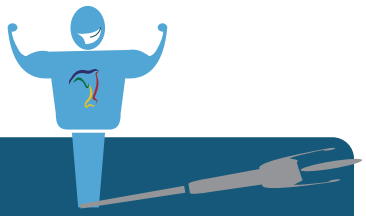
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

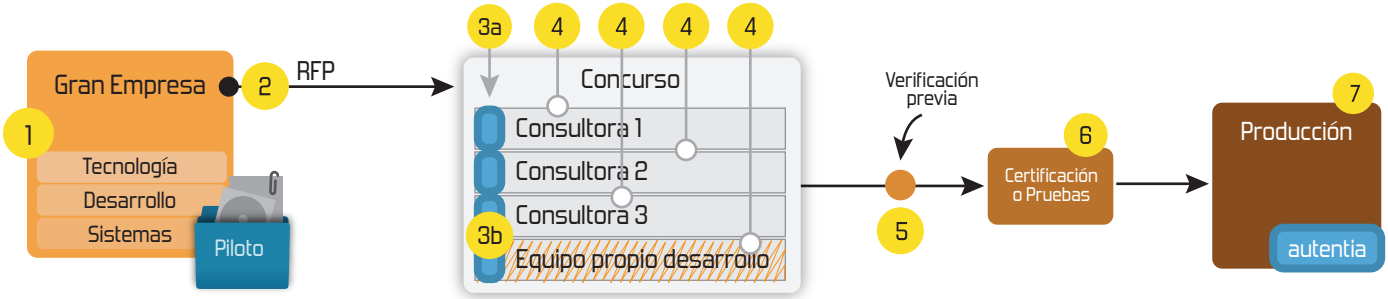
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

AdictosAlTrabajo
Terrakas 1x03 ¡¡Ya está en la web!!
terrakas.com
autentia Soporte a desarrollo informático
Hosting patrocinado por enredados
Entra en Adictos a través de
E-mail
Contraseña
Entrar
Deseo registrarme
Olvidé mi contraseña
Inicio Quiénes somos Formación Comparador de salarios Nuestro libro Más
» Estás en: Inicio Tutoriales Mejorando la calidad de nuestro código PHP

Daniel Hernandez del Peso

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2012-06-01 Tutorial visitado 1 veces Descargar en PDF

Mejorando la calidad del código PHP

Introducción

Desde Autentia llevamos tiempo intentando concienciar de la importancia de la calidad en el desarrollo de software. Hay que tener en cuenta que casi el 70% del esfuerzo invertido durante el tiempo de vida de un producto se destina a mantenimiento. Y el mantenimiento es significativamente más simple y rápido si se realiza sobre un buen código.

Por tanto, y teniendo en cuenta que todos tenemos vicios y manías adquiridos durante años de trabajo o durante el aprendizaje de un lenguaje, intentamos integrar en nuestro trabajo diario el uso de herramientas que nos faciliten la tarea de mantener unos mínimos de calidad. Algunos ejemplos de estas herramientas son PMD, Sonar o Jenkins (El producto antes conocido como Hudson)

Aunque muchas de estas herramientas están orientadas a lenguajes como Java, es posible encontrar herramientas similares que podamos aplicar a nuestros proyectos en PHP para tratar de mantener las cosas bajo control. Un ejemplo lo vimos con el tutorial de PHPUnit que publicamos hace unas fechas y ahora vamos a ver otras herramientas de análisis de código, como son PHP-Depend, PHPMD y PHPDCD

Antes de avanzar más, simplemente quiero deciros que el objetivo de este tutorial no es hacer una análisis pormenorizado y detalladísimo de las herramientas, sino una toma de contacto con las mismas, para darlas a conocer y conseguir que cualquiera pueda ponerse a "jugar" con ellas y sacarles todo el jugo

PHP Depend

PHP Depend se trata de una herramienta que realiza un análisis estático del código, tratando de calcular métricas de calidad del código como la complejidad ciclomática, el número de métodos en una clase, etc.

Instalación

La instalación, gracias a PEAR, es más que sencilla. Simplemente ejecutando las siguientes instrucciones en línea de comandos:

```
$ pear channel-discover pear.pdepend.org
$ pear install pdepend/PHP_Depend
```

Y tan fácil como esto... Ya estamos listos para empezar

Probando la herramienta

Una vez que lo tenemos instalado, vamos a empezar a probar la herramienta. Y para empezar por lo más sencillo, vamos a ver qué versión tenemos instalada, mediante el comando:

```
$ pdepend --version
```

Lo que genera la siguiente salida:

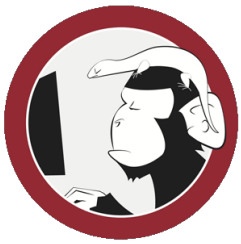
```
[illegible]$ pdepend --version
PHP_Depend 1.0.7 by Manuel Pichler
```

Lo siguiente que vamos a ver es el listado de opciones de que disponemos... Para ello ejecutamos el comando

```
$ pdepend --help
```

O ejecutar el comando "pdepend" sin parámetros. Lo que obtenemos es lo siguiente

Catálogo de servicios Autentia



Síguenos a través de:

Últimas Noticias

» Autentia colabora con la ONG Proyecto Ciclista Solidario

» Curso de Kanban Core en Madrid con Masa K. Maeda

» Failure demand

» Comentando el Libro: Lean StartUp (el método)

» VIII Autentia Cycling Day

Histórico de noticias

Últimos Tutoriales

» JQuery Waypoints: realizando acciones al llegar a un punto de la página con el scroll.

» Invocar a procedimientos almacenados de Oracle usando Spring JDBC

» Responsive CSS. Redimensionado de imagenes con Skeleton

» Primeros pasos con Cloud Foundry

» Subir ficheros al servidor con Spring MVC

Últimos Tutoriales del

```
PHP_Depend 1.0.7 by Manuel Pichler

Usage: pdepend [options] [logger] <dir[,dir[,...]]>

  --jdepend-chart=<file>    Generates a diagram of the analyzed packages.
  --jdepend-xml=<file>      Generates the package dependency log.

  --overview-pyramid=<file> Generates a chart with an Overview Pyramid for the
                            analyzed project.


  --summary-xml=<file>      Generates a xml log with all metrics.

  --coderank-mode=<*[,...]> Used CodeRank strategies. Comma separated list of
                            'inheritance'(default), 'property' and 'method'.
  --coverage-report=<file>  Clover style CodeCoverage report, as produced by
                            PHPUnit's --coverage-clover option.

  --configuration=<file>    Optional PHP_Depend configuration file.

  --suffix=<ext[,...]>      List of valid PHP file extensions.
  --ignore=<dir[,...]>      List of exclude directories.
  --exclude=<pkg[,...]>     List of exclude packages.

  --without-annotations     Do not parse doc comment annotations.

  --debug                   Prints debugging information.
  --help                    Print this help text.
  --version                 Print the current version.
  -d key[=value]            Sets a php.ini value.
```

Tenemos unas cuantas opciones, pero voy a centrarme sólo en las más útiles del producto, es decir, las que generan las métricas:

- jdepend-chart: Genera una imagen (SVG) en la que señala las dependencias entre clases, etc. de manera gráfica
- jdepend-xml: Devuelve un XML en el que se enumeran las clases concretas y abstractas por paquete
- overview-pyramid: La salida de esta opción es otra imagen en formato SVG en la que se presenta de manera compacta y agrupada la mayoría de métricas del proyecto
- summary-xml: Genera un fichero resumen con todas las métricas obtenidas del proyecto

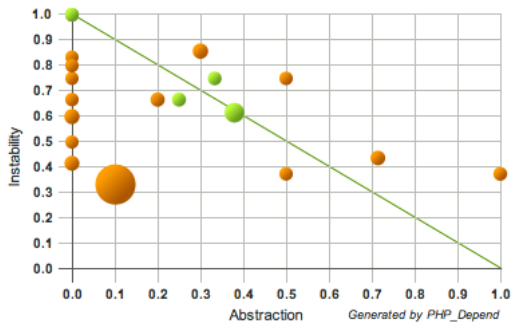
Es importante tener en cuenta, tal y como se explica en la página del producto, que podemos emplear todas las opciones anteriores en una sola invocación, de manera que obtengamos, en una sola pasada, todos los datos posibles

Por tanto, vamos a ejecutar PHP Depend sobre su propio código fuente:

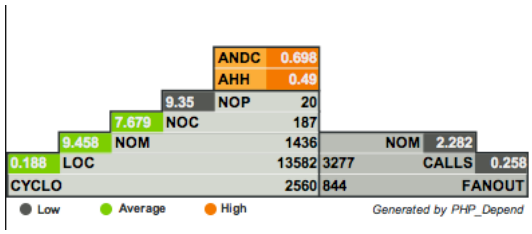
```
$ pdepend --overview-pyramid=RutaAInforme/pyramid.svg --jdepend-xml=RutaAInforme/report.xml --
summary-xml=RutaAInforme/summary.xml --jdepend-chart=RutaAInforme/jdependchart.svg
RutaA/PHP/Depend
```

Cuando acaba de ejecutarse, obtenemos los cuatro fichero que le hemos indicado. Particularmente, los XML que genera los encuentro incómodos, pues contienen demaisada información y XML no es un lenguaje especialmente cómodo para los seres humanos... pero es cierto que los datos que contiene pueden ayudarte a mejorar tu proyecto, así que lo ideal sería crearse unos XSL para convertirlo a otro lenguaje más legible como HTML. En cuanto a las imágenes, ofrecen información global del sistema, premitiendo hacerse una idea de cómo estamos. La verdad es que el mayor partido se le saca a la herramienta combinando los dos tipos de informe

A continuación os muestro el gráfico de dependencia generado. El eje vertical muestra el grado de acoplamiento enter paquetes, en base al acoplamiento aferente (Clases de otros paquetes que dependen del paquete actual) y el acoplamiento eferente (clases del paquete actual que dependene de clases de otros paquetes). Un valor de 0 en la vertical indica que el paquete no depende de ningún otro. Un valor de 1 en la vertical indica que ningún paquete depende del actual. En el eje horizontal se muestra la relación entre clases abstractas (o interfaces) y clases "concretas" (no abstractas). El valor 0 en este caso indica que todas las clases son concretas, mientras que el valor 1 indica que todas las clases del paquete son abstractas. Los paquetes que se sitúen en la línea que une (0,1) con (1,0) serán los que presenten mejor equilibrio entre estabilidad y grado de abstracción



La potencia de la visualización en pirámide está en su tamaño compacto, que nos permite ver de un vistazo el estado general del proyecto. Se divide en tres zonas: La superior contiene métricas relacionadas con el uso de la herencia, los valores mostrados a la derecha hablan del acoplamiento entre clases y en la parte izquierda se muestran medidas de la complejidad y el tamaño del proyecto. Por ejemplo:



Autor

» Invocar a procedimientos almacenados de Oracle usando Spring JDBC

» Subir ficheros al servidor con Spring MVC

» Subir una aplicación a BlackBerry App World

» Transformar aplicaciones Android en aplicaciones BlackBerry

» Migrar repositorios de Mercurial a Git

Últimas ofertas de empleo

2011-09-08
Comercial - Ventas - MADRID.

2011-09-03
Comercial - Ventas - VALENCIA.

2011-08-19
Comercial - Compras - ALICANTE.

2011-07-12
Otras Sin catalogar - MADRID.

2011-07-06
Otras Sin catalogar - LUGO.

En el gráfico se muestran un montón de acrónimos, y si consultamos los XML's tendremos muchos más. Afortunadamente, la página de PHP Depend ofrece una documentación bastante completa en este aspecto, en la que se muestra el listado de métricas, con su acrónimo, una descripción y sobre qué tipos de elemento se aplican (sobre métodos, clases, etc.) y una descripción detallada de cada tipo de gráfica, con su interpretación. En general, recomiendo leer esta documentación puesto que, además de explicar el proceso de instalación, la interpretación de las métricas, etc. aporta enlaces a documentos en los que se explica la importancia de algunas de estas métricas

Esta herramienta resuelve parte del problema, pues nos permite comprobar numéricamente ciertos parámetros relacionados con la calidad, pero no nos ha resuelto el problema completo, puesto que no nos ha localizado código muerto, duplicado, etc.

Por ello, vamos a probar otras herramientas...

PHPMD (PHP Mess Detector)

Esta herramienta es, según la propia página del proyecto, un "spin off" de PHP Depend, con una funcionalidad parecida a PMD para Java. Es decir, una herramienta automática para detectar código no utilizado, nombres de variables poco apropiados, etc. en base a un conjunto de reglas

Instalación

El proceso de instalación viene descrito en la página, y de nuevo es sumamente sencillo gracias a PEAR...

```
$ pear channel-discover pear.phpmd.org
$ pear channel-discover pear.pdepend.org
$ pear install --alldeps phpmd/PHP_PMD
```

Ya estamos listos... Ahora ejecutamos en línea de comandos:

```
$ phpmd
```

Para obtener la explicación de uso del script

```
Mandatory arguments:
1) A php source code filename or directory. Can be a comma-separated string
2) A report format
3) A ruleset filename or a comma-separated string of rulesetfilenames

Optional arguments that may be put after the mandatory arguments:
--minimumpriority: rule priority threshold; rules with lower priority than this will not be used
--reportfile: send report output to a file; default to STDOUT
--suffixes: comma-separated string of valid source code filename extensions
--exclude: comma-separated string of patterns that are used to ignore directories
--strict: also report those nodes with a @SuppressWarnings annotation
```

En este caso, la única opción interesante para esta prueba de concepto es la posibilidad de indicar el fichero de salida (en lugar de obtener la salida por consola). Además, entre los parámetros obligatorios del script se le debe indicar el conjunto de reglas que se quiere aplicar para la validación del código. PHPMD incluye cuatro juegos de reglas:

- Código no utilizado: Localiza métodos no invocados nunca, pero sólo localmente (métodos y atributos privados, variables locales, parámetros de llamada a un método)
- Tamaño del código: Complejidad ciclomática, longitud de métodos o clases, etc. Es similar a PHP Dpend, solo que aquí muestra avisos cuando algún valor es demasiado alto
- Diseño: Comprueba que no se empleen instrucciones estilo "goto", el grado de acoplamiento, la profundidad de la herencia...
- Nombrado: indica si se han cometido violaciones por usar nombres de variable demasiado cortos o largos, y otras convenciones de nombrado

Además, y aquí es donde está toda la potencia de PHPMD, podemos construir nuestras propias reglas y conjuntos de reglas, de manera que podamos personalizar el resultado de la salida a las políticas de codificación de nuestra empresa

Y ahora que hemos visto los distintos juegos de reglas que incorpora el producto de serie, vamos a usarlo para localizar código no utilizado sobre el código de PHP Depend

```
$ phpmd /RutaA/PHP/Depend html unusedcode --reportfile salida.html
```

Y obtenemos como resultado un HTML con las violaciones encontradas

PHPMD report

Problems found

#	File	Line	Problem
1	/Depend/Code/Exceptions/SourceNotFoundException.php	72	Avoid unused parameters such as '$owner'.
2	/Depend/Code/NodeIterator.php	211	Avoid unused parameters such as '$offset'.
3	/Depend/Code/NodeIterator.php	211	Avoid unused parameters such as '$value'.
4	/Depend/Code/NodeIterator.php	225	Avoid unused parameters such as '$offset'.
5	/Depend/Parser/VersionAllParser.php	301	Avoid unused local variables such as '$match'.
6	/Depend/Util/Configuration.php	120	Avoid unused parameters such as '$value'.
7	/Depend/Visitor/AbstractListener.php	290	Avoid unused parameters such as '$node'.
8	/Depend/Visitor/AbstractListener.php	302	Avoid unused parameters such as '$node'.

Como podéis ver, se trata de una tabla muy simple, con el fichero y la línea en la que se localiza la violación, e incluso un enlace a una desripción más detallada del problema localizado

Ya hemos dado un nuevo paso en la limpieza de nuestro código, y sólo hemos aplicado uno de los conjuntos de reglas... Aún así, no hemos completado la tarea, pues no hemos sido capaces de localizar todas las funciones no utilizadas en nuestro proyecto. Para esto, vamos a utilizar otra herramienta, PHPDCD

PHP DCD (PHP Dead Code Detector)

Instalación

Para saber como se instala esta funcionalidad, de nuevo acudimos a la página del proyecto. Y de nuevo, comprobamos

que la instalación es sencillísima gracias a PEAR. Ejecutamos dos comandos desde el terminal...

```
$ pear config-set auto_discover 1
$ pear install pear.phpunit.de/phpdcd
```

... Y listo :)

Ahora solo queda ejecutar la herramienta. Como hemos hecho anteriormente, vamos antes de nada a ejecutar el comando sin parámetros, para forzar el fallo y que nos indique la sintaxis correcta. Ejecutamos por tanto:

```
$ phpdcd
```

Y obtenemos la siguiente salida:

```
phpdcd 0.9.3 by Sebastian Bergmann.
Usage: phpdcd [switches] <directory|file> ...

  --recursive          Report code as dead if it is only called by dead code.

  --exclude <dir>      Exclude <dir> from code analysis.
  --suffixes <suffix>  A comma-separated list of file suffixes to check.

  --help               Prints this usage information.
  --version            Prints the version and exits.

  --verbose            Print progress bar.
```

La única opción interesante en este momento para mí es la que indica que marque un código como "muerto" si es llamados sólo por código "muerto". En cambio, echo de menos una opción para volcar la salida a fichero

Por tanto, vamos a hacer una ejecución para probar, dejando que vuelque los datos por pantalla. Para ello, ejecuto el comando:

```
$ phpdcd --recursive /RutaA/PHP/Depend
```

Y obtengo la siguiente salida

```
phpdcd 0.9.3 by Sebastian Bergmann.

  - PHP_Depend_Autoload::autoload()
    declared in /Autoload.php:81

  - PHP_Depend_Autoload::register()
    declared in /Autoload.php:68

Time: 2 seconds, Memory: 34.00Mb
```

De todas formas, antes de lanzarnos a borrar este código muerto (suponiendo que estuvieramos revisando nuestro propio código, claro), debemos tener en cuenta las limitaciones de la herramienta, tal y como nos las indica el propio autor en su página, porque puede tratarse de "falsos positivos". Además, debemos tener en cuenta que esos métodos pueden ser métodos públicos de un API o herramienta que estemos generando y que por tanto están pensados para ser llamados por terceros, no por nosotros mismos. Así que un poco de cuidado y de cabeza ;)

Conclusiones

A lo largo de este tutorial hemos visto tres productos que permiten "medir" la calidad de nuestro código PHP y mejorarla en la medida de lo posible, al igual que lo hacemos en nuestros proyectos en Java.

Como ya he dicho al comienzo, sólo pretendo presentaros los productos, así que no he profundizado demasiado en ellos, eso os lo dejo a vosotros ;) . Pero recordad que las herramientas son eso, herramientas, y que detrás de su uso debe haber una cabeza pensante, decidiendo cuando es necesario usar una u otra o cómo interpretar los resultados. Y recordad también que lo mejor que os puede pasar no es que las herramientas os den una información completísima de todos vuestros fallos, sino que no tenga nada que deciros :)

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «

Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

IMPULSA Impulsores Comunidad ¿Ayuda?

---
sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by karmacracy

Copyright 2003-2012 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS XML RSS XML ATOM