Création du dépôt SVN :

svnadmin create /home/fenyo/svnrep

Installation de J2EE

Installation du jdk de java 7

Attention : tout en 64 bits ou rien en 64

Installation d’Eclipse : Eclipse pour JavaEE (ou Flash Builder)

Installation de m2e :

Installation de maven integration for eclipse WTP :

Installer jboss maven support : via jboss tools (indigo) du marketplace

Installer spring IDE sur marketplace

Au final : ce qui est installé via Eclipse marketplace doit être :

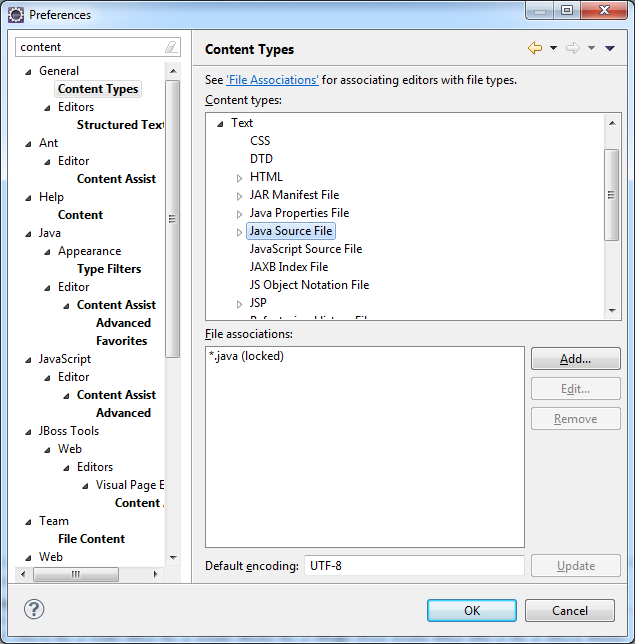
- Spring IDE

- Maven integration for eclipse (m2eclipse)

- Maven integration for eclipse WTP (m2eclipse-wtp)

- Jboss tools (indigo) : ça inclut notamment jboss maven support

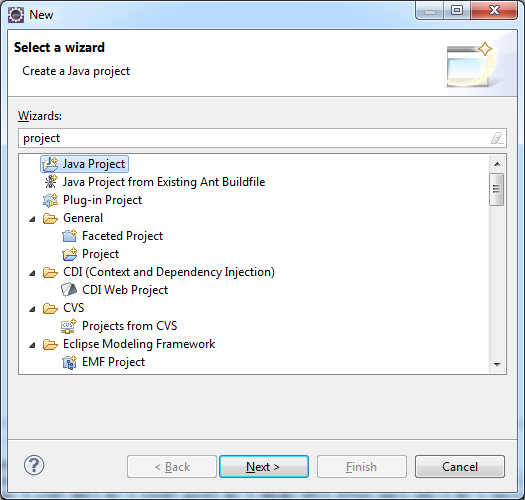
Dans Eclipse : Window / Préférences / Content types , Text / java source file, et remplir « default encoding » avec UTF-8 puis faire update :

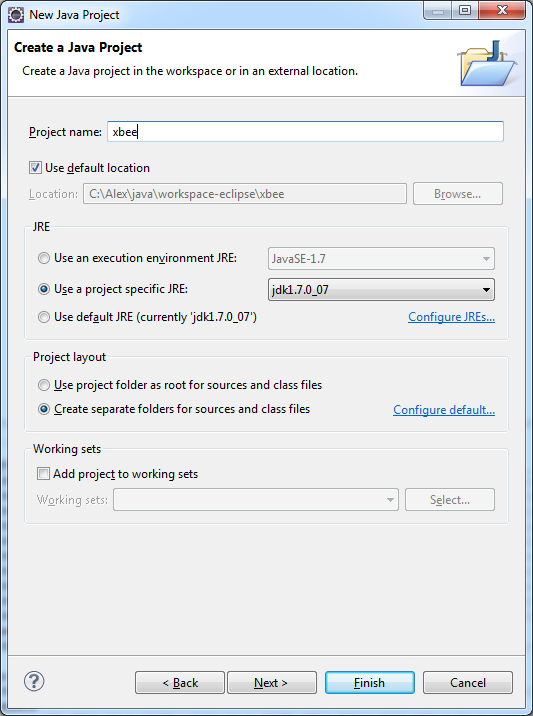


Rajouter le module SVN : Fenêtre / Eclipse Market Place et choisir Subversive et non Subclipse (laisser la sélection complète par défaut, et choisir la plus récente librairie SSH et SVN en Java – ne pas choisir la version qui fait un pont avec l’OS)

Principe général : utiliser les facets plutôt que « convertir … », pour ajouter des fonctionnalités

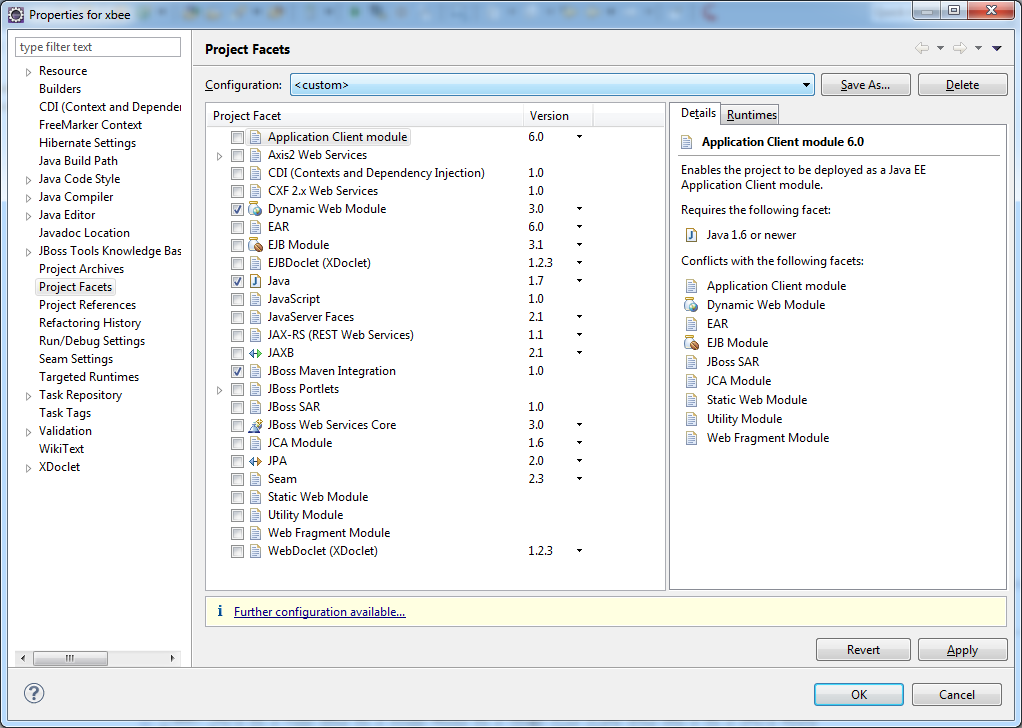
Créer un projet le plus simple possible :



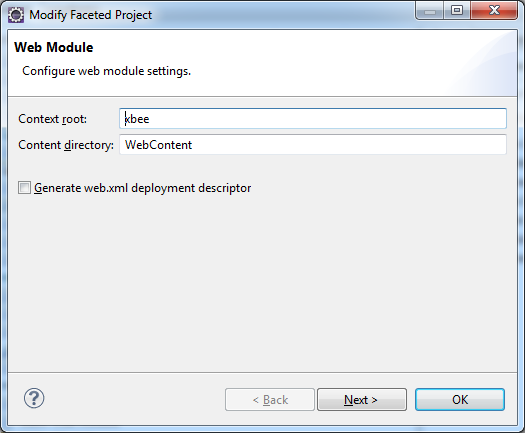


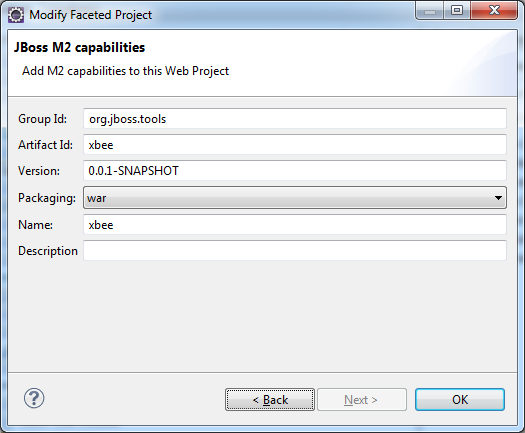
Convertif en facets

Choisir :

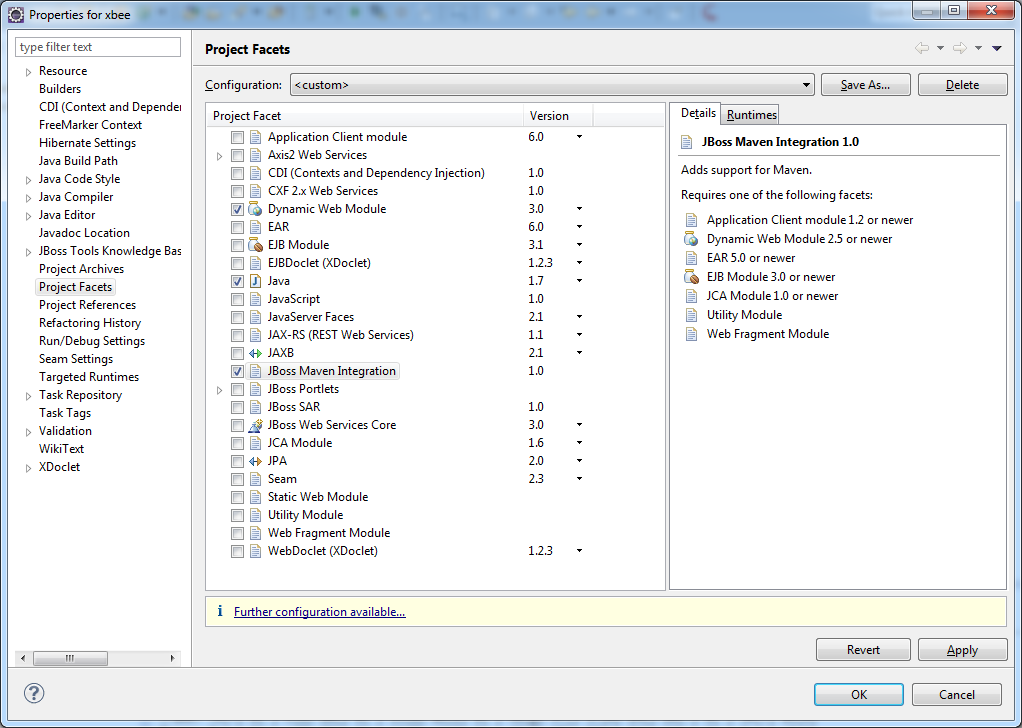


Cliquer sur « Further configuration available » après avoir sélectionné JBoss Maven Integration :





On a sélectionné les défauts (donc rien changé)



Cliquer sur OK

Cela a créé un POM minimaliste.

Faire runas / maven install

Ca doit faire « build success »

Rajouter dans le pom.xml  les dépendances à spring en mettant la dernière release qu’on veut dans la propriété suivante :

<!-- Shared version number properties -->

<properties>

<org.springframework.version>3.0.5.RELEASE</org.springframework.version>

</properties>

<dependencies>

<!--

Core utilities used by other modules.

Define this if you use Spring Utility APIs (org.springframework.core.\*/org.springframework.util.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Expression Language (depends on spring-core)

Define this if you use Spring Expression APIs (org.springframework.expression.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Bean Factory and JavaBeans utilities (depends on spring-core)

Define this if you use Spring Bean APIs (org.springframework.beans.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Aspect Oriented Programming (AOP) Framework (depends on spring-core, spring-beans)

Define this if you use Spring AOP APIs (org.springframework.aop.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Application Context (depends on spring-core, spring-expression, spring-aop, spring-beans)

This is the central artifact for Spring's Dependency Injection Container and is generally always defined

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Various Application Context utilities, including EhCache, JavaMail, Quartz, and Freemarker integration

Define this if you need any of these integrations

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Transaction Management Abstraction (depends on spring-core, spring-beans, spring-aop, spring-context)

Define this if you use Spring Transactions or DAO Exception Hierarchy

(org.springframework.transaction.\*/org.springframework.dao.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

JDBC Data Access Library (depends on spring-core, spring-beans, spring-context, spring-tx)

Define this if you use Spring's JdbcTemplate API (org.springframework.jdbc.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Object-to-Relation-Mapping (ORM) integration with Hibernate, JPA, and iBatis.

(depends on spring-core, spring-beans, spring-context, spring-tx)

Define this if you need ORM (org.springframework.orm.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Object-to-XML Mapping (OXM) abstraction and integration with JAXB, JiBX, Castor, XStream, and XML Beans.

(depends on spring-core, spring-beans, spring-context)

Define this if you need OXM (org.springframework.oxm.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Web application development utilities applicable to both Servlet and Portlet Environments

(depends on spring-core, spring-beans, spring-context)

Define this if you use Spring MVC, or wish to use Struts, JSF, or another web framework with Spring (org.springframework.web.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Spring MVC for Servlet Environments (depends on spring-core, spring-beans, spring-context, spring-web)

Define this if you use Spring MVC with a Servlet Container such as Apache Tomcat (org.springframework.web.servlet.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Spring MVC for Portlet Environments (depends on spring-core, spring-beans, spring-context, spring-web)

Define this if you use Spring MVC with a Portlet Container (org.springframework.web.portlet.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc-portlet</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Support for testing Spring applications with tools such as JUnit and TestNG

This artifact is generally always defined with a 'test' scope for the integration testing framework and unit testing stubs

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${org.springframework.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

Refaire run as / maven install

Créer WEB-INF/spring-business-context.xml :

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:jee=*"http://www.springframework.org/schema/jee"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd*

*http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc.xsd"*>

</beans>

Créer le web.xml minimal :

<web-app version=*"2.5"*

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<display-name>XBee web module</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-business-context.xml</param-value>

</context-param>

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>/WEB-INF/log4j.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<servlet>

<servlet-name>spring-mvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring-mvc</servlet-name>

<url-pattern>/app/\*</url-pattern>

</servlet-mapping>

</web-app>

Créer le log4j.xml minimal :

<?xml version=*"1.0"* encoding=*"ISO-8859-15"* ?>

<!DOCTYPE log4j:configuration SYSTEM "http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/doc-files/log4j.dtd">

<!-- build/src/jakarta-log4j-1.2.8/src/java/org/apache/log4j/xml/examples -->

<log4j:configuration xmlns:log4j=*"http://jakarta.apache.org/log4j/"*>

<appender name=*"consoleAppender"* class=*"org.apache.log4j.ConsoleAppender"*>

<layout class=*"org.apache.log4j.PatternLayout"*>

<!-- for debugging -->

<param name=*"ConversionPattern"* value=*"%d [%t] %p - %F:%L - %m%n"*/>

<!-- for fast debugging : no line number -->

<!-- <param name="ConversionPattern" value="%d [%t] %p - %F - %m%n"/> -->

</layout>

</appender>

<category name=*"org.springframework.scheduling"*>

<priority value=*"debug"*/>

<appender-ref ref=*"consoleAppender"*/>

</category>

<category name=*"net.fenyo"*>

<priority value=*"debug"*/>

<appender-ref ref=*"consoleAppender"*/>

</category>

<root>

<!-- <priority value="trace"/> -->

<!-- <priority value="debug"/> -->

<!-- <priority value="error"/> -->

<priority value=*"error"*/>

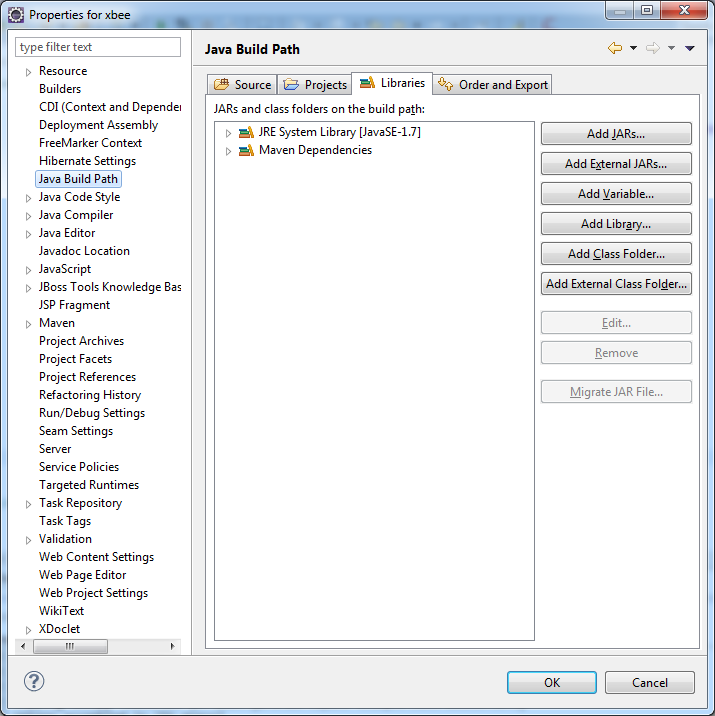
<!-- avec cet appender-ref, on a 2 fois les messages pour net.fenyo, s'il y a aussi le même consoleAppender pour net.fenyo -->

<!-- <appender-ref ref="consoleAppender"/> -->

</root>

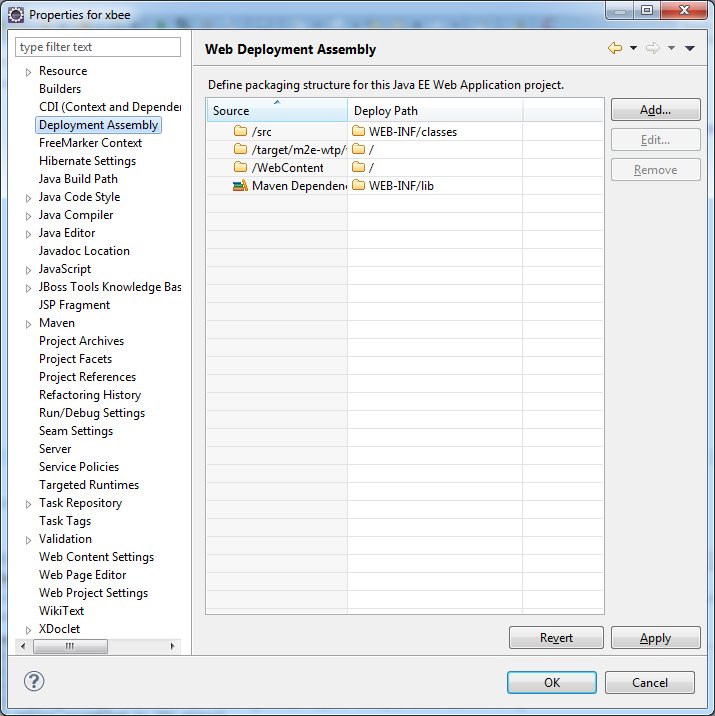
</log4j:configuration>

Ca compile mais ca publierait en erreur car si on publiait sur un serveur, car les librairies Spring dont maven a détecté le besoin sont bien dans le java build path :

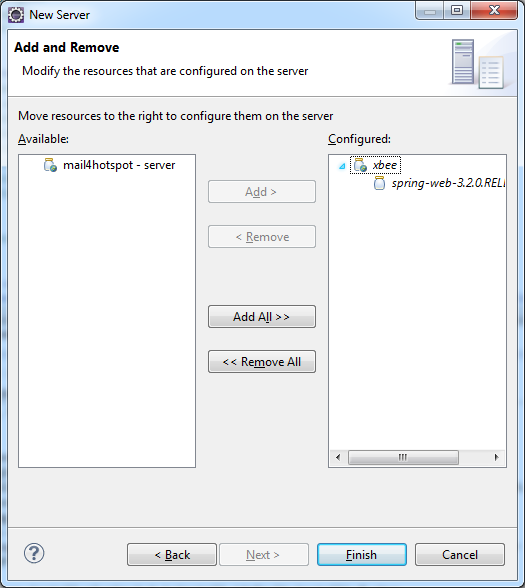


Mais elles ne sont pas forcément dans le déployment assembly, qui est ce qui est exporté vers le serveur quand on déploie l’appli :

Ex où elles les sont :

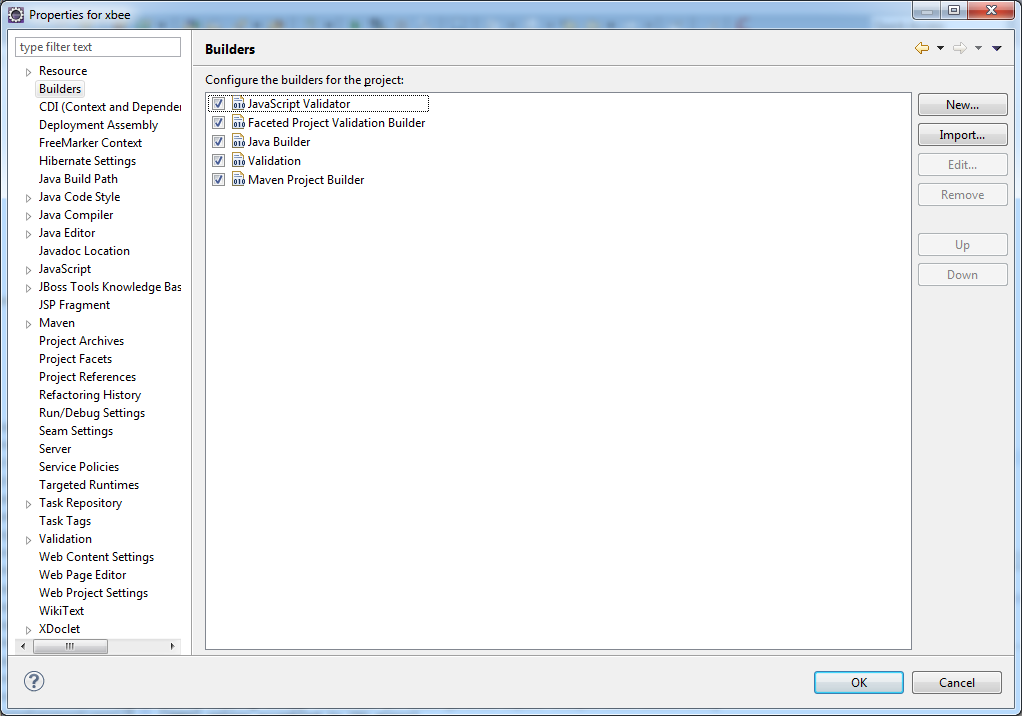


Quand ca publie en erreur, au lieu d’avoir spring-web-… dans l’écran suivant, on n’a que xbee pour notre appli :

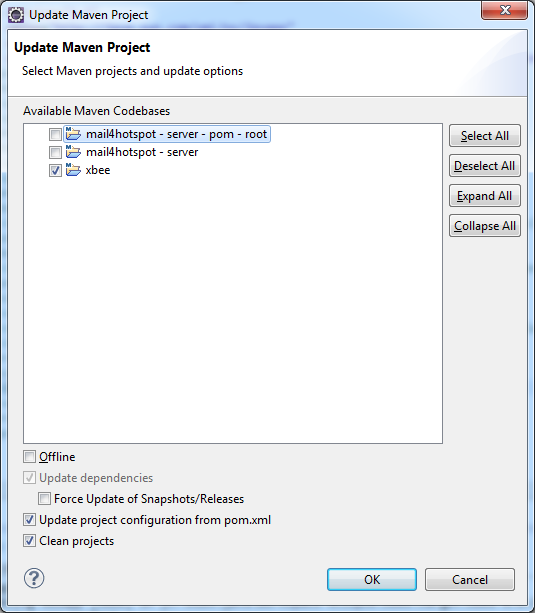


Si on n’avait pas Maven Dependencies, il faudrait :

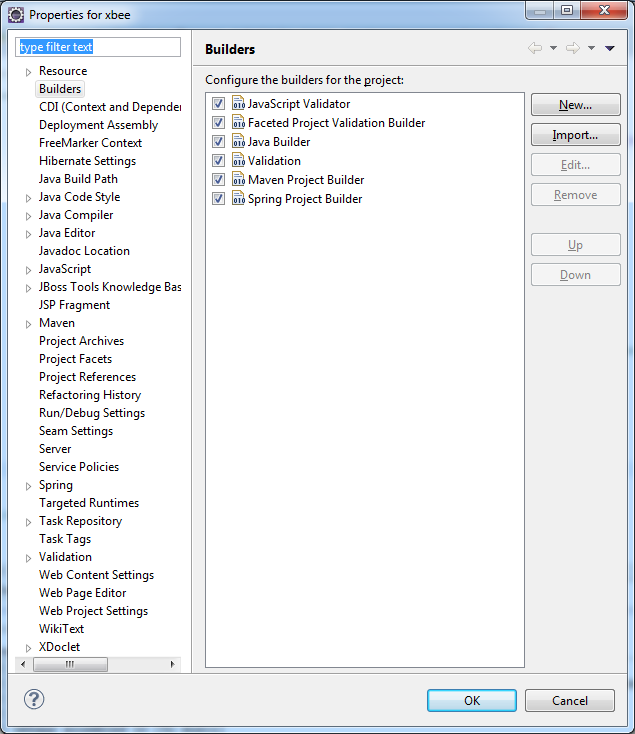
* Soit le rajouter à la main via Add dans l’écran précédent
* Soit faire update project dans le menu maven, pour que le plugin eclipse jboss maven integration (chargé par facets) voit qu’il y a des dépendances et rajoute ce qu’il faut dans deployment assembly
* Ce plugin rajoutera aussi le spring project builder qui n’y est pas déjà : actuellement (sans ce builder) :



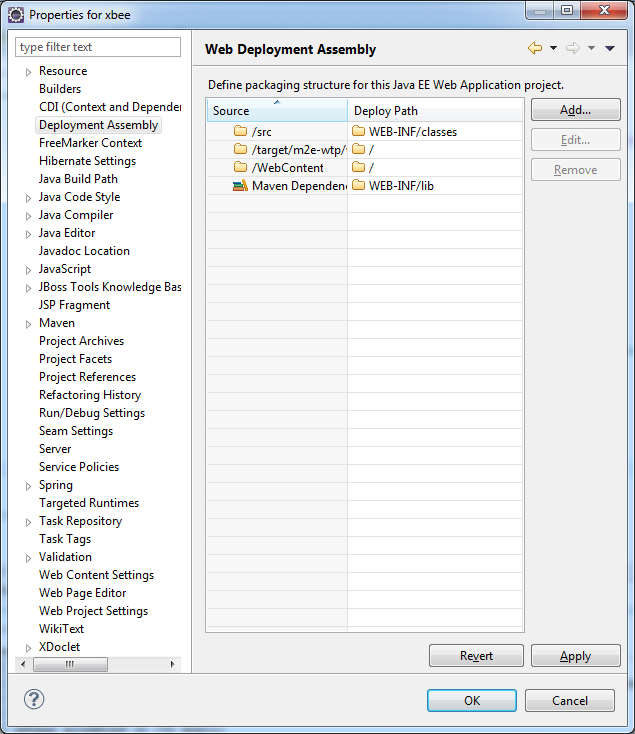
On fait donc update maven project :



Ca a rajouté spring project builder :



Et ca a aussi rajouté les dépendances maven dans le deployment assembly si elles n’y étaient pas déjà :



Quand on fait maven install, on a un message de warning au début :

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

[INFO] Scanning for projects...

Il semble que ce soit un bug du maven embarqué dans m2e :

http://stackoverflow.com/questions/11916706/slf4j-failed-to-load-class-org-slf4j-impl-staticloggerbinder-error

On en profite pour rajouter les dépendances avec des outils communs :

Dans properties :

Pour servlet-api, il ne faut pas la déployer car elle est déjà fournie par le serveur, mais il faut l’avoir pour compiler, donc scope provided :

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

Important : à chaque fois qu’on modifie le POM : faire maven / update project

Faire maven / download source et maven / download javadoc

On a donc rajouté :

<properties>

<org.springframework.version>3.2.0.RELEASE</org.springframework.version>

<version.log4j>1.2.14</version.log4j>

<version.commons.logging>1.1.1</version.commons.logging>

<version.standard.taglibs>1.1.2</version.standard.taglibs>

<version.commons.lang>3.1</version.commons.lang>

<version.jstl>1.2</version.jstl>

</properties>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${version.log4j}</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>${version.commons.logging}</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>${version.standard.taglibs}</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>${version.commons.lang}</version>

</dependency>

<!-- version à vérifier -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${version.jstl}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.6</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.3</version>

</dependency>

Le pom est maintenant :

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>org.jboss.tools</groupId>

<artifactId>xbee</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>xbee</name>

<build>

<sourceDirectory>${basedir}/src</sourceDirectory>

<resources>

<resource>

<directory>${basedir}/src</directory>

<excludes>

<exclude>\*\*/\*.java</exclude>

</excludes>

</resource>

</resources>

<plugins>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.2</version>

<configuration>

<warSourceDirectory>${basedir}/WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

<!-- Shared version number properties -->

<properties>

<org.springframework.version>3.2.0.RELEASE</org.springframework.version>

<version.log4j>1.2.14</version.log4j>

<version.commons.logging>1.1.1</version.commons.logging>

<version.standard.taglibs>1.1.2</version.standard.taglibs>

<version.commons.lang>3.1</version.commons.lang>

<version.jstl>1.2</version.jstl>

</properties>

<dependencies>

<!--

Core utilities used by other modules.

Define this if you use Spring Utility APIs (org.springframework.core.\*/org.springframework.util.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Expression Language (depends on spring-core)

Define this if you use Spring Expression APIs (org.springframework.expression.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Bean Factory and JavaBeans utilities (depends on spring-core)

Define this if you use Spring Bean APIs (org.springframework.beans.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Aspect Oriented Programming (AOP) Framework (depends on spring-core, spring-beans)

Define this if you use Spring AOP APIs (org.springframework.aop.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Application Context (depends on spring-core, spring-expression, spring-aop, spring-beans)

This is the central artifact for Spring's Dependency Injection Container and is generally always defined

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Various Application Context utilities, including EhCache, JavaMail, Quartz, and Freemarker integration

Define this if you need any of these integrations

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Transaction Management Abstraction (depends on spring-core, spring-beans, spring-aop, spring-context)

Define this if you use Spring Transactions or DAO Exception Hierarchy

(org.springframework.transaction.\*/org.springframework.dao.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

JDBC Data Access Library (depends on spring-core, spring-beans, spring-context, spring-tx)

Define this if you use Spring's JdbcTemplate API (org.springframework.jdbc.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Object-to-Relation-Mapping (ORM) integration with Hibernate, JPA, and iBatis.

(depends on spring-core, spring-beans, spring-context, spring-tx)

Define this if you need ORM (org.springframework.orm.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Object-to-XML Mapping (OXM) abstraction and integration with JAXB, JiBX, Castor, XStream, and XML Beans.

(depends on spring-core, spring-beans, spring-context)

Define this if you need OXM (org.springframework.oxm.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-oxm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Web application development utilities applicable to both Servlet and Portlet Environments

(depends on spring-core, spring-beans, spring-context)

Define this if you use Spring MVC, or wish to use Struts, JSF, or another web framework with Spring (org.springframework.web.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Spring MVC for Servlet Environments (depends on spring-core, spring-beans, spring-context, spring-web)

Define this if you use Spring MVC with a Servlet Container such as Apache Tomcat (org.springframework.web.servlet.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Spring MVC for Portlet Environments (depends on spring-core, spring-beans, spring-context, spring-web)

Define this if you use Spring MVC with a Portlet Container (org.springframework.web.portlet.\*)

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc-portlet</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<!--

Support for testing Spring applications with tools such as JUnit and TestNG

This artifact is generally always defined with a 'test' scope for the integration testing framework and unit testing stubs

-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${org.springframework.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>${version.log4j}</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>${version.commons.logging}</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>${version.standard.taglibs}</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>${version.commons.lang}</version>

</dependency>

<!-- version à vérifier -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${version.jstl}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

<version>1.6</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.3</version>

</dependency>

</dependencies>

</project>

Créer un fichier vide de properties META-INF/config.properties pour mettre les éléments de configs

Créer un fichier basique pour la config de la servlet spring-mvc :

WEB-INF/spring-mvc-servlet.xml :

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:task=*"http://www.springframework.org/schema/task"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.1.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.1.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd*

*http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task-3.1.xsd"*>

<!-- post-processors for all standard config annotations -->

<context:component-scan base-package=*"net.fenyo.xbee"*/>

<mvc:annotation-driven/>

<context:annotation-config/>

<task:annotation-driven/>

<bean id=*"viewResolver"* class=*"org.springframework.web.servlet.view.UrlBasedViewResolver"*>

<property name=*"viewClass"* value=*"org.springframework.web.servlet.view.JstlView"*/>

<property name=*"prefix"* value=*"/WEB-INF/views/"*/>

<property name=*"suffix"* value=*".jsp"*/>

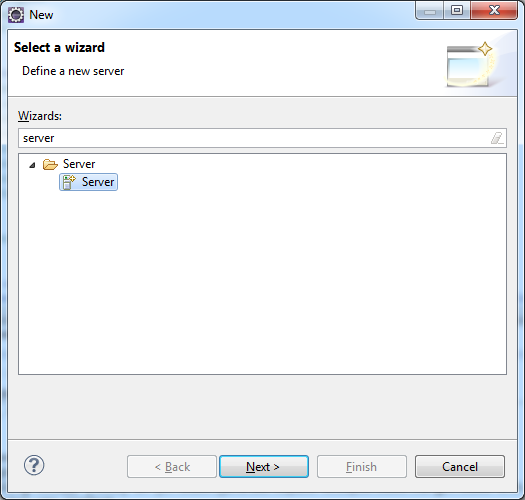
</bean>

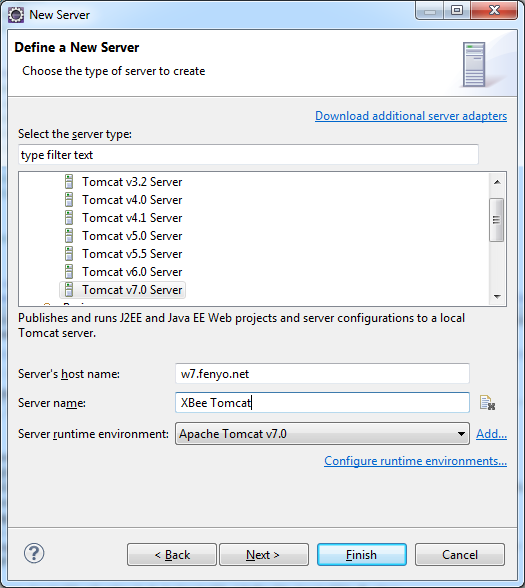
<!-- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/beans.html -->

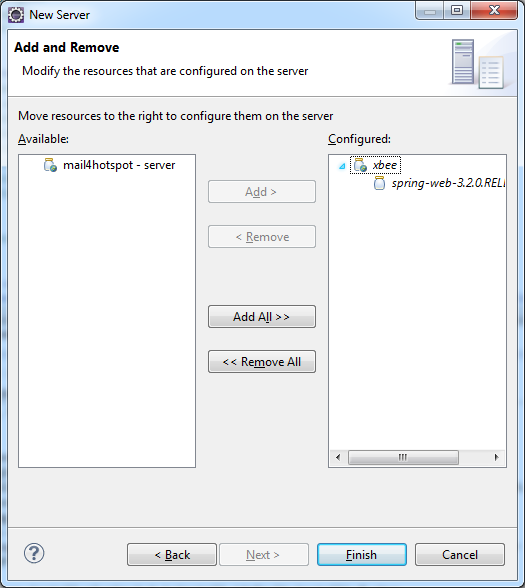
<context:property-placeholder location=*"META-INF/config.properties"*/>

</beans>

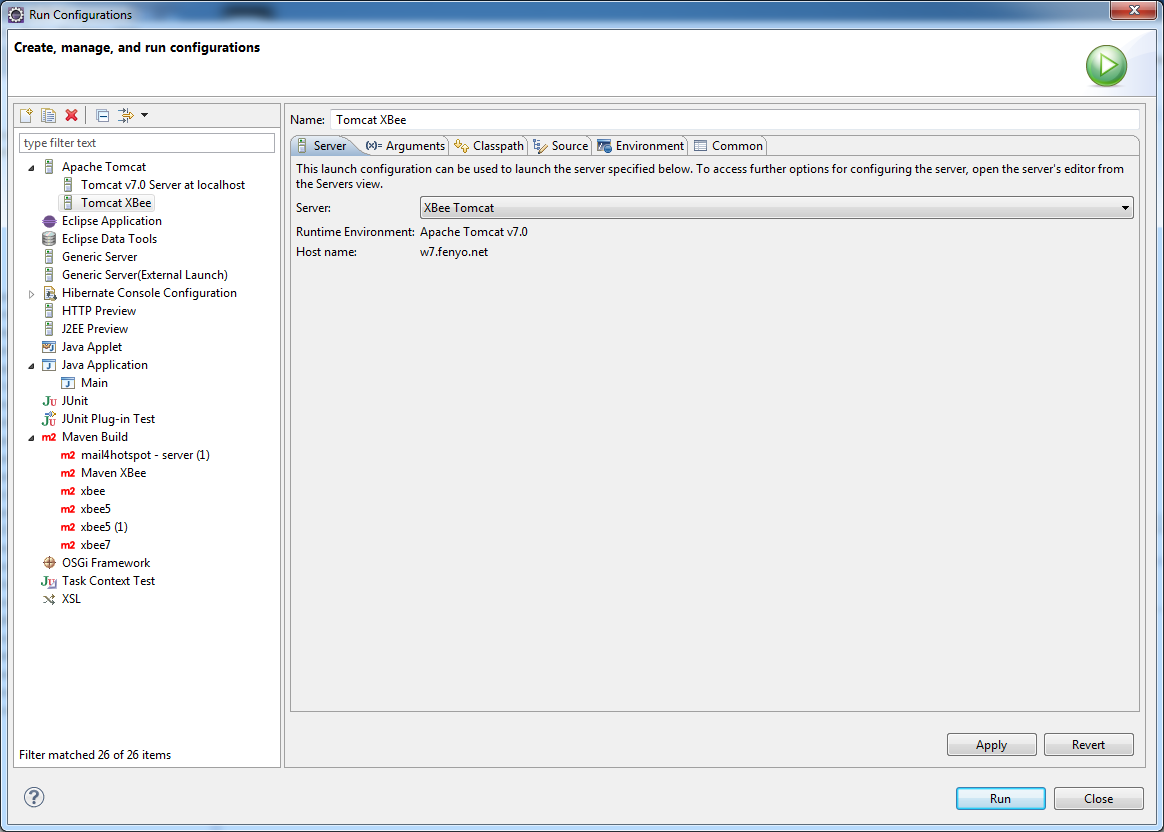
Créer un nouveau serveur :



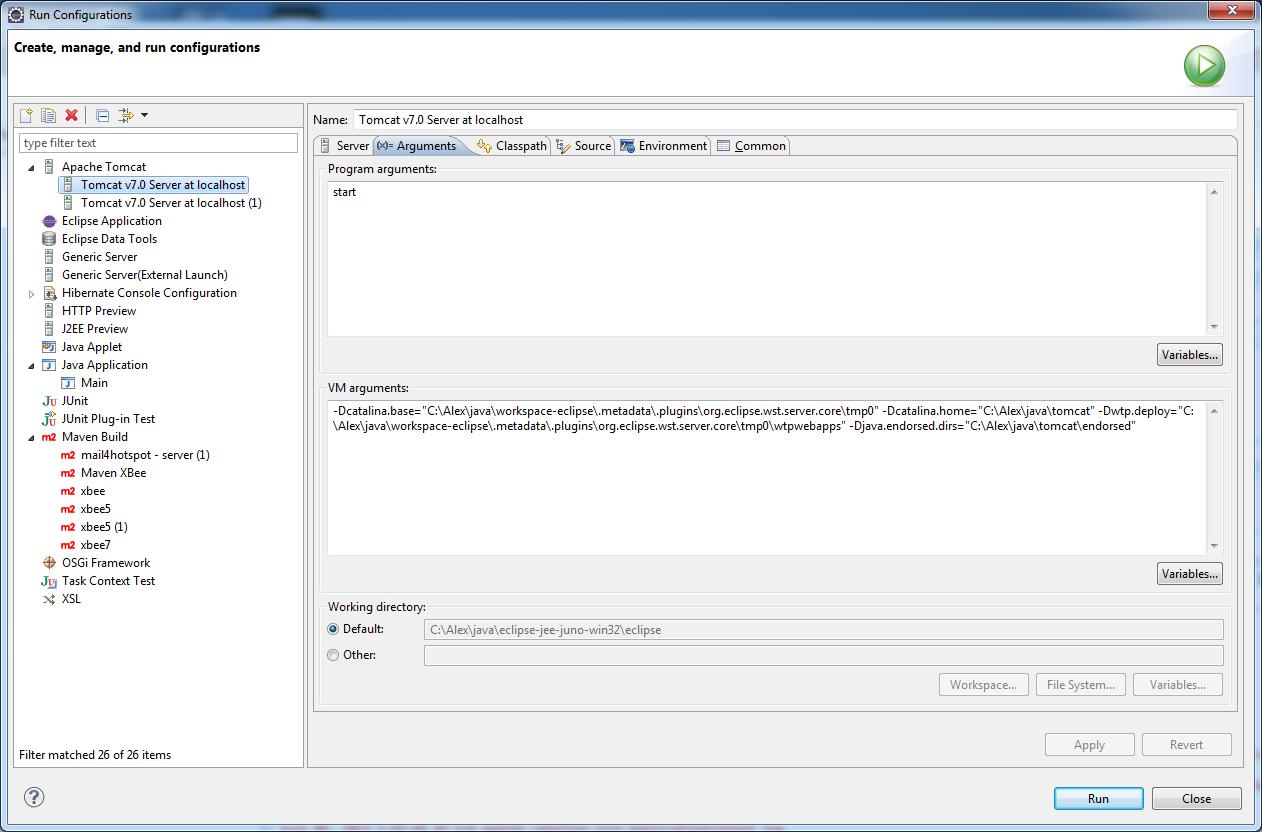
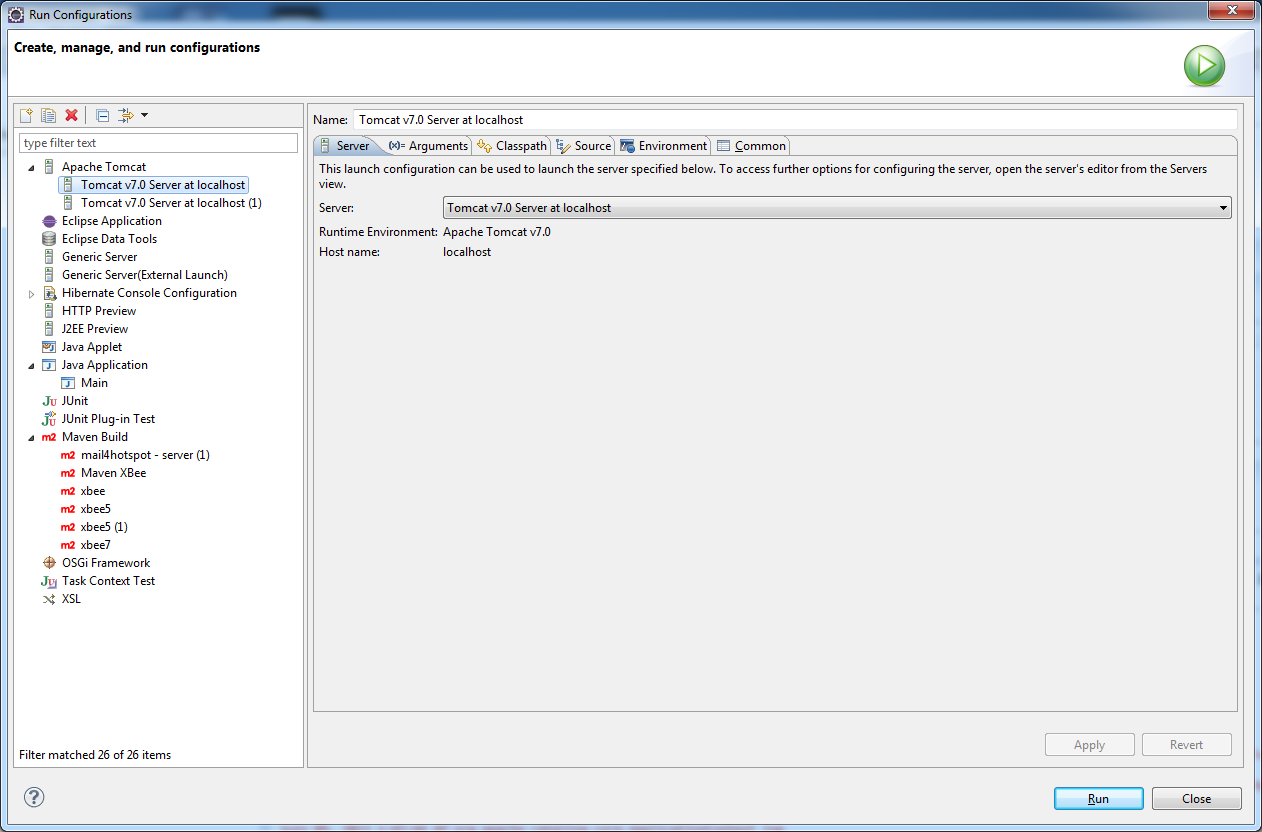


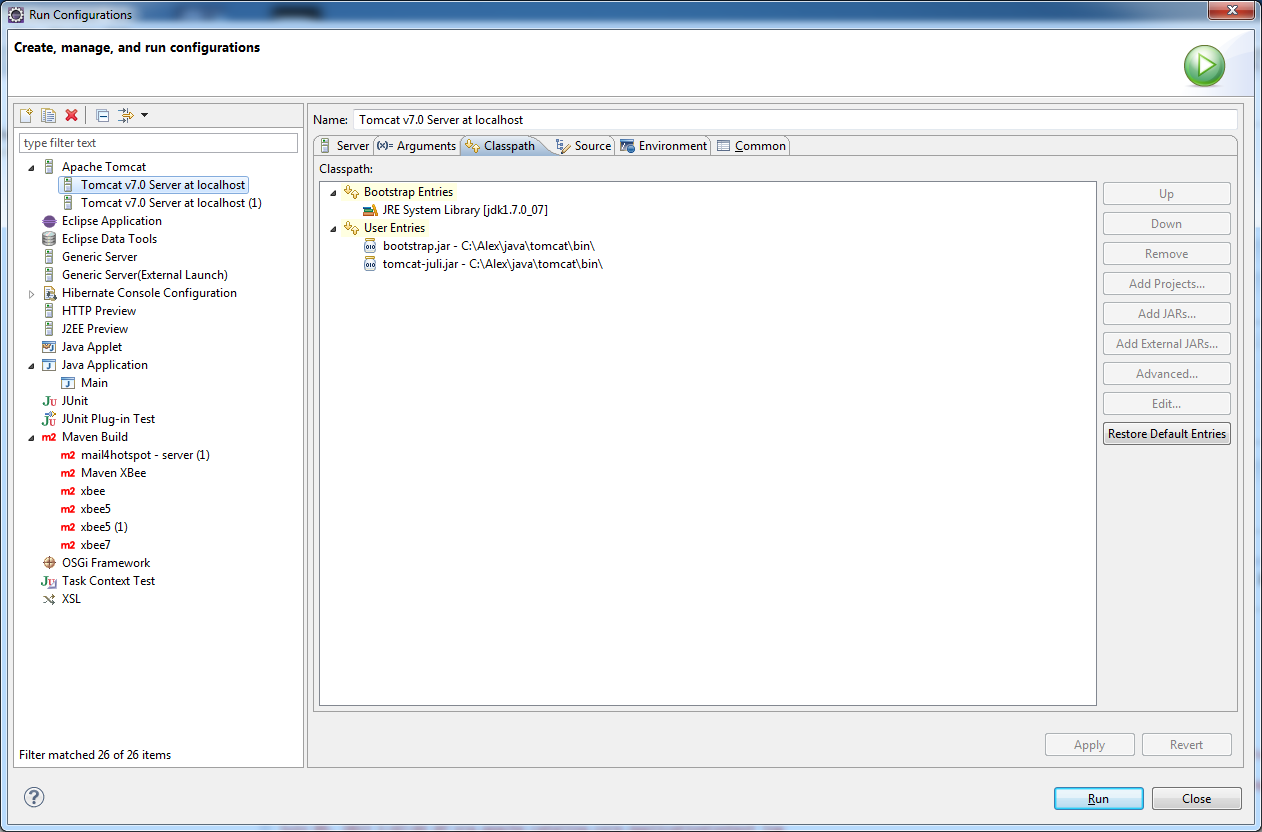
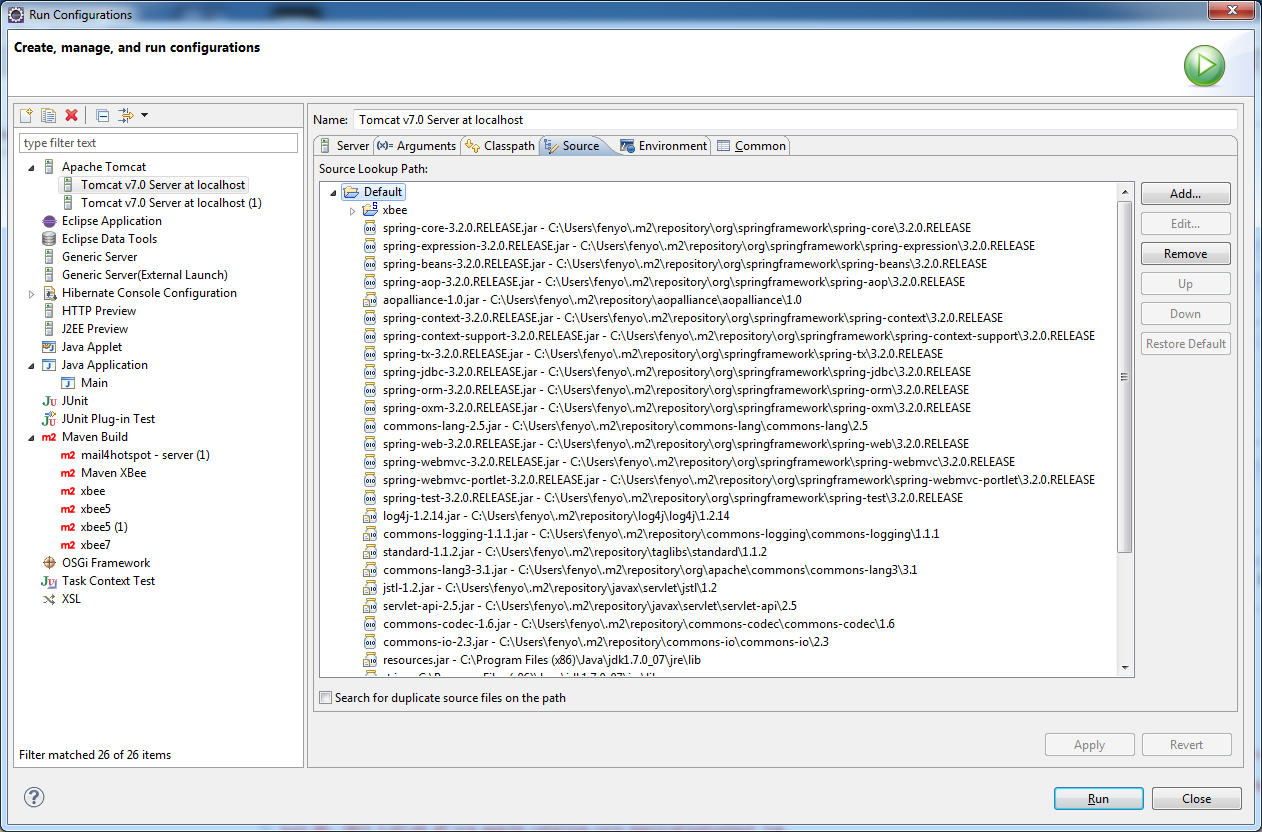
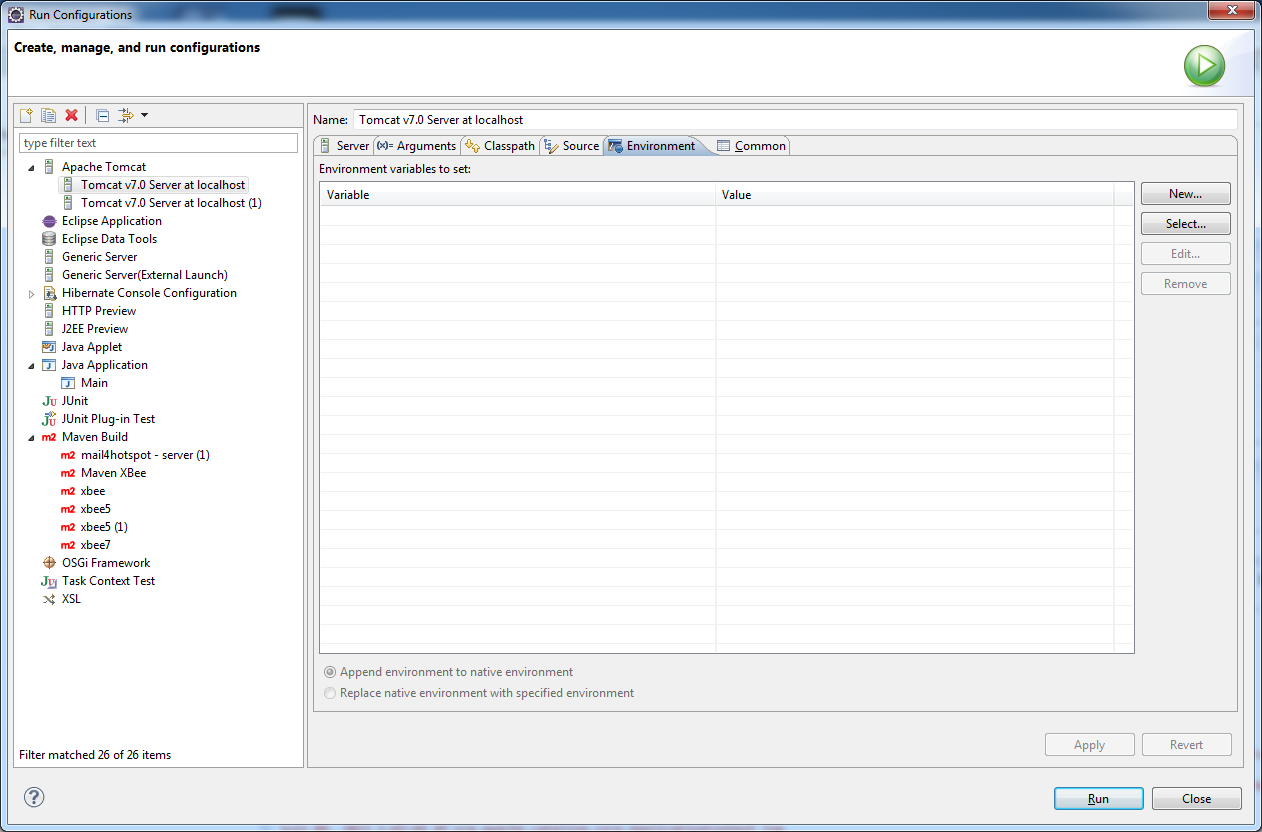
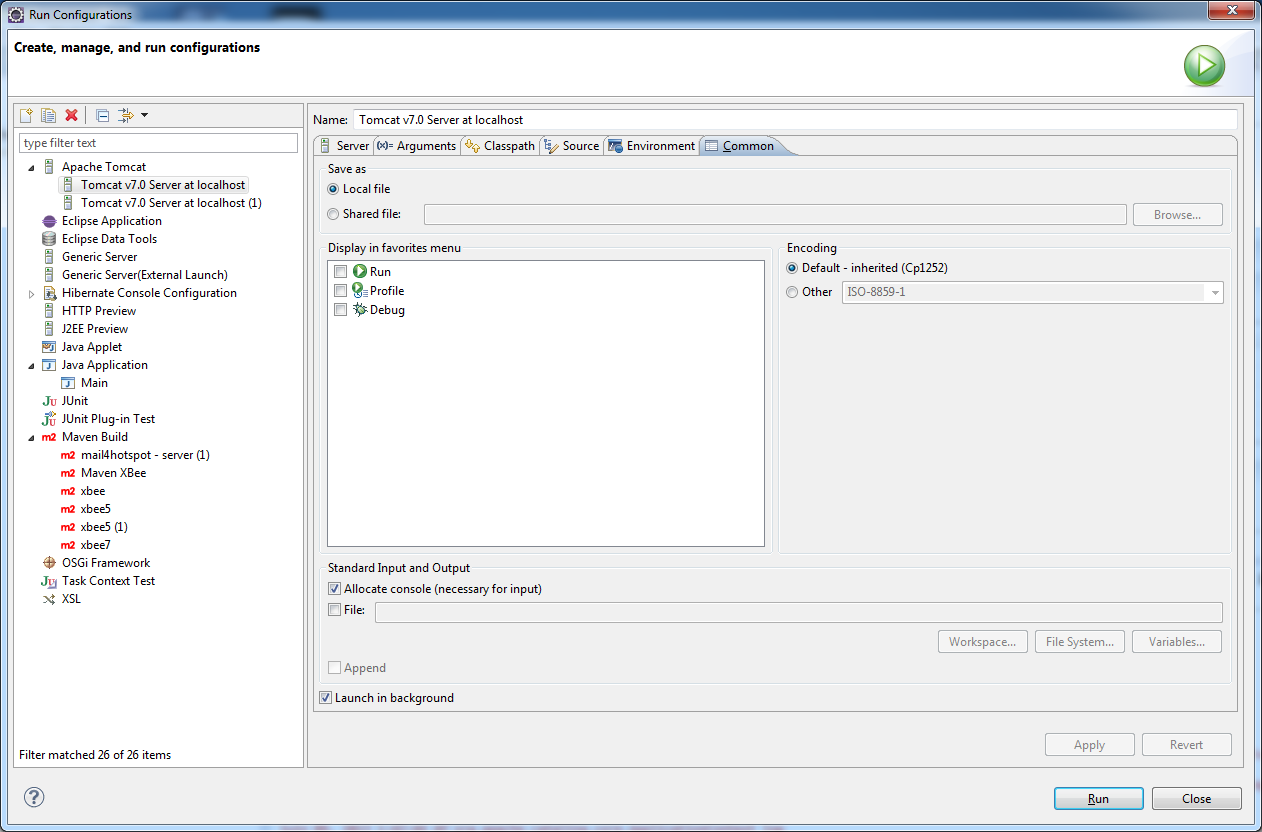


Créer une nouvelle debug config :

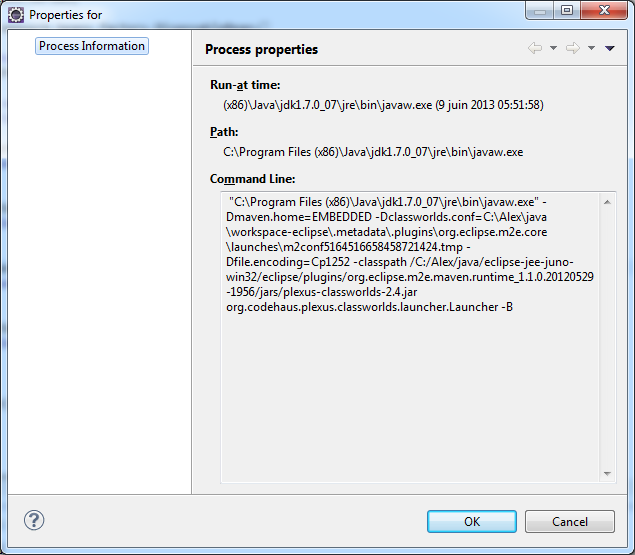


Si la debug config a « arguments » vide, je ne sais pas d’où vient ce bug (même en changeant de workspace ca continue à le faire), alors copier plutôt le serveur initial ou en créer un nouveau et remplir comme suit :



Astuce : si on veut savoir ce qui se passe quand eclipse lance par ex maven ou un serveur ou une appli, regarder la vue debug



Pour débugger :

Sur virt :

while true; do socat -d /dev/ttyUSB0,b57600,echo=0,raw TCP-LISTEN:9000,reuseaddr; done

Sur unix/linux :

stty raw; socat - tcp:virt:9000

pour tester :

virt: /home/fenyo/devel/scripts\_IR

echo send commands/final/tv\_power.ir 1 | ./netcatx w7 9999