



Programación orientada a aspectos Tutorial AspectJ



Brayan Sebastián Yepes
Joe Mendez
Juan Pablo Garzón
Diego Rubiano

</Tabla de contenido

AspectJ	{01}	→	{02}	Línea de tiempo
Weaving	{03}	→	{04}	Sintaxis
Instalación	{05}	→	{06}	Ejemplos
Quizziz	{07}		{08}	Referencias

</AspectJ

1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</AspectJ

AspectJ es una extensión de AOP (Aspect-Oriented Programming) para Java que permite la programación orientada a aspectos.

AspectJ proporciona una forma de separar y modularizar aspectos específicos de la funcionalidad de un programa que no encajan de manera natural en la estructura modular convencional basada en clases y objetos.

Estos aspectos, como la gestión de transacciones, el registro o la seguridad, pueden ser encapsulados en módulos llamados "aspectos".



</Línea de tiempo

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Línea de tiempo



</Weaving

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1

</Weaving

El weaving es el proceso de integrar (tejer) los aspectos en el código base de la aplicación.

Hay dos enfoques principales para el weaving en AspectJ: el weaving en tiempo de compilación y el weaving en tiempo de carga (load-time weaving).



</Weaving: Tiempo de compilación

- Ocurre durante la fase de compilación del código fuente.
- El compilador de AspectJ (ajc) integra los aspectos directamente en el bytecode generado de las clases base.
- El resultado es un conjunto de clases "tejidas" que contienen tanto el código base como el código de los aspectos.



</Weaving: Tiempo de carga

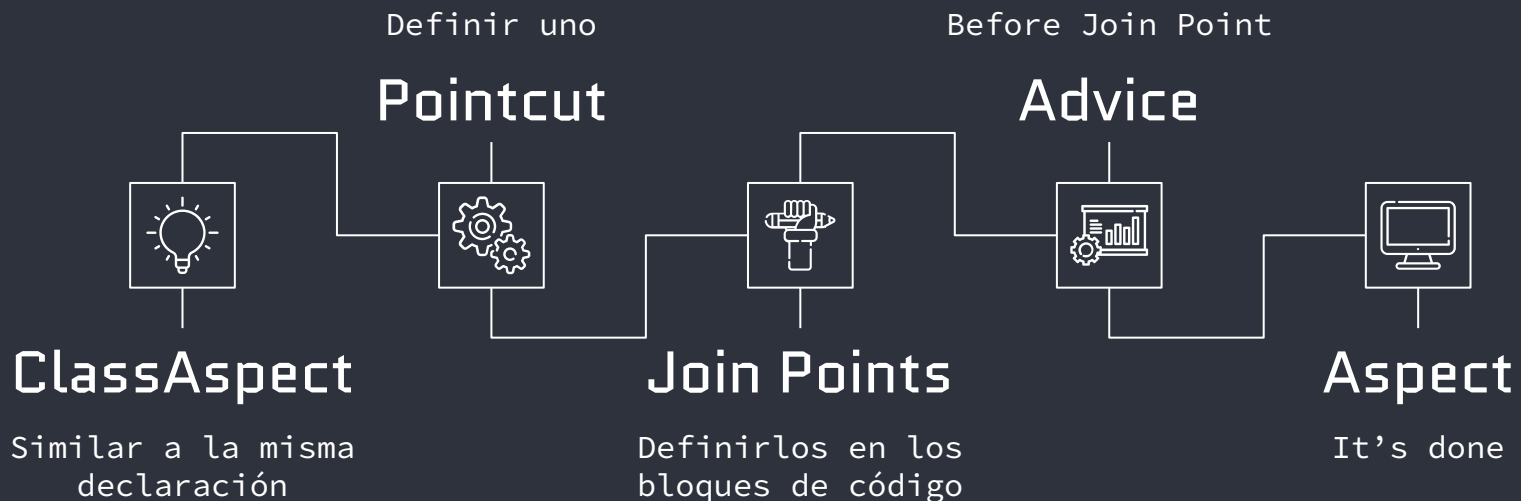


- Ocurre durante el tiempo de carga de las clases en tiempo de ejecución.
- El bytecode de las clases base se carga en la JVM, y en este punto, los aspectos se aplican dinámicamente al código ya cargado.
- Esto permite una mayor flexibilidad, ya que los aspectos pueden ser modificados sin requerir una nueva compilación.

</Sintáxis

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1

</Aspecto: Declaración



</Introducción: Join points

- Los programas viven para ejecutarse y cuando se ejecutan cosas suceden.
- Los métodos se llaman y los objetos se inicializan.
- Se accede a los campos y son actualizados.
- Los constructores son ejecutados y así sucesivamente...

Estos eventos cuando programa se está ejecutando AspectJ los llama JoinPoints

(Colyer, Clement, Harley & Webster, 2004, p.73)

</Join points

- AspectJ admite Join Points para llamadas realizadas a ejecución de métodos y constructores.
 - Puede referirse a:
 - Métodos individuales.
 - Clases completas.
 - Regiones de código específicas.
- Para la inicialización de clases y objetos, para accesos de campo y actualizaciones, para el manejo de excepciones, y algunos más.

</Pointcuts

- Puntos de corte o accesos directos
- Los Join Points que interesan se escriben en términos de Pointcuts
- Los Point Cuts son como filtros.
 - El programa se ejecuta y se producen un flujo de puntos de unión.
 - La tarea de un Pointcut es seleccionar los que le interesan.
 - Los Join Points que interesan ahora son `notifyListeners()`

(Colyer, Clement, Harley & Webster, 2004, p.73)

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Pointcuts: Sintaxis



```
pointcut NombreDelPointcut(): ModificadorDeAcceso TipoDeRetorno ClaseMetodo.metodo(parametros);
```

- NombreDelPointcut: Un identificador que se asigna
- ModificadorDeAcceso: Puede ser public, private, protected, o package.
- TipoDeRetorno: El tipo de retorno del método o * para cualquier tipo.
- ClaseMetodo: El nombre de la clase que contiene el método.
- metodo: El nombre del método al que el pointcut se aplicará.
- parametros: La lista de parámetros del método.

</Pointcuts:Join points

- execution:
 - Captura la ejecución de un método.
 - Tiene:
 - Modificador de acceso, El tipo de retorno, La clase y el nombre del método, Tipos de parámetros.
- call: Captura la llamada a un método desde otro método. Similar a execution, pero representa el punto de llamada, no el de ejecución.
- within: Captura la ejecución dentro de un cierto tipo de clase.
- this: Captura la ejecución de cualquier método cuyo receptor sea una instancia del tipo especificado.
- Target, Args,cflow,if entre otros

</Avisos

- Los advices son:
 - Fragmentos de código ejecutados en ciertos puntos de ejecución de la aplicación.
 - Similar a los "join points" que son puntos específicos de ejecución, como la llamada a un método o la creación de un objeto.
 - "Decide qué se hará con el join"

(Colyer, Clement, Harley & Webster, 2004, p.73)

</Avisos: Warnings

- Son declaraciones que permiten emitir mensajes de advertencia en tiempo de compilación.
- Son útiles para proporcionar información o recomendaciones sobre el código.
- A menudo se utilizan para comunicar sugerencias o mejores prácticas.
- No afectan directamente el flujo de ejecución del programa; son informativas.

</Avisos: Sintaxis

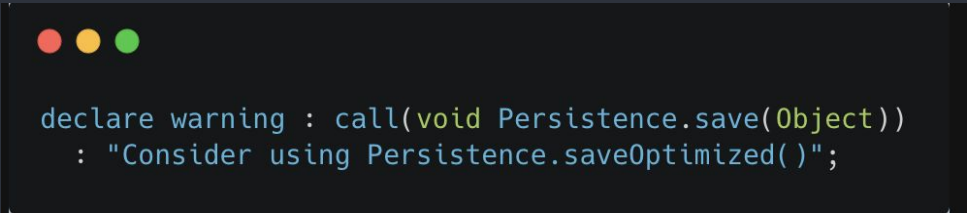
- Tipo de Join Point al que se aplicará la advertencia.
 - Execution
 - Call
 - Within
- Pattern: Patrón del join point al que se aplicará la advertencia.
- "Mensaje de Advertencia": Es el mensaje que se mostrará como advertencia.



```
declare warning: TipoJoinPoint Pattern  
: "Mensaje de Advertencia";
```

</Avisos Ejemplo

- se emite una advertencia en tiempo de compilación cuando se llama al método `save` en la clase `Persistence`.
- Sugiere considerar el uso de `Persistence.saveOptimized()` en su lugar.



```
declare warning : call(void Persistence.save(Object))
: "Consider using Persistence.saveOptimized()";
```

Ejemplo tomado y modificado de:

<https://eclipse.dev/aspectj/sample-code.html#declares-exceptionSpelunking>

</Declaración en tiempo de compilación

- Es una instrucción transversal estática que permite añadir advertencias y errores al detectar determinados patrones de uso.
- Tiene diferentes tipos:
 - Declaraciones de Tipos
 - Declaraciones de Importación
 - Declaraciones de Interfaz
 - Declaraciones de Anotaciones
 - Declaraciones de Advertencia o Error
 - Declaraciones de Pragmas o Directivas del Compilador

</Introducciones y declaraciones

- Añade instrucciones a las clases, interfaces y aspectos sin afectarlos directamente
- El aspecto es la unidad central de AspectJ. Analógicamente una clase en JAVA.
- Contiene el código que expresa las reglas de tejido para dinámicas y estáticas.

```
public aspect ExampleAspect {
    before() : execution(void Account.credit(float)) {
        System.out.println("About to perform credit operation");
    }

    declare parents: Account implements BankingEntity;
    declare warning : call(void Persistence.save(Object))
        : "Consider using Persistence.saveOptimized()";
}
```

Ejemplo tomado y adaptado de (Laddad,2009, p.36)

</Consejos

- “Modifying behavior with dynamic crosscutting”. Las reglas son:
 - Los consejos dicen que hacer (what to do).
 - Los Pointcuts cuándo aplicarlos (when to apply the advice).
 - Hay tres tipos de consejos.
 - Tiene una “Anatomía general”

(Laddad, 2009, p.160)

</Consejos

- Una comprobación de seguridad: Antes de ejecutar el Join Point.
- Un registro de excepciones: Después del Join Point y sólo si se lanza una excepción.
- El almacenamiento en caché:
 - Obtener un valor de la caché.
 - Si la caché no contiene valor, añadirlo y después de ejecutar el código.

(Laddad, 2009, p.160)

</Consejos

- El consejo “Before” se ejecuta antes de la ejecución del punto de unión
- El consejo “After” se ejecuta después de la ejecución del punto de unión:
 - “After finally” se ejecuta después de la ejecución del “Join Point” de acuerdo con la salida.
 - “After returning” se ejecuta después de una ejecución satisfactoria de un “Join Point” (Sin lanzar una excepción)
 - “After throwing” se ejecuta después de una ejecución fallida de un “Join Point” (Lanzar una excepción)

(Laddad, 2009, p.160)

</Consejos: Sintaxis

```
before() : PointcutExpression { //Code }
before() : execution(void MyClass.miMetodo()) {
    System.out.println("Antes de ejecutar miMetodo");
}
/////////////////////////////////////////////////////////////////

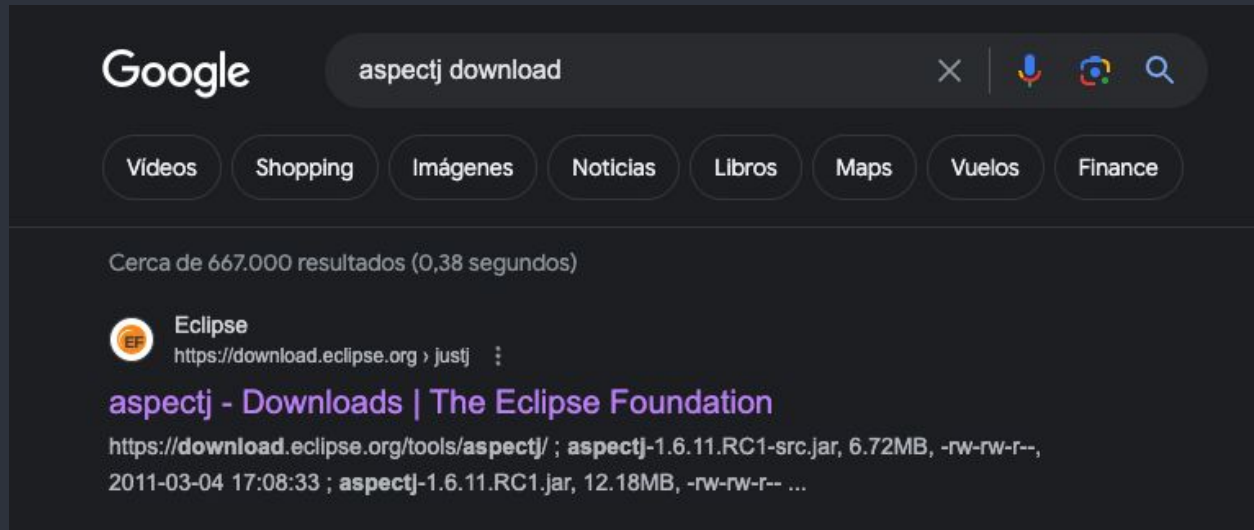
after() returning(TipoDeRetorno variable) : PointcutExpression { //Code }
after() returning(TipoDeRetorno variable) : PointcutExpression { //Code }
/////////////////////////////////////////////////////////////////
after() throwing(TipoDeExcepcion variable) : PointcutExpression {}
after() throwing(Exception ex) : execution(void MyClass.metodoConExcepcion()) {
    System.out.println("Después de lanzar la excepción en metodoConExcepcion: " + ex.getMessage());
}
/////////////////////////////////////////////////////////////////
after() : PointcutExpression { //Code }
after() : execution(* MyClass.*()) {
    System.out.println("Después de ejecutar cualquier método en MyClass");
}
```

</Instalación

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

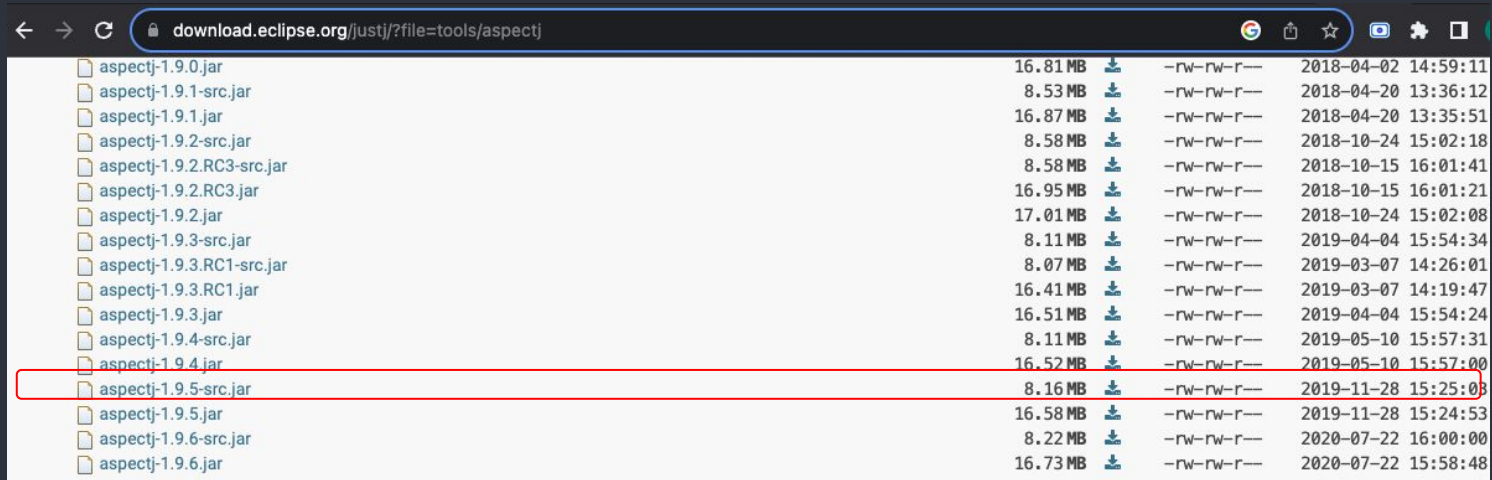
</¿Cómo instalarlo?

En el buscador de google escribir lo siguiente:



</¿Cómo instalarlo?

Descargar la última versión (en este caso la última es del 2020-07-22 pero puede variar según la fecha de consulta):



The screenshot shows a web browser window with the URL `download.eclipse.org/justj/?file=tools/aspectj`. The page displays a list of AspectJ jar files for download. The file `aspectj-1.9.5.jar` is highlighted with a red box, indicating it is the latest version available at the time of the screenshot.

File Name	Size	Permissions	Date	Time
aspectj-1.9.0.jar	16.81 MB	-rw-rw-r--	2018-04-02	14:59:11
aspectj-1.9.1-src.jar	8.53 MB	-rw-rw-r--	2018-04-20	13:36:12
aspectj-1.9.1.jar	16.87 MB	-rw-rw-r--	2018-04-20	13:35:51
aspectj-1.9.2-src.jar	8.58 MB	-rw-rw-r--	2018-10-24	15:02:18
aspectj-1.9.2.RC3-src.jar	8.58 MB	-rw-rw-r--	2018-10-15	16:01:41
aspectj-1.9.2.RC3.jar	16.95 MB	-rw-rw-r--	2018-10-15	16:01:21
aspectj-1.9.2.jar	17.01 MB	-rw-rw-r--	2018-10-24	15:02:08
aspectj-1.9.3-src.jar	8.11 MB	-rw-rw-r--	2019-04-04	15:54:34
aspectj-1.9.3.RC1-src.jar	8.07 MB	-rw-rw-r--	2019-03-07	14:26:01
aspectj-1.9.3.RC1.jar	16.41 MB	-rw-rw-r--	2019-03-07	14:19:47
aspectj-1.9.3.jar	16.51 MB	-rw-rw-r--	2019-04-04	15:54:24
aspectj-1.9.4-src.jar	8.11 MB	-rw-rw-r--	2019-05-10	15:57:31
aspectj-1.9.4.jar	16.52 MB	-rw-rw-r--	2019-05-10	15:57:00
aspectj-1.9.5.jar	8.16 MB	-rw-rw-r--	2019-11-28	15:25:03
aspectj-1.9.5.jar	16.58 MB	-rw-rw-r--	2019-11-28	15:24:53
aspectj-1.9.6-src.jar	8.22 MB	-rw-rw-r--	2020-07-22	16:00:00
aspectj-1.9.6.jar	16.73 MB	-rw-rw-r--	2020-07-22	15:58:48

</¿Cómo instalarlo?

Para instalarlo se debe escribir el siguiente comando en la carpeta donde quedo guardado, usualmente si no se tiene parametrizado queda en descargas

```
~/Downloads
```

```
java -jar aspectj-1.9.6.jar|
```

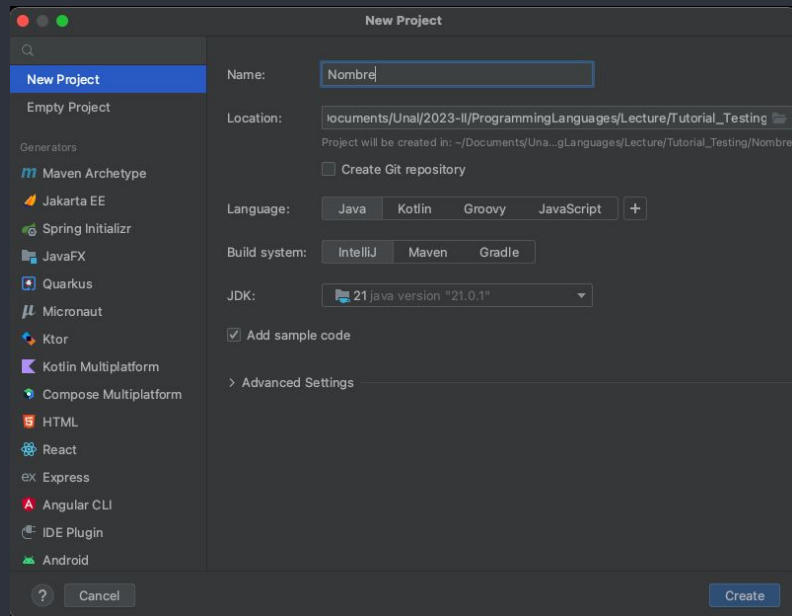
</¿Cómo instalarlo?

Deben quedar los siguientes documentos después de ejecutar la instalación dentro de la carpeta (Se hace por consola porque depende del sistema operativo)

```
~/aspectj1.9/lib (0.034s)  
ls  
aspectjrt.jar    aspectjtools.jar  aspectjweaver.jar
```

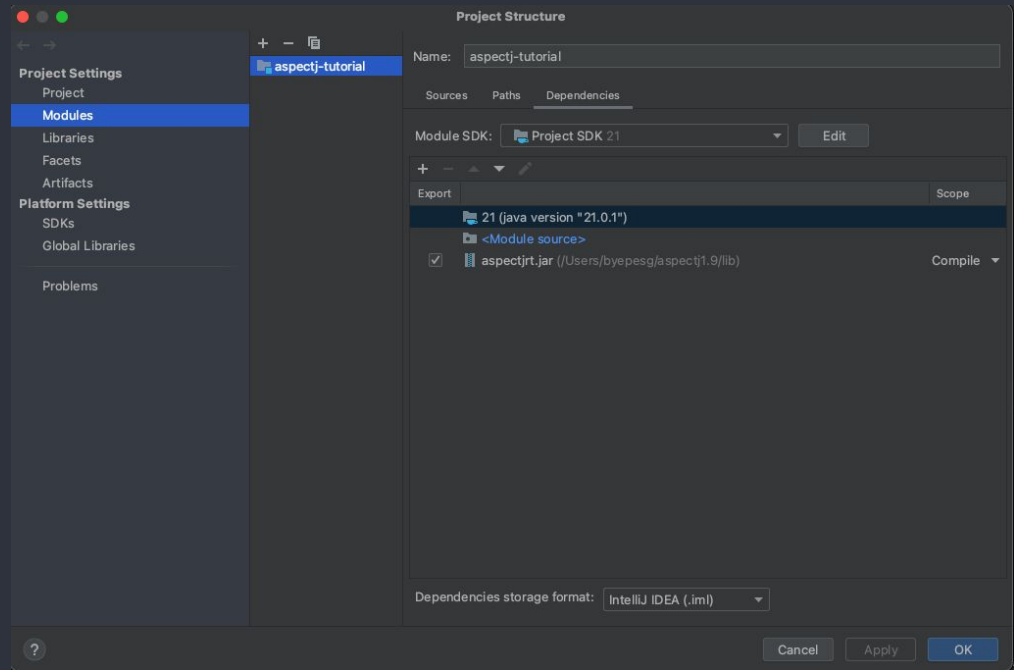

</¿Cómo instalarlo?

Crear un nuevo proyecto en IntelliJ:



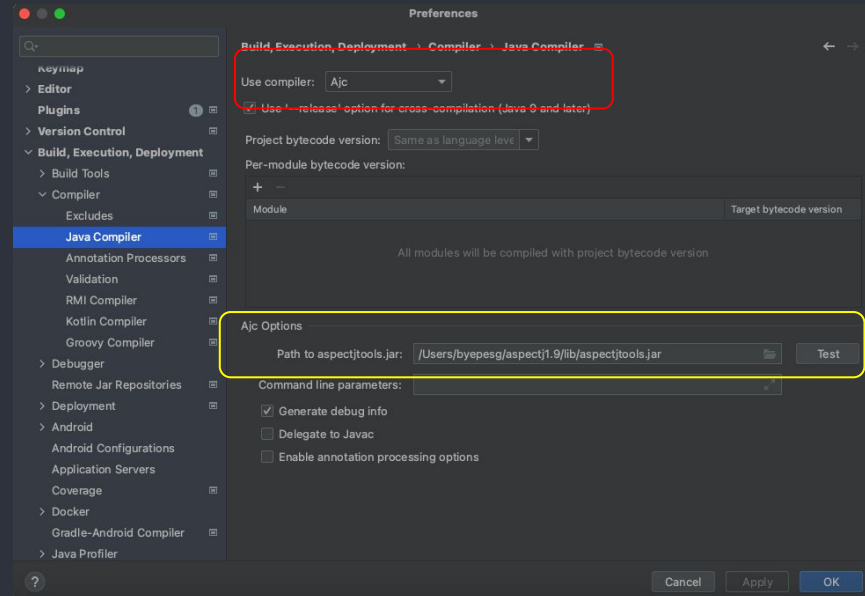
</¿Cómo instalarlo?

Importar el jar en “Module Settings”:



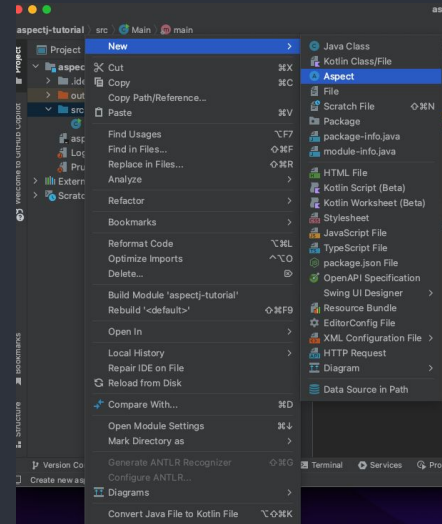
</¿Cómo instalarlo?

- Se debe ir a Build, Execution, Deployment > Compiler > Java Compiler
- Usar el compilador AJC como está demarcado en color rojo
- Se debe incluir la ruta como se encuentra demarcado en color amarillo



</¿Cómo instalarlo?

Finalmente crear un archivo correspondiente a “Aspect”:



</Ejemplos

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Ejemplos

- <https://github.com/jzmendezl/examples-aspectj>
- <https://github.com/jzmendezl/spring-aspectj>

</Quizziz

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Referencias

- Colyer, A., Clement, A., Harley, G., & Webster, M. (2004). Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the Eclipse AspectJ Development Tools. Addison Wesley Professional.
- Eclipse Foundation. (Año). AspectJ Documentation. Recuperado de <https://eclipse.dev/aspectj/sample-code.html#declares-exceptionSpelunking>
- Laddad, R. (2009). Aspectj in action: enterprise AOP with spring applications. Simon and Schuster.
- <https://ferestrepoca.github.io/paradigmas-de-programacion/poa/tutoriales/aspectJ/index.html#carouselExampleControls>