# Selenium

Selenium es un conjunto de herramientas de código abierto que nos ayuda a automatizar acciones que un usuario puede realizar sobre aplicaciones web. Cada herramienta dentro de este conjunto tiene un enfoque diferente para apoyar el proceso de automatización de pruebas.

Los cuatro componentes de Selenium son:

1. Selenium IDE: es un entorno de desarrollo integrado para scripts de Selenium. Se implementa como una extensión de Firefox y permite grabar, editar y depurar pruebas.Permite grabar y reproducir de forma fácil las pruebas en el entorno real en el cual se ejecutarán.
2. Selenium RC (Remote Control): es una herramienta para automatizar pruebas de interfaz de usuario (UI) de aplicaciones web. Consta de dos componentes: a) un servidor que actúa como proxy para controlar e interactuar con un navegador web. b) bibliotecas para crear programas
3. Selenium WebDriver: también es una herramienta para automatizar pruebas UI de aplicaciones web pero implementa un enfoque más moderno y estable que Selenium RC. WebDriver, a diferencia de RC no utiliza un middleware sino que controla el navegador comunicándose directamente con él.
4. Selenium Grid: se especializan en ejecutar múltiples pruebas a través de diferentes navegadores, sistemas operativo y máquinas. Puede conectarse con Selenium Remote especificando el navegador, la versión del navegador y el sistema operativo que desee.

Selenium WebDriver :es un framework de automatización web que permite ejecutar casos de prueba sobre distintos navegadores. Debido a que es posible utilizar lenguajes de programación para la creación de scripts de pruebas, podemos tener estructuras de control como condiciones y bucles para controlar el comportamiento. Algunos de los lenguajes soportados son: Java, C#, Python, Ruby, PHP y JavaScript.

# MAVEN

* -**Construye** nuestro código del proyecto (build)
* Ejecuta las pruebas (**testing**)
* Trabaja de **gestor de dependencias**: nos importa automáticamente las bibliotecas desde un repositorio remoto ¡Se acabó el importarlas manualmente! Y almacena (cachea) estas bibliotecas en un repositorio local para agilizar las importaciones posteriores de las bibliotecas que ya tiene
* **Sube nuestro propio artefacto** (proyecto) para que otros se lo descarguen: no tiene por qué ser a un sitio público. Si trabajas en una empresa imagina las posibilidades de tener un repositorio con todas las bibliotecas o proyectos que hemos creado para toda la empresa. O para trabajar con los amigos y que se puedan descargar nuestra biblioteca que hemos creado y utilizarla.
* **Despliega el artefacto** (proyecto) en un servidor.
* **Descarga arquetipos** (artefactos/proyectos que son plantillas): nos descarga proyectos que son plantillas, como la estructura de un sitio web para Java EE, para tener ya la estructura de carpetas creada, con las clases más usadas ya escritas e importados las bibliotecas que siempre se necesitan; arrebatándonos la tediosa tarea de tener que crear lo mismo una y otra vez nosotros cuando alguien ya lo ha hecho antes
* Crea y despliega una **web con la documentación** del artefacto (proyecto).
* Se integra perfectamente con JIRA, SVN, Git

Para que veas todo el poder de Maven, te pongo un ejemplo rápido de lo que hace Maven: queremos hacer una web con Java EE. Primero desarrollaríamos en local la web. Y crearíamos unos test para probar nuestro código. Estamos trabajando en local; sabemos que cuando subamos el proyecto al servidor real las cosas no funcionan siempre igual de bien. Para subir el proyecto al servidor real de producción (el que verán los clientes o usuarios) tenemos que realizar los siguientes pasos:

1. Primero **validaríamos** y **compilaríamos** el proyecto (sino habrá que corregirlo en desarrollo)
2. Luego pasaríamos las **pruebas** hasta que estén todas correctas (si falla alguna prueba, a desarrollo otra vez)
3. **Empaquetaríamos** el proyecto en un archivo WAR (WAR es como JAR que sirve para distribuir una web, normalmente en Java)
4. A continuación **subiríamos al servidor** **real de pruebas**/desarrollo el fichero WAR
5. **Desplegaríamos** el fichero WAR
6. Volveremos a pasar las**pruebas de integración** en el servidor (si falla, ya sabes, a desarrollo)
7. Entonces el fichero WAR lo tenemos que **subir al servidor de producción** (o pre-producción, dependiendo de la arquitectura que sigas)
8. Para terminar con **desplegar** el proyecto, que está en el fichero WAR, en el servidor de producción (o pre-producción, dependiendo de la arquitectura que sigas)
9. Ya terminada la nueva versión del proyecto también nos interesará crear la **página de documentación**.



# PageObjects

<https://qanewsblog.com/2014/08/07/patrones-de-diseno-en-automatizacion-page-objects/>