proyecto, he conseguido afianzar mis conocimientos sobre XSL. En menor medida, el resto de tecnologías usadas (servidor web, servidor base de datos, Java EE, servlets, sesiones, etc), aunque ya las conocía del pasado, me han permitido recordar conocimientos olvidados y que considero importantes.

La conclusión general del proyecto es muy satisfactoria, se ha creado la herramienta cumpliendo los objetivos iniciales y adquiriendo un conocimiento en varios aspectos importantes dentro de la ingeniería informática.

Anexo

GNU Free Documentation License

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or

PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY
If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS
You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous

versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.
If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS
You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one

section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS
You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS
A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION
You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE
The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and

conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING
"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those

products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10

makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work,

and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit. Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source. This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge. You need not require recipients to copy the
Corresponding Source along with the object code. If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source. Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.

A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own

removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a)

provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all. For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work. The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time. Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number. If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation. If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

Later license versions may give you additional or different
permissions. However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM

IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Bibliografía

i. OTC: https://es.wikipedia.org/wiki/Mercado_extraburs%C3%A1til
ii. Interest Rate Swap: https://en.wikipedia.org/wiki/Interest_rate_swap
iii. FpML: http://www.fpml.org/
iv. MySQL: https://www.mysql.com/
v. GlassFish: https://glassfish.java.net/
vi. Egit: https://www.eclipse.org/egit/?gclid
vii. Eclipse: https://eclipse.org/home/index.php
viii. Murex: https://www.murex.com/
ix. Eclipse Public License: https://www.eclipse.org/legal/epl-v10.html
x. Calypso: http://www2.calypso.com/
xi. London Clearing House: http://www.lchclearnet.com/home
xii. DTD: https://en.wikipedia.org/wiki/Document_type_definition
xiii. JSP: https://en.wikipedia.org/wiki/JavaServer_Pages
xiv. Apache2: https://www.apache.org/
xv. Servlet: https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/Servlet.html