|  |
| --- |
| http://www.tecgurus.net/images/logojavagurusconfondotrasn.png |
| **Java Frameworks EE** |
| Apuntes de curso |
|  |
| **Raúl Gerardo Vázquez González**  **Giovanny Rodriguez Aguilar** |
| **Junio 2013** |

|  |
| --- |
| Este documento es destinado como apuntes de apoyo y ejercicios para los alumnos del curso Java Frameworks EE. |

Contenido

[Introducción 3](#_Toc359277241)

[Maven 4](#_Toc359277242)

[Maven eclipse plugin 5](#_Toc359277243)

[Instalar maven plugin desde el sitio de marketplace de eclipse 5](#_Toc359277244)

[Instalar maven eclipse plugin desde Elcipse/Help/Eclipse Marketplace… 5](#_Toc359277245)

[Laboratorio 1: Creando un proyecto maven 6](#_Toc359277246)

[Estructura proporcionada por Maven. 11](#_Toc359277247)

[Struts2 12](#_Toc359277248)

[Creación del proyecto 13](#_Toc359277249)

[Dependencias en el pom.xml 13](#_Toc359277250)

[Configuración web.xml 14](#_Toc359277251)

[Archivo struts.xml 14](#_Toc359277252)

[Laboratorio 2: Creando nuestra propia acción 15](#_Toc359277253)

[¿Por dónde empezar? 16](#_Toc359277254)

[Crear el proyecto usando Maven 16](#_Toc359277255)

[Agregando nuestra nueva acción 17](#_Toc359277256)

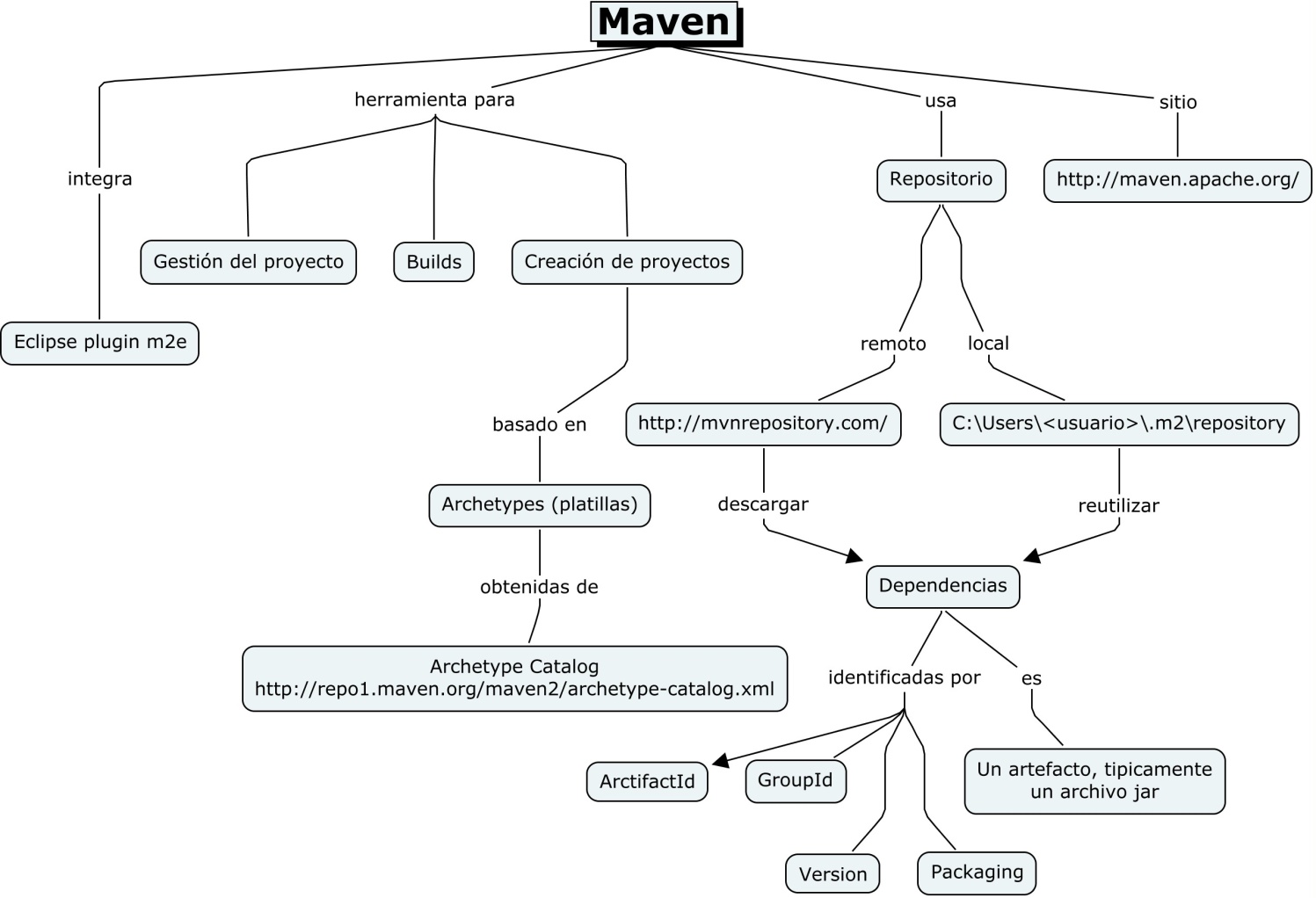
[Configurando Struts2 para funcionar con nuestra nueva acción 17](#_Toc359277257)

# Introducción

El presente manual no debe considerarse terminado; todo lo que aparece en el tendrá que ser revisado varias veces antes. Las gráficas y las pantallas que se muestran aquí tienen el único propósito de ilustrar la funcionalidad y pasos a seguir durante la evolución del presente curso.

# Maven

Apache Maven[[1]](#footnote-1) es un software de gestión de proyectos y una herramienta de comprensión. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar construcción de un proyecto, los informes y la documentación de una pieza central de la información.



## Maven eclipse plugin

Maven puede ser integrado como herramienta a nuestra instalación de eclipse favorita, existe más de una forma de integrar el plugin de maven en eclipse, a continuación discutiremos un par de ella:

### Instalar maven plugin desde el sitio de marketplace de eclipse

Work In Progess…

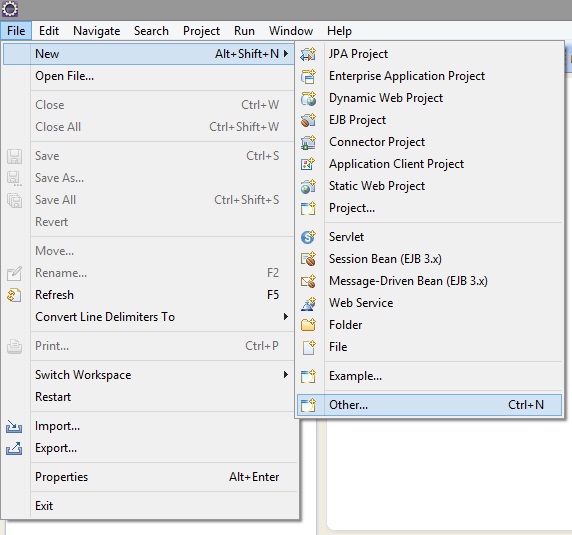
### Instalar maven eclipse plugin desde Elcipse/Help/Eclipse Marketplace…

Work In Progess…

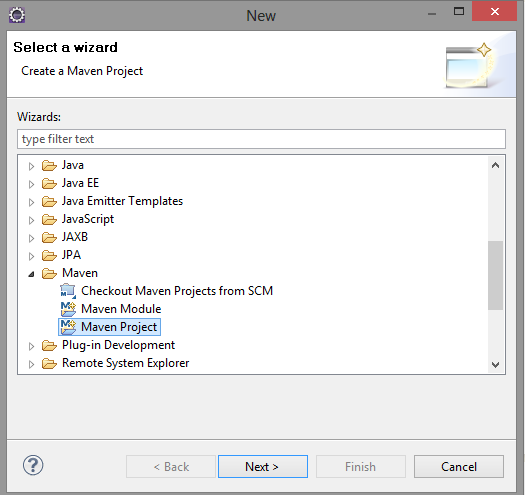
## Laboratorio 1: Creando un proyecto maven

Podemos crear un nuevo proyecto Maven desde nuestro eclipse en cualquier momento que deseemos, para lo cual seguiremos los siguientes pasos:

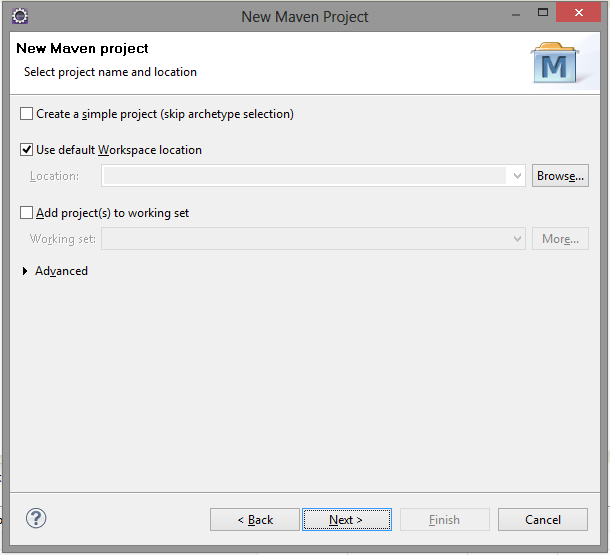
Menú File 🡪 New 🡪 Other…



Dentro de la ventana buscamos “Maven Project”

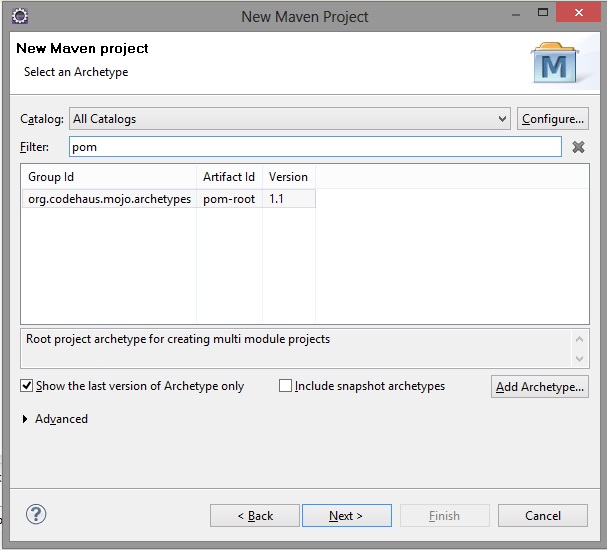


Dentro del primer dialogo dejamos las opciones default y damos “Next>”



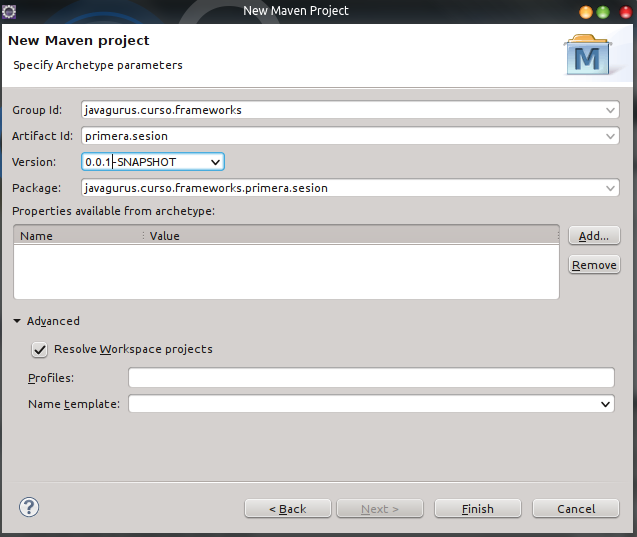
***Tip: Seleccionemos esta casilla si deseamos crear un proyecto Maven default, es cual es un proyecto java*** *simple (.jar)*

Dentro de la ventana de selección de un Archetype, en el cuadro de “Filter”, tecleamos “pom-root[[2]](#footnote-2)”, si no encontramos resultados al teclear “pom-root” en el filtro, probemos agregando un “remote catalog[[3]](#footnote-3)”



El siguiente menú nos pedirá información acerca de cómo queremos nombrar a nuestro proyecto, puedes ocupar los nombres que gustes, las convenciones son:

* Group Id: Es el nombre del proyecto principal, o de toda la aplicación.
* Artifact Id: Es el nombre del proyecto que se creará, un módulo de toda la aplicación.



* Versión Id: Es el nombre de la versión actual del artefacto/módulo que se está creando.

Generalmente el nombrado de los proyectos se hace de acuerdo a políticas o acuerdos de la organización que estará desarrollando el software, sin embargo hay algunos principios generales que puedes consultar en los siguientes enlaces:

[Para el nombrado de proyecto](http://maven.apache.org/guides/mini/guide-naming-conventions.html).

[Para el nombrado de versiones](http://www.sonatype.com/books/mvnref-book/reference/pom-relationships-sect-pom-syntax.html).

Al dar clic en finalizar se creará el proyecto.

## Estructura proporcionada por Maven.

Maven nos proporciona una plantilla de organización de carpetas para el código,

* **src/main/java** : Es una carpeta para colocar todo el código que sea programado con java, todo lo que está aquí se compilará a un archivo de extensión .class y correrá con la aplicación.
* **src/main/resources**: Es la carpeta para colocar todo lo que sean recursos de configuración y archivos que no se compilan.
* **src/test/java**: Aquí va todo el código de java que se usará para pruebas, pruebas unitarias y pruebas de integración, generalmente lo que está aquí se compila pero no corre con la aplicación.
* **src/main/webapp**: Aquí van archivos de configuración para el contenedor de aplicaciones ([web.xml](http://docs.oracle.com/javaee/6/tutorial/doc/bnadx.html)), y la parte de la “vista”, es decir, jsp, recursos de html, javascript y css.

# Struts2

[Struts2](http://struts.apache.org/development/2.x/) es uno de los Frameworks Java web más extensamente usado, está diseñado para funcionar con el paradigma de MVC

## Creación del proyecto

Para crear un proyecto Struts2 con la ayuda de Maven, es recomendable utilizar el archetype “struts2-archetype-blank”, la platilla “struts2-archetype-blank”, no simplifica la vida creando y configurando un proyecto con lo necesario empezar a trabajar con el framework Struts2, pero ¿Qué es lo necesario para hacer funcionar Struts2 en un proyecto Web?, ¡muy buena pregunta!, exploremos juntos la lo que el archetype creo por nosotros:

### Dependencias en el pom.xml

Nos agregó las librerías de Struts2 al proyecto:

…

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

<version>${struts2.version}</version>

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-config-browser-plugin</artifactId>

<version>${struts2.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.4</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>2.0</version>

<scope>provided</scope>

</dependency>

…

<raíz>/pom.xml

### Configuración web.xml

Nos ayudo registrando el filter del framework de Struts2

…

<filter>

<filter-name>struts2</filter-name>

<filter-class>

org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter

</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

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

</filter-mapping>

…

<raíz>/src/main/webapp/WEB-INF/web.xml

**Nota: El valor *url-pattern* indica el patrón de url que estará escuchando el filtro**

### Archivo struts.xml

Es el archivo corazón de la configuración del framework de Struts2:

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

"http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

<constant name=*"struts.enable.DynamicMethodInvocation"* value=*"false"*/>

<constant name=*"struts.devMode"* value=*"true"*/>

<include file=*"example.xml"*/>

<!-- Add addition packages and configuration here. -->

</struts>

<raíz>/src/main/resources/struts.xml

Resaltando la tag *<include file=”example.xml”>*, ¿Qué significa? ¿Para qué sirve?, bueno de entrada hace un poco más que ocuparnos una línea en el archivo ;), la plantilla de Maven nos está sugiriendo configurar nuestro Struts2 utilizando archivos XML, pero lo que podemos tener en un archivo XML de 800 mil líneas, lo podemos partir en 8 archivos de 100 mil líneas, o en 8 mil archivos de mil líneas, claro respetando siempre las reglas de XML, lo que se nos haga más comodo.

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

"http://struts.apache.org/dtds/struts-2.3.dtd">

<!--

- This file is included by the struts.xml file as an example

- of how to break up the configuration file into multiple files.

-->

<struts>

<package name=*"example"* namespace=*"/example"* extends=*"struts-default"*>

<action name=*"HelloWorld"* class=*"com.tecgurus.whattimeisit.web.example.HelloWorld"*>

<result>/example/HelloWorld.jsp</result>

</action>

<action name=*"Login\_\*"* method=*"{1}"* class=*"com.tecgurus.whattimeisit.web.example.Login"*>

<result name=*"input"*>/example/Login.jsp</result>

<result type=*"redirectAction"*>Menu</result>

</action>

<action name=*"\*"* class=*"com.tecgurus.whattimeisit.web.example.ExampleSupport"*>

<result>/example/{1}.jsp</result>

</action>

<!-- Add additional "example" package actions here. -->

</package>

</struts>

<raíz>/src/main/resources/example.xml

**Nota: Contiene la configuración default para el funcionamiento**

**Nota: namespace corresponde al fragmento de la url que contiene las acciones */example*/HelloWorld**

**Nota: name corresponde al fragmento final de la url */example*/HelloWorld**

## Laboratorio 2: Creando nuestra propia acción

Muy bien nuestro más querido usuario no ha solicitado un nuevo requerimiento:

Hola mundo!!

**/holamundo/HolaMundo.action**

*Como un usuario de la nueva super aplicación, quiero poder ver el mensaje de “hola mundo!!”.*

Al cliente lo que pida verdad, pues ¡manos a la obra!

### ¿Por dónde empezar?

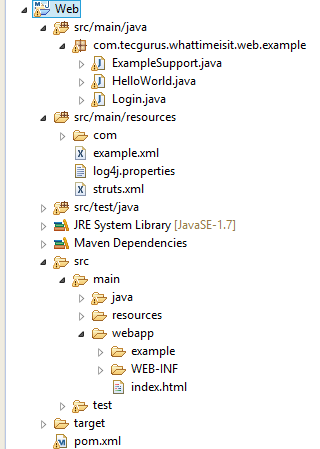
Yo creo que por el principio, no es una idea que se me haya ocurrido a mí, a lo mejor por eso es buena idea ;).

### Crear el proyecto usando Maven

Necesitamos crear un nuevo proyecto Maven, utilizando el archetype “struts2-archetype-blank”, si por alguna razón no sale como resultado del filtro, agreguemos el catalogo remoto de archetypes.

Nota: Sientete libre de ***utilizar el grupo, artifacId, versión y package a tu gusto.***

Ya que tenemos el proyecto de Maven listo sigamos adelante.



### Agregando nuestra nueva acción

Para este punto ya tenemos un nuevo, configura y reluciente proyecto Struts2, listo para empezar a ensuciarnos las manitas, pues ¡adelante!

Best practice: Crear los paquetes de .actions, .models a nivel del paquete principal

package com.tecgurus.whattimeisit.web.actions;

import com.opensymphony.xwork2.ActionSupport;

public class HolaMundoAction extends ActionSupport {

@Override

public String execute() throws Exception {

return *SUCCESS*;

}

}

<raíz>/src/main/java/com/tecgurus/whattimeisit/web/actions/HolaMundoAction.java

**Tip: Hereda de la clase de ActionSupport, te ahorrará más de una línea de código en futuro cercano**

**Nota: El método execute es el método llamado al ejecutar la acción, debe de llamarse “execute”**

En nuestro paquete de actions creamos una nueva clase llamada HolaMundoAction.

Best practice: Terminar el nombre de nuestras clases acción con “Action”

### Configurando Struts2 para funcionar con nuestra nueva acción

1. Para más información visite <http://maven.apache.org/> [↑](#footnote-ref-1)
2. “pom-root” es el archetype de base usado para la creación de un proyecto modular, puede ser sustituido por cualquier archetype deseado. [↑](#footnote-ref-2)
3. Un remote catalog, es un archivo XML con las descripciones de archetypes usados por maven, el remote catalog por excelencia es **http://repo1.maven.org/maven2/archetype-catalog.xml** [↑](#footnote-ref-3)