

Jenkins Java

En este tutorial describiremos los pasos realizados para integrar un proyecto Java en la herramienta Jenkins.

Configuración de Jenkins

Antes de comenzar con el proyecto debemos realizar un pequeño proceso de configuración que consistirá en añadir los plugins necesarios y otras herramientas que puedan hacernos falta.

Instalación de Plugins

Para este proyecto los plugins que tendremos que tener instalados son los siguientes:

Ant: Este plugin será necesario para compilar usando Ant

Javadoc y Jenkins JavaTest report plugin: Ambos plugins nos servirán para documentación tanto del código como de las pruebas realizadas.

Instalación de Ant

Será necesario también instalar la herramienta Ant en nuestro equipo para poder realizar la compilación del proyecto desde Jenkins. En un sistema Unix (Mac OS X en este caso), simplemente debemos descargar el binario en el siguiente enlace:

<http://ant.apache.org/bindownload.cgi>

Una vez tengamos el archivo descargado y descomprimido, lo movemos a un directorio reconocido y añadimos la ruta al PATH. Este último paso cambiará en función del sistema operativo, por lo que se recomienda consultar cómo *añadir variables de Entorno* en el sistema que estemos utilizando.

En Mac OS X, ejecutaremos desde un terminal:

```
open ~/.profile
```

En el fichero que se nos abre añadimos la línea:

```
PATH={ruta-de-ant}/bin:$PATH
```

Guardamos y cerramos, ahora comprobaremos que se ha instalado correctamente, ejecutando desde línea de comandos la instrucción:

```
ant
```

A lo que deberíamos recibir, en caso de estar correctamente configurado, la siguiente salida:

```
Buildfile: build.xml does not exist!
```

```
Build failed
```

Creación de la tarea

Finalizada la configuración previa del sistema, pasaremos a crear una nueva tarea. Nos dirigimos ahora a nuestro navegador e introducimos la dirección:

localhost:8080 (Si se ha modificado la configuración de Jenkins, debemos introducir el puerto o dirección que hayamos especificado).

Nos encontramos con la siguiente imagen:

Jenkins

buscar

DESACTIVAR AUTO-REFRESCO

insertar descripción

Nueva Tarea

Personas

Historial de construcción

Administrar Jenkins

Trabajos en cola

No hay trabajos en cola

Estado de los nodos

#	Estado
1	Inactivo
2	Inactivo

Todo +

S	W	Nombre ↓	Último éxito	Último fallo	Última duración
		Tarea3	6 días 1 Hor (#37)	6 días 0 Hor (#49)	0,67 Seg

Ícono: S M L

Guía de íconos RSS de todos RSS de los fallidos RSS de los más recientes

Avudamos a traducir ésta página

Página generada el: 26-dic-2012 13:01:52 REST API! Jenkins ver. 1.494

Seleccionamos la primera opción, **Nueva Tarea**, que nos llevará a una nueva pantalla donde comenzaremos a integrar nuestro proyecto.

De entre todas las opciones presentes seleccionaremos **Crear un proyecto de estilo libre**.

También podría resultar interesante la opción **Crear un proyecto maven2/3**, ya que simplificaría muchísimo la configuración de la tarea y la integración con otras herramientas con Sonar (Requiere Maven para funcionar con Jenkins). Sin embargo esta opción requiere que el proyecto cuente con un fichero POM. Sin embargo al desconocer el formato y funcionamiento de estos ficheros hemos optado por una opción más artesanal.

Jenkins

buscar

Jenkins Todo

Nueva Tarea

Personas

Historial de construcción

Administrar Jenkins

Trabajos en cola

No hay trabajos en cola

Estado de los nodos

#	Estado
1	Inactivo
2	Inactivo

Nombre del Tarea TareaJava

☒ **Crear un proyecto de estilo libre**

Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.

☐ **Crear un proyecto maven2/3**

Ejecuta un proyecto maven2/3. Jenkins es capaz de aprovechar la configuración presente en los ficheros POM, reduciendo drásticamente la configuración.

☐ **Crear un proyecto multi-configuración**

Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en múltiples entornos, ejecutar sobre plataformas concretas, etc.

☐ **Monitorizar una tarea externa**

Este tipo de tareas te permite registrar la ejecución de un proceso externo a Jenkins, incluso en una máquina remota. Está diseñado para usar Jenkins como un panel de control de tu sistema de automatización. Para más información consulta esta [página](#).

☐ **Copiar un Tarea existente**

Copiar desde

OK

Avudamos a traducir ésta página

Página generada el: 26-dic-2012 13:02:40 REST API! Jenkins ver. 1.494

Pulsamos **Ok**, y ya tenemos nuestra tarea creada. Pasamos ahora a la configuración de la misma.

En la siguiente pantalla nos aparecerán todas las opciones de configuración que nos ofrece Jenkins para cualquier tarea.

Lo primero que veremos es dos campos de texto, uno para el nombre y otro para la descripción.

Más abajo encontraremos otras opciones que en este caso no vamos a tocar, ya que no son necesarias para el tutorial que nos ocupa.

The screenshot shows the Jenkins job configuration interface. At the top, there are two text input fields: 'Proyecto nombre' (Project name) containing 'TareaJava' and 'Descripción' (Description) containing 'Tarea de un proyecto Java para el tutorial de Jenkins. Laboratorio de desarrollo y herramientas' and 'Grupo triple A'. Below these is a 'Vista previa' (Preview) button. A list of checkboxes follows: 'Desechar ejecuciones antiguas' (Discard old builds), 'Esta ejecución debe parametrizarse' (This build must be parametrized), 'Desactivar la ejecución (No se ejecutará nuevamente hasta que el proyecto sea reactivado.)' (Disable the build), and 'Lanzar ejecuciones concurrentes en caso de ser necesario' (Launch concurrent builds if necessary). A section titled 'Opciones avanzadas del proyecto' (Advanced project options) contains more checkboxes: 'Periodo de espera' (Wait period), 'Contador de reintentos' (Retry counter), 'Congelar el lanzamiento cuando haya un proyecto padre ejecutándose' (Freeze the launch when there is a parent project running), 'Bloquear la ejecución cuando un proyecto relacionado está en ejecución' (Block the build when a related project is running), and 'Utilizar un directorio de trabajo personalizado' (Use a custom workspace directory). At the bottom is a 'Nombre a mostrar' (Name to display) field. On the right side of the page, there is a vertical column of question mark icons.

El siguiente paso será elegir el origen del código fuente. La herramienta nos facilitará integrar la tarea con un proyecto desde distintos repositorios, sin embargo, nosotros disponíamos de un código que se encontraba en una máquina en local, por lo que nos interesaba trabajar con esa opción. Por eso en este paso elegimos **Ninguno**.

Toca ahora configurar cómo y qué se va a ejecutar en nuestro proyecto.


En función de los plugins que tengamos instalados, nos aparecerán más o menos opciones en la lista desplegable de **Añadir un nuevo paso**.

A nosotros nos interesa la opción de **Ejecutar Ant**.

Creamos una ejecución de Ant, y esto nos permitirá añadir opciones como el Fichero Ant que queremos usar o las opciones de Java a la hora de realizar la compilación.

Como no tenemos gran conocimiento de la herramienta, dejaremos la configuración por defecto.

Jenkins TareaJava configuración



Configurar el origen del código fuente

☐ CVS
☐ CVS Projectset
☒ Ninguno
☐ Subversion

Disparadores de ejecuciones

☐ Ejecutar después de que otros proyectos se hayan ejecutado
☐ Consultar repositorio (SCM)
☐ Ejecutar periódicamente

Ejecutar

Ejecutar Ant

Versión de Ant: Por defecto

Destinos

Fichero Ant

Propiedades

Opciones de java

Borrar

Añadir un nuevo paso

Acciones para ejecutar después.

Añadir una acción

Guardar Aplicar los cambios

Ayudanos a traducir ésta página

Página generada el: 26-dic-2012 13:09:13 [REST API](#) Jenkins ver. 1.494

Pulsamos **Guardar** y nos llevará a la siguiente pantalla, donde veremos un panel a la izquierda con diferentes opciones.

Jenkins

Jenkins TareaJava

 [Volver al Panel de control](#)

 [Estado](#)

 [Cambios](#)

 [Espacio de trabajo](#)

 [Construir ahora](#)

 [Borrar Proyecto](#)

 [Configurar](#)

Historia de tareas (tendencia)

#1 26-dic-2012 13:09:10

 RSS para todos  RSS para los fracasos

Proyecto TareaJava

Tarea de un proyecto Java para el tutorial de Jenkins. Laboratorio de desarrollo y herramient

 [Espacio de trabajo](#)

 [Cambios recientes](#)

Enlaces Permanentes

- [hace Última ejecución \(#1\), 12 Min](#)
- [hace Última ejecución estable \(#1\), 12 Min](#)
- [hace Última ejecución correcta \(#1\), 12 Min](#)

Ayudanos a traducir ésta página

Si elegimos la opción de construir ahora veremos que la ejecución se realiza correctamente, sin embargo ¿qué estamos haciendo?. ¡Pues realmente nada!, no hemos configurado el espacio de trabajo.

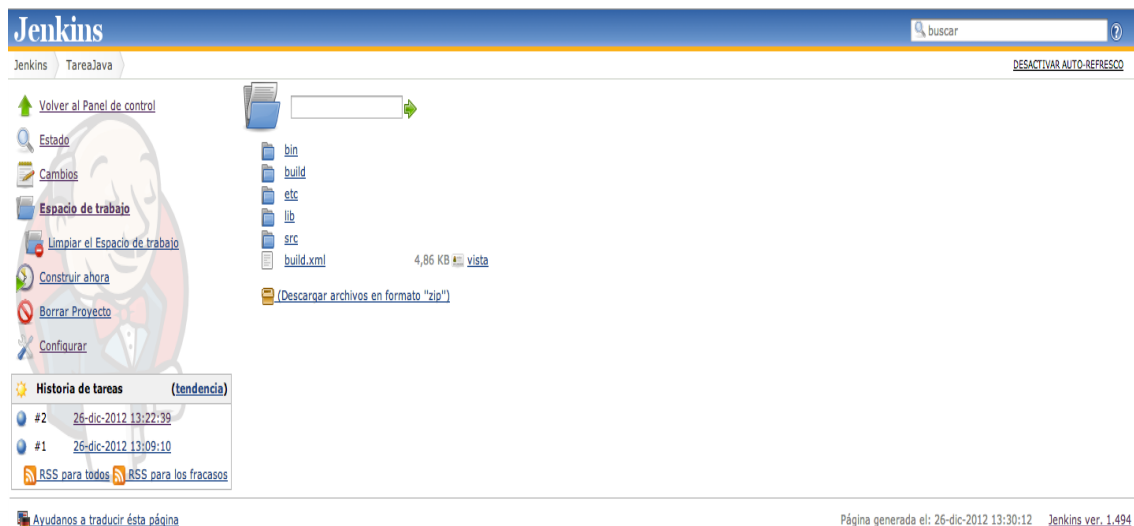
Cargar los ficheros en el Workspace

Como ya hemos dicho, este proyecto lo vamos a trabajar desde un directorio local, por lo que es necesario cargar los ficheros en el Workspace de la tarea. Si pulsamos la opción de **Espacio de Trabajo**, nos indicará que no hay ficheros en el directorio.

Para solucionar esto debemos buscar el directorio Workspace de nuestra tarea en nuestro sistema, en este caso en la ruta:

`/Users/Shared/Jenkins/Home/workspace/TareaJava/`

Y copiar los ficheros del proyecto. Si recargamos la pantalla del espacio de trabajo veremos como ahora aparecen todos los ficheros que hemos copiado.



Ya tenemos nuestro proyecto cargado en el directorio de trabajo, ahora solo falta compilarlo, para verificar que no existen fallos.

Compilar con Ant

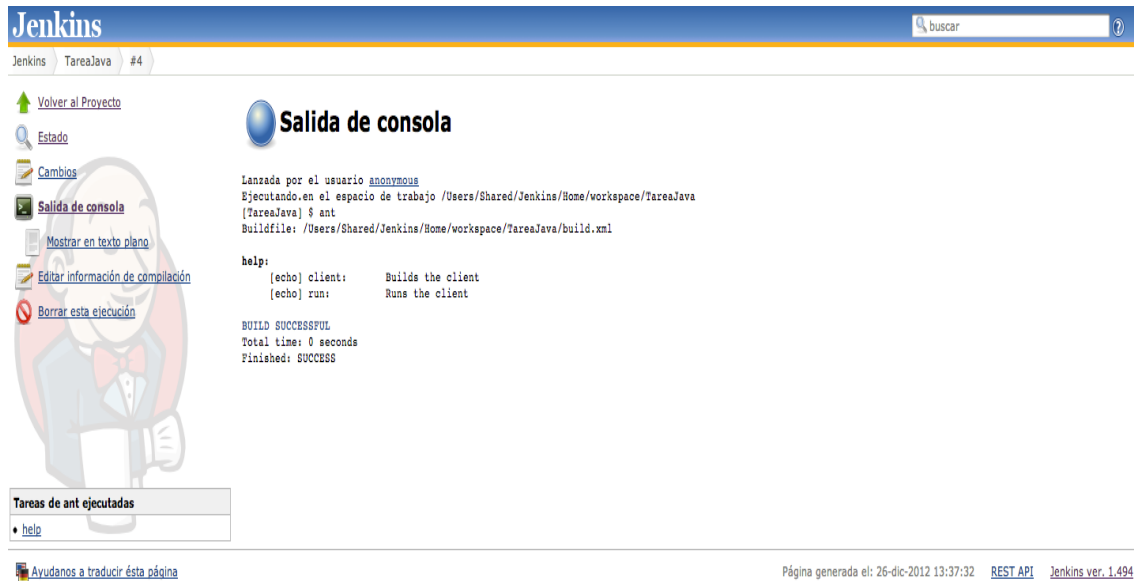
Como le indicamos a Jenkins en la configuración de la tarea, utilizaremos la herramienta Ant para compilar nuestro proyecto.

Este es un proceso realmente sencillo, donde la única dificultad la encontramos a la hora de crear el documento de configuración, un documento similar al Makefile al que estamos acostumbrados, pero en este caso se trata de un XML.

Al no tratarse de algo que hayamos estudiado con anterioridad, no conocemos suficientemente su funcionamiento, sin embargo si disponemos de un IDE como NetBeans o Eclipse para desarrollar nuestro proyecto, el propio Entorno se encargará de genera el fichero build.xml

Una vez hayamos superado este paso, simplemente pulsamos **Construir ahora**, y nos dirigimos a **Salida de Consola**, para ver el resultado.

Si no existe ningún error en la configuración del XML y el proyecto compila correctamente tendremos la siguiente salida.



The screenshot shows the Jenkins web interface. At the top, there's a blue header with the Jenkins logo and a search bar. Below the header, the breadcrumb navigation shows 'Jenkins' > 'TareaJava' > '#4'. On the left sidebar, there are links: 'Volver al Proyecto', 'Estado', 'Cambios', 'Salida de consola' (highlighted), 'Mostrar en texto plano', 'Editar información de compilación', and 'Borrar esta ejecución'. The main content area is titled 'Salida de consola' and shows the following text:

```
Lanzada por el usuario anonymous
Ejecutando en el espacio de trabajo /Users/Shared/Jenkins/Home/workspace/TareaJava
[TareaJava] $ ant
Buildfile: /Users/Shared/Jenkins/Home/workspace/TareaJava/build.xml

help:
  [echo] client:    Builds the client
  [echo] run:       Runs the client

BUILD SUCCESSFUL
Total time: 0 seconds
Finished: SUCCESS
```

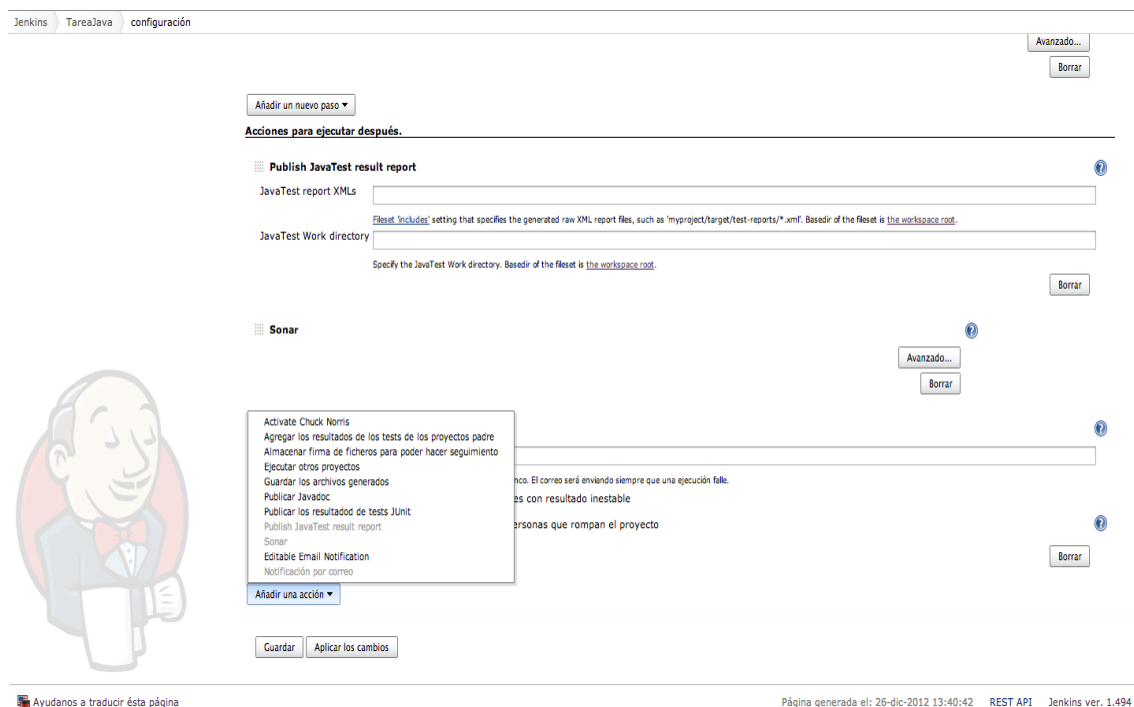
Below the console output, there's a section 'Tareas de ant ejecutadas' with a link to 'help'. At the bottom right, it says 'Página generada el: 26-dic-2012 13:37:32' and 'Jenkins ver. 1.494'.

Y ya tendríamos nuestro proyecto Java configurado e integrado con la herramienta Jenkins. Solo faltaría ver como podemos utilizarlo para realizar pruebas y obtener documentación.

Pruebas y Documentación

Para seleccionar estas opciones, debemos dirigirnos nuevamente a la configuración de tarea, e ir a la sección **Acciones para ejecutar después**.

Aquí cabe destacar las opciones de **Notificar por correo**, integrar con **Sonar**, **Publicar Javadoc**, **Publicar los resultados de test JUnit** o **Publicar los resultados de JavaTest**.



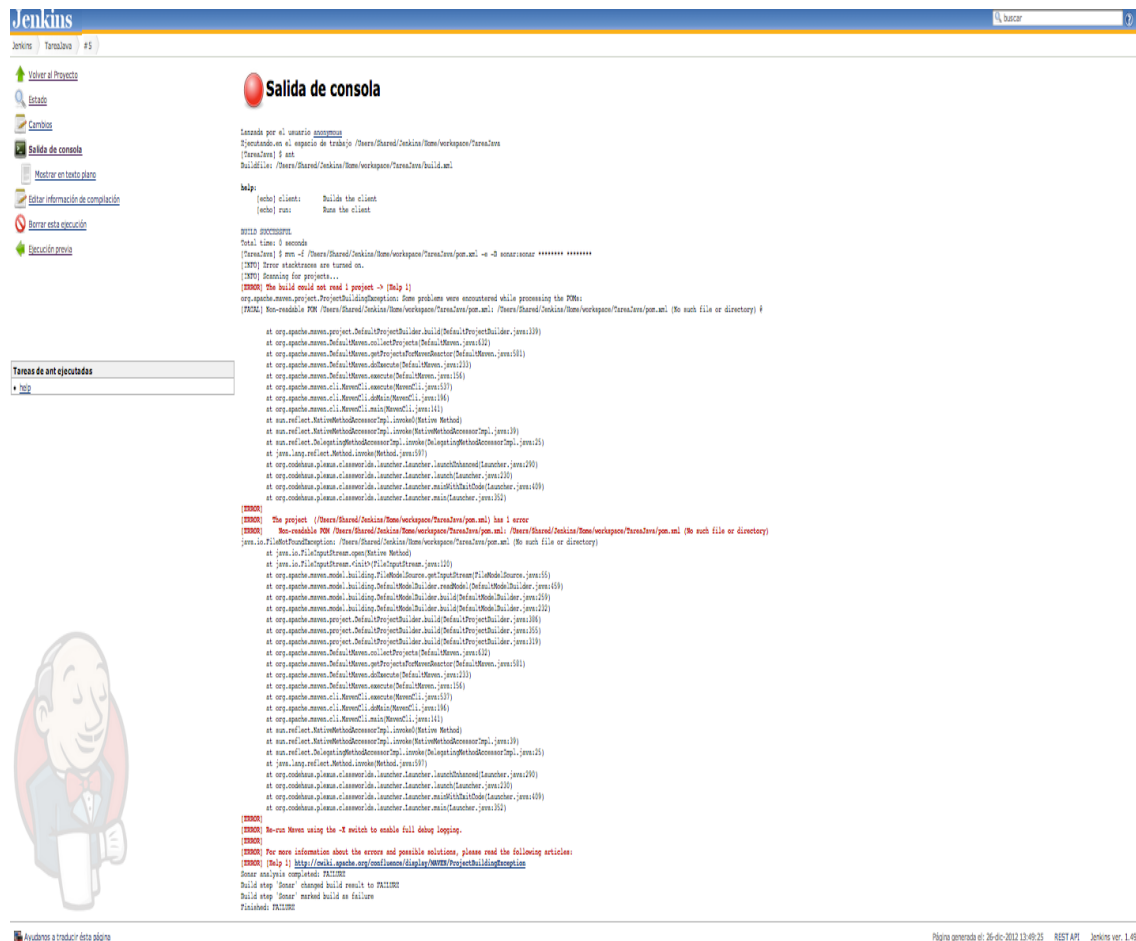
The screenshot shows the Jenkins configuration page for the 'TareaJava' job. The breadcrumb navigation shows 'Jenkins' > 'TareaJava' > 'configuración'. On the right side, there are buttons: 'Avanzado...', 'Borrar', and 'Añadir un nuevo paso'. The main content area is titled 'Acciones para ejecutar después.' and contains several configuration sections:

- Publish JavaTest result report**: Includes fields for 'JavaTest report XMLs' and 'JavaTest Work directory'. The 'JavaTest Work directory' field has a tooltip that says 'Specify the JavaTest Work directory. Basedir of the fileset is the workspace root.'
- Sonar**: Includes a list of actions to perform after the build, such as 'Activar Chuck Norris', 'Agregar los resultados de los tests de los proyectos padre', 'Almacenar firma de ficheros para poder hacer seguimiento', 'Ejecutar otros proyectos', 'Guardar los archivos generados', 'Publicar Javadoc', 'Publicar los resultados de tests JUnit', 'Publish JavaTest result report', 'Sonar', 'Editable Email Notification', and 'Notificación por correo'. There are buttons for 'Avanzado...', 'Borrar', and 'Añadir una acción'.

At the bottom, there are buttons for 'Guardar' and 'Aplicar los cambios'. At the bottom right, it says 'Página generada el: 26-dic-2012 13:40:42' and 'Jenkins ver. 1.494'.

De todas estas opciones tratamos de probar la integración con Sonar, cuya configuración corresponde explicarla a otro compañero, sin embargo, tras muchos intentos de configuración, nos topamos con que nos exige el uso de Maven, el cual exige, de la misma forma que Ant un XML, un fichero POM que no supimos configurar, por lo que no nos fue posible comprobar el resultado de la integración de ambas herramientas.

El resultado que obteníamos al realizar una ejecución era el siguiente



Y hasta aquí este tutorial de uso de la herramienta Jenkins para un proyecto en Java.

Realizado por:

*Adrián González Hernández
Alejandro Correa Rodríguez
Andrés Nacimiento García*