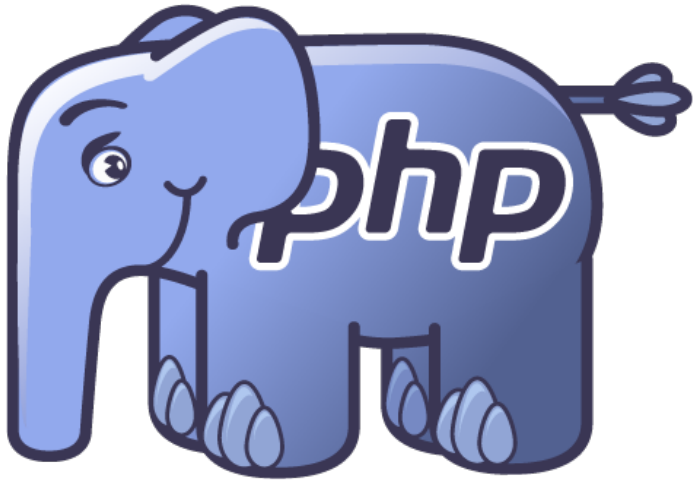
PHP Avanzado

Federico Ferreri

Agosto 2015



INSIDETECH

Unidad 5
Testing

- Por que testear?
- Cuando testear?
- Que testear?
- Integrated / PHPUnit

Por que testear?

1. Para estar seguros de que nuestro código funciona y que va a seguir funcionando luego de hacer cambios en el.
2. Para estar seguros de que el código que se sube a producción reúne los requisitos básicos de calidad para considerarlo código "apto producción".
3. Para documentar el código (TDD).

- Aunque no lo sea, el proceso de testing es algo que debería ser natural para todo programador.
- No todos los equipos cuentan con recursos de testing.
- El testing debería ser considerado como una inversión a futuro y no como un gasto en el proceso de desarrollo.
- Los tests son parte del código, no un producto de marketing para decorarlo.

Cuando testear?

- Antes, durante y después. O sea, siempre.
- Antes
  - Comenzar el proyecto con la idea de que va a ser testeado.
  - Instalar las herramientas necesarias para el testing
- Durante
  - Crear y ejecutar tests para cada pieza del software.
  - Cada clase, cada método, cada nueva página del proyecto web debería ser testeado mientras se lo desarrolla. De esa manera se pueden detectar errores en forma temprana, corregirlos y evitar que se propaguen bugs.
- Después
  - Todo fix que sea hecho debe ser incorporado a los casos de testeo.
  - Los tests aseguran que cambios hechos al código no afecten a la funcionalidad existente.

Que testear?

- A bajo nivel, se debería testear cada componente del software: clases, métodos y funciones para asegurarnos que funcionen en aislamiento. Eso es, a groso modo, "Unit Testing".
- A mas alto nivel, se debería testear que las clases, métodos y funciones se comporten bien al funcionar en conjunto. Para ello se utilizan los tests llamados "de integración".
- A mucho mas alto nivel todavía, se debería testear que la aplicación haga lo que el cliente especificó, desde como luce la UI hasta el resultado que devuelve una búsqueda. Para ello se utilizan los tests llamados "funcionales". Estos tests deberían ser definidos por el project manager y su ejecución quedar a cargo del equipo de QA.

- Las forma mas popular de testing en PHP son las de Unit e Integration Testing y el producto mas usado para hacerlo es PHPUnit o derivados del mismo.
- Hay otros frameworks de testing como PHPSpec, CodeCeption y Behat entre otros.

PHPUnit / Integrated

- "Integrated" es un componente de testing que usa PHPUnit como base.
- Es muy sencillo de usar y los tests son muy fáciles de leer y probar.
- Se lo utiliza para hacer tests de integración, que son un paso intermedio entre los tests unitarios y los test funcionales.
- Se instala dentro del proyecto usando composer
  - `composer require phpunit/phpunit –dev`
  - `composer require laracasts/integrated –dev`
- Se lo incluye en los files de testing instanciando alguna de las clases incluidas:
  - `class MiTest extends Laracasts\Integrated\Extensions\Goutte {}`
- Cada método en estas clases marcado con el docblock @test son una prueba que será ejecutada por PHPPunit al ejecutar
  - `./vendor/bin/phpunit tests/`
- La documentación de Integrated se encuentra en GitHub
  - `https://github.com/laracasts/Integrated/wiki`