Configuración de Ambientes Locales

El presente documento tiene como objetivo describir los pasos fundamentales para configurar los ambientes locales necesarios para ejecutar las aplicaciones de “Efluentes”.

Denominamos “efluentes” a un grupo de aplicaciones desarrolladas en plataforma Java. A saber:

* Gestión Aplicaciones
* Gestión Territorial
* Servicio AFIP
* Gestión Personas
* Gestión de Efluentes Líquidos
  + Web Interna
  + Web Externa

Requerimientos

* JAVA Plaform JDK 8
  + No sirve solamente el JRE, debe ser necesariamente JDK
  + Solo sirve la versión oficial de Oracle
* Netbeans IDE 8.1(o posterior)
* Servidor Wildfly 10.1.0.final
* Apache Maven 3.3.9 (o superior)
  + Se recomienda no usar la versión incluida en Netbeans, sino configurar el IDE para que utilice la versión externa de Maven
* Base de Datos PostgreSQL 9.3(o superior)
  + Se recomienda utilizar como cliente de base de datos pgAdmin 3 o superior
* PostgreSQL JDBC Driver 9.4.1212 (o superior)
* MySQL Java Connector 5.1.40 (o superior)
* Git

Repositorios

Solicitar acceso en el servidor Git para los siguientes repositorios:

* gestionAplicaciones
* gestionTerritorial
* srvAfip
* gestionPersonas
* gestionEfluentesLiquidos

Configuración de la Base de datos

Una vez instalado PostgreSQL, es necesario crear en el servidor local las siguientes bases de datos:

* gestionAplicaciones
* gestionTerritorial
* gestionPersonas
* gestionEfluentesLiquidos

Es importante respetar los nombres listados anteriormente para facilitar las configuraciones posteriores y tener los diferentes ambientes locales lo más parecido posible entre sí.

Una vez creadas las bases, hay que completar las mismas con la estructura de tablas, las constraints y los datos para poder luego ejecutar las aplicaciones. Para ello existen varias posibilidades. Si se tiene acceso a las bases de datos de los ambientes productivo, pre-productivo o desarrollo es posible realizar un backup completo de las bases y restaurarlo en el ambiente local que se acaba de crear. También es posible realizar un scripts con las tablas y las constraints, y por otro lado un backup de los datos.

Sería conveniente tener los scripts actualizados directamente disponibles en el repositorio.

Configuración de Wildfly

Descomprimir el .zip en algún directorio, preferentemente en la raíz del disco C en Windows.

Configurar una variable de entorno JBOSS\_HOME apuntando a la carpeta en que la que se descomprimió Wildfly (ej: C:\wildfly-10.1.0.Final). Luego agregar a la variable PATH lo siguiente: %JBOSS\_HOME%\bin

Luego es necesario crear un usuario de Wildlfy para poder acceder a la consola de administración.

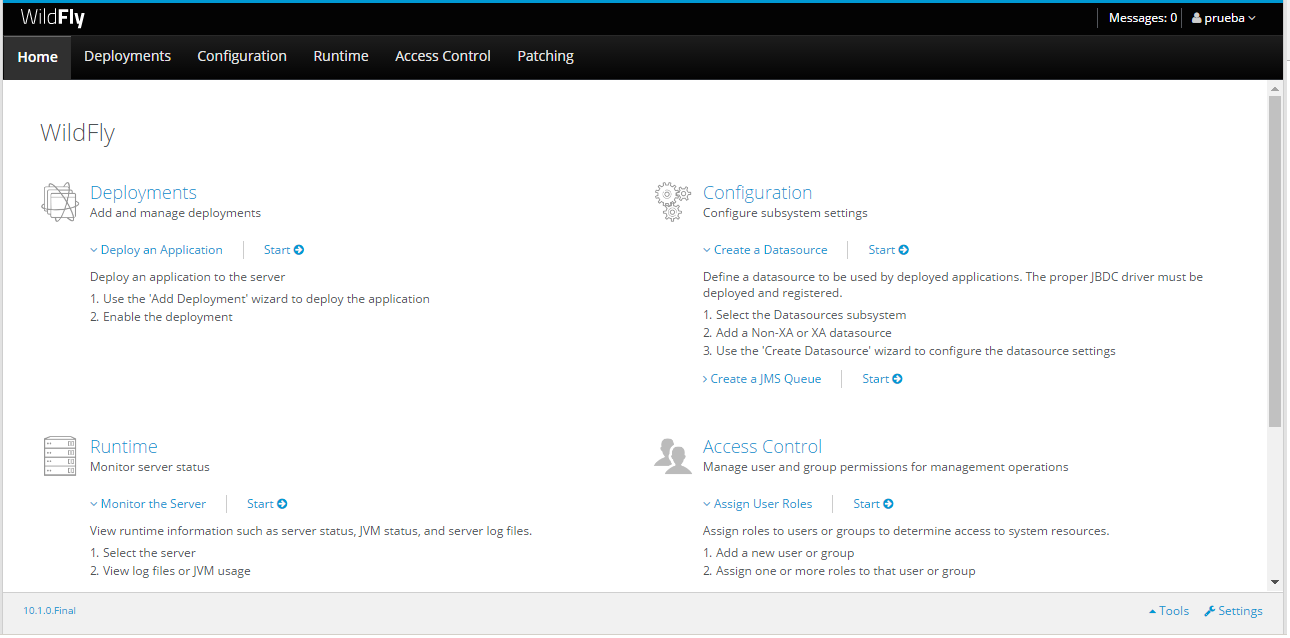
Para ello ejecutamos por línea de comandos el siguiente archivo: JBOSS\_HOME\bin\add-user.bat (o add-user.sh en Linux) y seguimos los siguientes pasos:

* Seleccionamos la opción a) “Management User”
* Elegimos un nombre de usuario. (Se sugiere usar el mismo del dominio)
* Elegimos la contraseña.
* Repetimos la contraseña.
* Cuando nos consulta por los grupos a los que queramos que pertenezca el usuario, simplemente presionamos enter.
* Nos pregunta si queremos agregar el usuario a “ManagementRealm”, respondemos que sí
* A la última pregunta contestamos también que sí.

Para comprobar que el usuario fue creado correctamente, probamos levantar el servidor e ingresar a la consola de administración:

* Por línea de comandos ejecutar el siguiente archivo JBOSS\_HOME\bin\standalone.bat
* Una vez que el servidor levante correctamente, ingresamos por cualquier navegador a la siguiente dirección: <http://localhost:9990>.
* Ingresamos el usuario y contraseña que habíamos creado previamente.

Si funcionó correctamente deberíamos ver una pantalla como la siguiente:



Configuración Maven

Bajar la versión binaria de maven en .zip y descomprimirla en algún directorio, preferentemente en la raíz del disco C en Windows.

Configurar la variable de entorno M2\_HOME apuntando al directorio creado anteriormente. Luego agregar a la variable PATH lo siguiente: %M2\_HOME%\bin.

Para verificar que Maven esté correctamente instalado, ejecutar por línea de comando lo siguiente: mvn –version.

Es posible que esto falle si no está configurada la variable JAVA\_HOME. Configurarla apuntando al directorio del JDK que se va a utilizar.

Para poder utilizar esta versión de Maven en Netbeans seguir los siguientes pasos:

* Ir a Tools->Options
* Ir a la pestaña Maven
* En el ítem “Maven Home” agregar el path de la instalación que hicimos anteriormente

Configuración de Drivers para Bases de Datos

Hay varias formas de configurar los drivers en Wildfly, pero dependiendo de cada driver funcionan mejor ciertas formas que otras.

PostgreSQL JDBC Driver

Este driver conviene agregarlo a través de la consola de administración de Wildlfy, de la siguiente forma:

* Ingresar a <http://localhost:9990>
* Ir a “Deployments”
* Hacer click en “add”
* Seleccionar la opción “Upload a new deployment” y hacer click en “Next”
* Hacer click en “Seleccionar archivo”
* Buscar el archivo con el .jar de postgreSQL en nuestro disco local.
* Dejar los nombres que figuran por defecto y hacer click en “finish”

MySQL Java Connector

Este driver es necesario para que funcioné la conexión a la base de datos de SIMEL que utiliza el proyecto srvAfip. Seguir los siguientes pasos:

* Desde la línea de comandos situarse en la carpeta JBOSS\_HOME\bin
* Ejecutar jboss-cli.bat –connect
* Ejecutar:
  + module add --name=system.layers.base.com.mysql.driver  --dependencies=javax.api,javax.transaction.api --resources=/mysql-connector-java-5.1.33.jar
  + (Modificar el path y el nombre del archive según corresponda)
* Ejecutar:
  + :reload
* Ejecutar:
  + /subsystem=datasources/jdbc-driver=mysql/:add(driver-module-name=system.layers.base.com.mysql.driver,driver-name=mysql,jdbc-compliant=false,driver-class-name=com.mysql.jdbc.Driver

Configuración de los Datasources

Para que funcionen las aplicaciones de efluentes es necesario configurar los respectivos datasources. Existen dos formas de realizar esto: por la consola de administración de Wildfly o modificando el archivo JBOSS\_HOME\standalone\configuration\standalone.xml

Si modificamos el archivo standalone.xml:

* Nos ubicamos en server->profile->subsystem->datasources
* Allí agregamos un nuevo datasource con la siguiente estructura:

<datasource jndi-name=**"java:jboss/datasources/NOMBRE\_APP"** pool-name=**"NOMBRE\_APP"** enabled=**"true"** use-java-context=**"true"**>

<connection-url>**jdbc:postgresql://localhost:5432/NOMBRE\_APP**</connection-url>

<driver>**DRIVER**</driver>

<security>

<user-name>**USER**</user-name>

<password>**PASS**</password>

</security>

</datasource>

Reemplazando:

* NOMBRE\_APP por el nombre de la aplicación para la que se crea el datasource
* DRIVER por el nombre con que se deployó el driver
* USER: usuario de la base de datos
* PASS: contraseña de la base de datos

Ejemplo:

<datasource jndi-name=**"java:jboss/datasources/gestionAplicaciones"** pool-name=**"gestionAplicaciones"** enabled=**"true"** use-java-context=**"true"**>

<connection-url>**jdbc:postgresql://localhost:5432/gestionAplicaciones**</connection-url>

<driver>**postgresql-9.4.1212.jar**</driver>

<security>

<user-name>**postgres**</user-name>

<password>**postgres**</password>

</security>

</datasource>

Configurar Aplicaciones

Es importante configurar las aplicaciones en el orden que menciona a continuación, ya que de lo contrario no funcionaran. Es necesario bajarse todos los cambios de los repositorios mencionados anteriormente y situar el repositorio local en la rama “develop”. Una vez hecho esto, abrir los proyectos con Netbeans y configurarlos uno por uno como se indica a continuación.

Configuración General de Aplicaciones

Copiar los archivos que se mencionan a continuación a la misma carpeta en que se encuentran, renombrándolos:

* APLICACION\src\main\resources\Config.properties.dist
  + copiarlo a la misma carpeta renombrándolo como “Config.properties”
* APLICACION \src\main\resources\META-INF\persistence.xml.dist
  + copiarlo a la misma carpeta renombrándolo como “persistence.xml”

(Aclaración: en algunos proyecto podrían no estar los archivos mencionados anteriormente si no son necesarios)

Una vez hecho esto ejecutar en Netbeans seguir los siguientes pasos:

* Click derecho sobre el proyecto y “clean”
* Si todo ok, click derecho y “Build with dependencies”
* Si todo ok, click derecho y “run”

Si el build se realizó correctamente y el servidor levantó el proyecto, se debería abrir una ventana con el login, o si el usuario ya se encuentra logueado se debería mostrar la pantalla principal del proyecto levantado.

A continuación tenemos una descripción breve de cada proyecto y algunas de sus características principales.

gestionAplicaciones

Esta aplicación tiene por objeto principal gestionar las aplicaciones a las cuales tienen acceso los usuarios. Para realizarlo implementa un logueo único contra el directorio activo del dominio mediante el protocolo LDAP.

Distingue dos roles de usuarios: el usuario común que solo puede validar sus credenciales y acceder al listado con todas las aplicaciones que tenga disponibles, y el administrador, que además de tener las funcionalidades del usuario común, también podrá gestionar las aplicaciones existentes con la correspondiente gestión de usuarios y agregar nuevas.

* No consume de otros web services
* Expone el siguiente web service:
  + AccesoAppWebService

gestionTerritorial

Esta aplicación gestiona la información correspondiente a la organización territorial de nuestro país. Contiene una API de servicios web para ser consumidos tanto por aplicaciones internas como externas, de manera de proveer un registro único de Provincias, departamentos, localidades, municipios y regiones temáticas.

La acreditación de esta aplicación es gestionada mediante el servicio brindado por gestionAplicaciones.

Distingue dos roles de usuarios, el administrador que gestiona los contenidos y el seguridad, que gestiona usuarios y roles. Para asignar usuarios a la aplicación, previamente deben ser vinculados mediante gestionAplicaciones.

* Consume del siguiente web service:
  + AccesoAppWebService
* Expone el siguiente web service:
  + CentrosPobladosWebService

srvAfip

El presente servicio tiene por objeto brindar información sobre los establecimientos registrados en el Sistema de Manifiestos en línea (SIMEL). Dado que su base de datos contiene una tabla actualizada diariamente por AFIP en la cual consigna el vínculo entre CUIT y razón social o persona titular, se utiliza para validar estos datos con cualquier persona que posea CUIT. Asimismo, el servicio contiene otro método para obtener el expediente de tramitación de un establecimiento registrado en el SIMEL, principalmente para el registro de Declaraciones Juradas del Sistema de control de vertido de efluentes líquidos.

* No consume de otros web services
* Expone el siguiente web service:
  + CuitAfipWs

gestionPersonas

Esta aplicación web administra el servicio de gestión de personas que tiene como objetivo principal la gestión de la información relativa a las personas, tanto físicas como jurídicas, que interactúen con las diferentes áreas del MAyDS mediante sus respectivas aplicaciones.

Como las restantes aplicaciones de administración de servicios, esta tendrá dos perfiles de usuario: Administrador y Seguridad, este último tendrá la capacidad de gestionar usuarios y roles para la aplicación más allá de todo su contenido. El perfil de seguridad será el de los usuarios del sector Desarrollo de Aplicaciones y el de Administrador, de los usuarios designados por los dueños de las aplicaciones clientes, para tal fin.

* Consume de los siguientes web services:
  + AccesoAppWebService
  + CentrosPobladosWebService
  + CuitAfipWs
* Expone el siguiente web service
  + PersonasWebService

gestionEfluentesLiquidos

Esta aplicación de tipo empresarial se compone de dos interfases de usuario: -webInt y webExt, un backend compartido por ambas -ejb y un módulo ejb consumido como servicio por -webInt: registroUnicoPersonas-ejb.

registroUnicoPersonas-ejb

* No consumo ni expone ningún web service

gestionEfluentesLiquidos-ejb

* No consume ni expone ningún web service

gestionEfluentesLiquidos-webInt

Es la interfaz para usuarios internos que reune la funcionalidad necesaria para la administración de la información relativa a la getión de Establecimientos vertedores de efluentes líquidos que realiza la Dirección de Prevención y Recomposición Ambiental (DPyRA). Hasta la actualidad, según lo acordado oportunamente solo incluye las funcionalidades necesarias para la gestión de la información necesaria para que los Establecimiento puedan registrar las declaraciones juradas anuales, mediante el uso de la otra interface. No obstante se preve a partir de la siguiente etapa completar las funcionalidades restantes que hacen a la operatoria diaria de la repartición.

* Consume de los siguientes web services:
  + AccesoAppWebService
  + CentrosPobladosWebService
* No expone ningún web service

gestionEfluentesLiquidos-webExt

Es la interfaz para usuarios externos que reúne la funcionalidad necesaria para el registro de las declaraciones juradas anuales, por parte de los Establecimientos administrados, cada uno de los cuales constituyen un usuario externo. Consta principalmente de dos grupos de funcionalidad, una pública que brinda información respecto de las normativas vigentes, guía de trámite, preguntas frecuentes, glosario y explica los procedimientos necesarios para la acreditación de usuario externos; y otra para usuarios registrados a los que se permite el registro de la declaración y la actualización de los datos geográficos de la ubicación de los Establecimientos.

* Consume del siguiente web service
  + CuitAfipWs

Configuración Java Mail Wildlfy

En el archivo standalone.xml:

1. En la etiqueta <subsystem xmlns="urn:jboss:domain:mail:2.0">

<mail-session name=**"ambiente"** debug=**"true"** jndi-name=**"java:/mail/ambiente"** from=**"[REMITENTE]@ambiente.gob.ar"**>

<smtp-server outbound-socket-binding-ref=**"efluentes"** username=**"medioambiente\servicemailtest"** password=**"Desarroll017"**/>

</mail-session>

2. En la etiqueta <socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">

<outbound-socket-binding name=**"efluentes"**>

<remote-destination host=**"casarray.medioambiente.gov.ar"** port=**"25"**/>

</outbound-socket-binding>

Problemas Detectados

Con el JDK 1.8, al momento de hacer un build tira el error

`Failed to read schema document 'xjc.xsd', because 'file' access is not allowed due to restriction set by the accessExternalSchema property.`

La solución consiste en crear el archivo `jaxp.properties` en `/path/to/jdk1.8.0/jre/lib` y escribir la siguiente regla:

javax.xml.accessExternalSchema = all

Solución basada en:

* <http://docs.oracle.com/javase/7/docs/api/javax/xml/XMLConstants.html#ACCESS_EXTERNAL_SCHEMA>
* <http://stackoverflow.com/questions/23011547/webservice-client-generation-error-with-jdk8>