unidad didáctica 05

Actividades

1. Monolog

Actividad 501

Crea un nuevo proyecto con *Composer* llamado `Monologos`:

- Incluye como librería la última versión de Monolog.
- Crea la clase `Dwes\Monologos\HolaMonolog`.
- Define una propiedad privada nombrada `miLog` para guardar el log.
- Define en el constructor un `RotatingFileHandler` que escriba en la carpeta `logs` del proyecto, y que almacene los mensajes a partir de *debug*.
- Crea los métodos `saludar` y `despedir` que hagan un log de tipo *info* con la acción correspondiente.

Actividad 502

Siguiendo con el proyecto `Monologos`:

- Crea un archivo llamado `inicio.php` que permita probar `HolaMonolog`.
- Comprueba que los mensajes aparecen en el *log*.
- Cambia el nivel para que el manejador solo muestre los mensajes a partir de *warning*.
- Vuelve a ejectuar `inicio.php` y comprueba el archivo de log.

Actividad 503

Modifica la clase `HolaMonolog`:

- En el constructor, añade a la pila un manejador que escriba a la salida de error conjunto al procesador de introspección, mostrando mensajes desde el nivel *debug*.
- Añade una propiedad denominada `hora`, la cual se inicializa únicamente como parámetro del constructor. Si la `hora` es inferior a 0 o mayor de 24, debe escribir un log de *warning* con un mensaje apropiado.
- Modifica los métodos `saludar` y `despedir` para hacerlo acorde a la propiedad `hora` (buenos días, buenas tardes, hasta mañana, etc...)

2. proyecto Videoclub IV

Actividad 511

Como ya tenemos *Composer* instalado:

- Inicialízalo dentro de tu proyecto *Videoclub.*
- Incluye *Monolog* y *PhpUnit*, cada una en su lugar adecuado.
- Añade el autoload al archivo `composer.json`, y haz los cambios necesarios en las clases para utilizar el *autoload* de *Composer*.
- Sube los cambios a *GitHub* y crea la etiqueta `v0.511`.

Actividad 512

Modifica la clase `Cliente` para introducir un `Logger` de Monolog.

- Añade el log como una propiedad de la clase e inicialízalo en el constructor, con el nombre del canal `VideoclubLogger`.
- Se debe almacenar en `logs/videoclub.log` mostrando todos los mensajes desde *debug*.
- Antes de lanzar cualquier excepción, debe escribir un log de tipo *warning*.
- Sustituir los `echo` que haya en el código, que ahora pasarán por el log con el nivel info, a excepción del método `muestraResumen` que seguirá haciendo `echo`.

Actividad 513

Vuelve a hacer lo mismo que en el ejercicio anterior, pero ahora con la clase `Videoclub`. Además:

- Siempre que se llame a un método del log, se le pasará como segundo parámetro la información que dispongamos.
- Ejecuta el archivo de prueba y comprueba que el log se rellena correctamente.

Actividad 514

Vamos a refactorizar el código común de inicialización de *Monolog* que tenemos repetidos en los constructores a una factoría de *Monolog*, la cual colocaremos en `\Dwes\Videoclub\Util\LogFactory`. Comprueba que sigue funcionando correctamente.

Actividad 515

Modifica la factoría para que devuelva `LogInterface` y comprueba que sigue funcionando. Sube los cambios a GitHub con la etiqueta `v0.515`.

3. phpDocumentor

Actividad 521

Comprueba que en el contenedor de Docker funciona *phpDocumentor*. Ejecuta phpdoc sobre tu proyecto *Monolog* y comprueba el api que se crea. Comenta tanto la clase como los métodos, y posteriormente, vuelve a ejecutar phpdoc.

Actividad 522

Documenta el proyecto *Videoclub*, y genera la documentación. Empieza por las clases de `Soporte` y sus hijos. Comprueba el resultado. Luego sigue con `Cliente` y finalmente `Videoclub`.

4. Web Scraping

Actividad 531

A partir de los datos de http://www.seleccionbaloncesto.es, calcula la altura y edad media del equipo de baloncesto masculino. Observa que tienes los datos dentro de una tabla debajo de las noticias.

Actividad 532

Volviendo al Videoclub, en `Soporte` añade una propiedad llamada `metacritic` para almacenar la URL de cada soporte. A continuación, modifica los métodos `incluirXXX` de `Videoclub` para que admitan como primer parámetro dicha URL. Tras ello, modifica el fichero `inicio3.php` para pasarle la URL de cada soporte (para ello deberás consultarlos en Metacritic haciendo búsquedas manuales). Por ejemplo, en el caso de la película Cazafantasmas, su URL es https://www.metacritic.com/movie/ghostbusters.

Actividad 533

Finalmente, añade un método abstracto en `Soporte` llamado `getPuntuacion`, que haciendo uso de *Web Scraping* se conecte a Metacritic y obtenga su puntuación. Modifica `inicio3.php` para obtener todos los alquileres de un cliente mediante `getAlquileres() : array`, y para cada uno de ellos, además del título, muestra su puntuación.

5. phpUnit

Actividad 541

A partir de la clase `HolaMonolog`, modifica los métodos para que además de escribir en en log, devuelvan el saludo como una cadena.

Crea la clase `HolaMonologTest` y añade diferentes casos de prueba para comprobar que los saludos y despedidas son acordes a la hora con la que se crea la clase.

Actividad 542

Vamos a simular *Test Driven Development*. Queremos que nuestra aplicación almacene los últimos tres saludos que ha realizado. Para ello:

- Crea las prueba necesarias (invoca al método `saludar` varias veces y llama al método que te devuelva los saludos almacenados)
- Implementa el código para pasar las pruebas
- Refactoriza el código

Actividad 543

Crea una nueva prueba que utilice proveedores de datos para comprobar esta última funcionalidad, pasándole:

- Un saludo.
- Tres saludos.
- Cuatro saludos.

Actividad 544

¿Recuerdas que si la hora es negativa o superior a 24 escribíamos en el log un *warning*? Ahora debe lanzar una excepción de tipo `InvalidArgumentException` (como la excepción forma para de PHP, hay que poner su FQN: `\InvalidArgumentException`). Vuelve a aplicar TDD y completa tus casos de prueba.

Actividad 545

Comenta la última prueba realizada (la comprobación de las excepciones) y realiza un informe de cobertura de pruebas. Analiza los resultados obtenidos. Elimina los últimos comentarios sobre la última prueba y vuelve a generar y analizar el informe de cobertura.

6. proyecto Videoclub V

El objetivo de los siguientes ejercicios es conseguir de manera incremental una cobertura de pruebas superior al 95%.

Actividad 551

Crea pruebas dentro de la carpeta `tests` para las clases `Soporte`, `CintaVideo`, `Dvd` y `Juego`. Recuerda respetar el espacio de nombres. Los métodos `muestraResumen`, tras hacer echo de los mensajes, deben devolver una cadena con el propio mensaje.

Actividad 552

Crea pruebas para la clase `Cliente`, aprovechando todo el código que teníamos para comprobar la funcionalidad. Utiliza proveedores de datos para añadir conjuntos de datos mayores que los empleados. Comprueba que funciona con diferentes cupos, que al intentar alquilar un soporte marcado como ya alquilado debe lanzar una excepción, que no coincidan los ids de los soportes, etc...

Actividad 553

Crea las pruebas para la clase `Videoclub`. Ten en cuenta los últimos métodos añadidos que permitían alquilar y devolver soportes, tanto de manera individual como mediante un array.

Actividad 554

Crea el informe de cobertura. Una vez creado, analiza los datos de cobertura (>= 90%) y comprueba el valor de CRAP, de manera que siempre sea <= 5. En caso de no cumplirse, crea nuevos casos de prueba y/o refactoriza el código de tu aplicación.

Sube los cambios a GitHub con la etiqueta `v0.554`.

7. ampliación

Actividad 561

Queremos que en `Videoclub`, cuando un cliente no existe (tanto al alquilar como al devolver) se lance una nueva excepción: `ClienteNoExisteException`. Además, dado el número creciente de excepciones, queremos mover las excepciones al namespace `Dwes\Videoclub\Exception`.

Siguiendo TDD, primero crea las pruebas, y luego modifica el código de aplicación.

Vuelve a generar el informe de cobertura y comprueba la calidad de nuestras pruebas.

Actividad 562

¿Nadie se ha dado cuenta que en los Dvd no estamos almacenando su duración? Haz todos los cambios necesarios, primero en las pruebas y luego en el código.

Actividad 563

Tras años luchando contra la tecnología, decidimos introducir los Blu-ray en nuestra empresa. Hemos decido que `Bluray` herede de `Soporte`. Además del `título` y la `duracion`, nos interesa almacenar si `es4k`. Haz todos los cambios necesarios, primero en las pruebas y luego en el código.

Sube los cambios a GitHub con la etiqueta `v0.563`.