Proves i depuració PHP

Introducció

Context i importància

En el desenvolupament de programari, el codi no només ha de funcionar, sinó que ha de ser fiable, segur i eficient.

Errors i vulnerabilitats poden tenir un impacte significatiu, com ara:

- Pèrdua de dades
- Experiència d'usuari deficient
- Riscos de seguretat.

És aquí on les proves i la depuració esdevenen pilars essencials per garantir la qualitat del programari.

Introducció

Objectius de la sessió

- Comprendre els tipus de proves:
  - Conèixer les diferents tècniques per assegurar que el codi funciona correctament en totes les situacions possibles.

- Explorar eines per a proves i depuració:
  - Familiaritzar-se amb eines populars com PHPUnit, Xdebug i altres.

- Aprendre les millors pràctiques de proves:
  - Aplicar un enfocament estructurat i sistemàtic per maximitzar l'eficiència i l'eficàcia de les proves.

Introducció

Objectius de la sessió

- Desenvolupar habilitats per depurar errors:
  - Identificar i corregir problemes utilitzant tècniques i eines existents.

- Realitzar exercicis pràctics:
  - Veure exemples reals de proves unitàries i depuració pas a pas.

Introducció

Conceptes clau

- Proves:
  - Procés que assegura que un programa funciona segons les especificacions.

  - Tipus de proves:

    - ➢ **Unitàries**: Validació de funcions o mòduls individuals.
    - ➢ **Integració**: Verificació de la interacció entre components.
    - ➢ **Seguretat**: Identificació de vulnerabilitats.
    - ➢ **Rendiment**: Mesura de la resposta del sistema sota càrrega.

Introducció

Conceptes clau

- Depuració:
  - Localització i correcció d'errors en el codi.

  - Estratègies comunes:
    - ➢ Aïllar el problema.
    - ➢ Utilitzar logs per entendre el comportament.
    - ➢ Depurar pas a pas amb eines com Xdebug.

- Eines de proves i depuració:
  - Proves automàtiques: PHPUnit, Selenium.
  - Proves de seguretat: OWASP ZAP .
  - Depuració: Xdebug.

Desenvolupament del tipus de prova

Tipus de prova

1. Proves Unitàries

- Definició:
  - Són el primer nivell de proves en el desenvolupament de programari.
  - Estan dissenyades per provar unitats de codi de manera individual i aïllada.
  - Aquestes unitats poden ser funcions, mètodes, o petites parts del programa que es poden provar independentment.

- Objectiu:
  - Garantir que cada unitat de codi funciona segons les especificacions.
  - Detectar errors en fases inicials del desenvolupament.
  - Millorar la confiança en els canvis de codi, ja que qualsevol disfunció és detectada ràpidament.

Tipus de prova

1. Proves Unitàries

- Funció a provar:

```
function suma($a, $b) {
    return $a + $b;
}
```

- Prova unitària:

```
public function testSuma() {
    $this->assertEquals(5, suma(2, 3)); // Comprova que suma(2, 3) retorna 5
    $this->assertEquals(0, suma(0, 0)); // Comprova que suma(0, 0) retorna 0
    $this->assertEquals(-1, suma(-3, 2)); // Comprova que suma(-3, 2) retorna -1
```

Tipus de prova

1. Proves Unitàries

- Resultat esperat:

```
Runtime:       PHP 8.2.12

.                                                                   1 / 1 (100%)

Time: 00:00.027, Memory: 8.00 MB

OK (1 test, 3 assertions)
```

Tipus de prova

2. Proves d'integració

- Definició:
  - Validen que diferents components o mòduls del sistema treballen correctament junts.

- Objectiu:
  - Assegurar que les interaccions entre components no generin errors.

- Exemple en una aplicació web (PHP):
  - Provar si un formulari d'inici de sessió interactua correctament amb la base de dades i retorna el resultat esperat.

Tipus de prova

3. Proves de seguretat

- Definició:
  - Són proves dissenyades per detectar vulnerabilitats que podrien ser explotades per atacants.

- Objectiu:
  - Assegurar que el sistema està protegit contra atacs com SQL Injection, Cross-Site Scripting (XSS) o robatori de sessions.

- Exemple d'una prova de seguretat per SQL Injection:
  - Prova automàtica per comprovar si la consulta preparada protegeix contra un SQL Injection.

Tipus de prova

4. Proves de rendiment

- Definició:
  - Avaluen com el sistema respon sota càrrega, com per exemple un gran nombre d'usuaris simultanis.

- Objectiu:
  - Mesurar el temps de resposta, la càrrega màxima suportada i l'ús de recursos.

- Exemple amb una eina com Apache JMeter:
  - Simular 100 usuaris accedint al servidor al mateix temps.
  - Mesurar el temps de resposta de les peticions.

Tipus de prova

5. Proves d’acceptació

- Definició:
  - Comproven si el sistema compleix els requisits funcionals des del punt de vista de l'usuari final.

- Objectiu:
  - Validar que el sistema compleix les expectatives del client o usuari.

- Exemple d'escenari pràctic:
  - Verificar que un usuari pot completar correctament una compra en una botiga online:
    - ➢ Inicia sessió.
    - ➢ Selecciona productes.
    - ➢ Finalitza la compra amb èxit.

Tipus de prova

6. Proves de regressió

- Definició:
  - Asseguren que funcionalitats ja existents continuen funcionant després d'actualitzacions o canvis.

- Objectiu:
  - Evitar que noves funcionalitats introdueixin errors en funcionalitats antigues.

- Exemple amb PHPUnit:
  - Executar totes les proves unitàries i d'integració automàticament després d'una actualització.

```
public function testSuite() {
    $this->testSuma();
    $this->testLoginIntegracio();
    $this->testSeguretatSQL();}
```

Eines per a realitzar proves

Eines per a proves unitàries en PHP

PHPUnit

PHPUnit és un framework per a proves unitàries en PHP. És àmpliament utilitzat per la seva facilitat d'ús, integració amb altres eines i funcionalitats avançades.

Característiques de PHPUnit:

- Permet definir proves de manera estructurada.
- Ofereix mecanismes per verificar resultats esperats (assertions).
- Genera informes detallats sobre l'estat de les proves.
- Permet executar proves automàticament en conjunts (suites).

Eines per a proves unitàries en PHP

PHPUnit

Estructura bàsica de una prova unitària:

- Preparació: Configurar el context necessari per executar la prova.
- Execució: Invocar la unitat de codi que es vol provar.
- Verificació: Comparar el resultat obtingut amb el resultat esperat.

```
class CalculatorTest extends PHPUnit\Framework\TestCase {
    public function testSuma() {
    // Preparació
    $a = 2; $b = 3;
    // Execució
    $resultat = suma($a, $b);
    // Verificació
    $this->assertEquals(5, $resultat, "La suma de 2 i 3 hauria de ser 5.");
    }}
```

Eines per a proves unitàries en PHP

PHPUnit

- Instal·lació de PHPUnit (si encara no està instal·lat):

  composer require --dev phpunit/phpunit

- Estructura del Projecte:

  projecte/
  ├── functions.php (Codi principal amb les funcions)
  ├── tests/
  │ └── FunctionsTest.php (Proves unitàries)
  ├── composer.json (Fitxer de dependències)

- Executar Proves:

  vendor\bin\phpunit tests\

Eines per a proves d’integració PHP

Selenium

Selenium permet simular interaccions d'usuari amb aplicacions web i pot ser molt útil per provar formularis com el d'accés al teu sistema.

Funcionalitats principals:

- Simular accions d'usuari (clics, entrades de text, navegació).
- Automatitzar proves completes, des del frontend fins al backend.

Exemple d’ús:

- Validar un formulari de login amb PHP (explicat en detall en sessions anteriors).

Eines per a proves de seguretat PHP

OWASP ZAP

- Descripció: Una eina gratuïta i de codi obert que detecta vulnerabilitats de seguretat en aplicacions PHP.

- Funcionalitats principals:
  - Escanejar aplicacions PHP per a vulnerabilitats comuns, com ara SQL Injection o XSS.
  - Simular atacs per comprovar la robustesa de l'aplicació.
- Exemple d'ús:
  - Analitzar un lloc desenvolupat en PHP i identificar si les consultes SQL estan protegides adequadament.

Eines per a proves de rendiment PHP

Apache JMeter

- Descripció: JMeter simula càrregues massives per comprovar el rendiment i la capacitat de resposta d'aplicacions PHP.

- Funcionalitats principals:
  - Mesurar temps de resposta de scripts PHP.
  - Simular diversos usuaris simultanis interactuant amb l'aplicació.

- Exemple d'ús:
  - Provar un endpoint d'una API PHP amb 100 usuaris simultanis i verificar que la resposta és inferior a 2 segons.

Procés de depuració d’errors en PHP

Identificació de l'error

Objectiu:

- Determinar la presència d'un error al sistema
- Localitzar la situació on el codi no es comporta com s'espera.

Mètodes per identificar errors:

- Missatges d'error:
  - Analitza els missatges generats pel sistema.
  - Exemples comuns en PHP són advertències de sintaxi, errors de tipus o missatges de funcions no definides.

- Logs: Revisa fitxers de registre per obtenir més detalls.
  - Exemple: Utilitza error_log("Missatge personalitzat", 3, "log_usuari.log");.

- Eines de depuració: Utilitza depuradors integrats, com Xdebug per a PHP, per realitzar una anàlisi en temps real.

Aillament de l’error

Un cop identificat l'error, cal determinar exactament quin fragment de codi el causa.

Estratègies per aïllar l'error:

- Divideix i conquerirà: Comenta blocs de codi per reduir l'àrea d'investigació.
- Prova amb dades controlades: Substitueix entrades variables per valors constants per eliminar incerteses.
- Punts de verificació: Afegeix punts de control al codi amb funcions com **var_dump** o **print_r** per verificar l'estat de les variables.

Resolució de l’error

Quan l'error s'ha identificat i aïllat, és hora de corregir-lo modificant el codi defectuós.

Bones pràctiques en la resolució:

- Entén el problema abans de modificar-lo: No facis canvis fins a comprendre totalment què provoca l'error.
- Documenta el canvi: Anota el problema i la solució aplicada per facilitar futures revisions.
- Refactoritza si cal: Si l'error prové d'una implementació poc òptima, considera reescriure el codi per millorar-lo.

Verificació

Un cop realitzada la correcció, és fonamental assegurar-se que l'error ha estat eliminat i que no s'han introduït altres problemes.

Estratègies per verificar:

- Reexecuta les proves afectades: Prova la funcionalitat corregida amb els mateixos escenaris que causaven l'error.
- Executa proves addicionals: Verifica que altres funcionalitats no han estat afectades per la correcció.
- Valida amb dades reals: Prova amb entrades similars a les que es trobaran en producció.

Eines per a la depuració

Depuradors integrats

Els depuradors integrats permeten analitzar i diagnosticar el comportament del codi pas a pas, ajudant a localitzar i corregir errors.

Exemples d'eines integrades:

- Xdebug (per a PHP):
  - Permet fer seguiment línia a línia.
  - Mostra l'estat de les variables en temps real.
- Debuggers del navegador:
  - Chrome DevTools:
    - Permet inspeccionar el DOM, CSS i JavaScript.
    - Inclou punts d'interrupció (breakpoints) per pausar l'execució i inspeccionar variables.
  - Firefox DevTools:
    - Similars a les eines de Chrome, amb funcionalitats avançades per depurar rendiment i xarxa.

Registre i Logging

El registre és fonamental per entendre què està passant dins d'una aplicació, especialment en entorns on no és viable utilitzar un depurador en temps real.

Eines i mètodes de registre:

- Utilitzar error_log() en PHP
- Monolog (framework avançat de registre per a PHP):