UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

POO (2)

Constructor

Constructor

Permite ejecutar un código cuando se llama a new…

```php
class Persona {
   public $nombre;

   function __construct() {
       $this->nombre = "Valor inicial";
   }
}
```

Utilizado para inicializar propiedades

Excepciones

Excepciones

Nos permite definir la forma en la cual podemos manejar los errores que se producen en nuestra aplicación

Excepciones

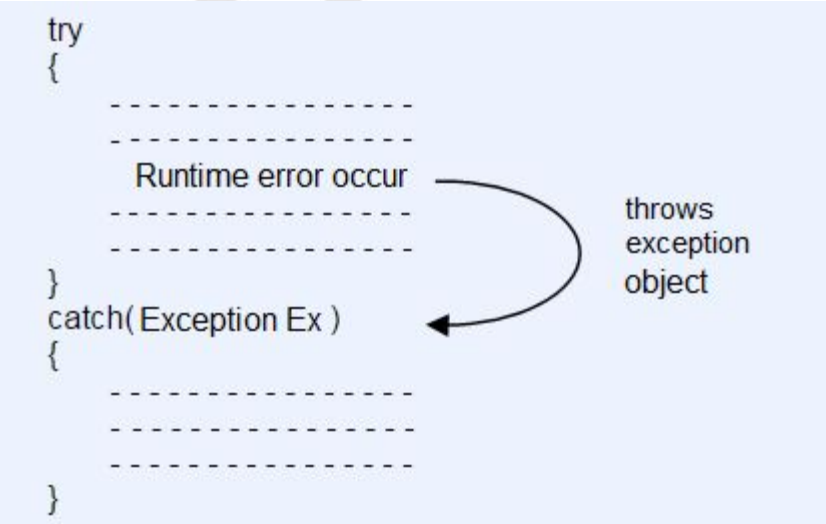
```
try{
```

Exceptions...

Gotta catch 'em all!

```
}catch( Exception ){
    //Do nothing
}
```

Excepciones



Excepciones

Dentro del try {} ponemos todo el código sin manejo de errores

Dentro de catch() {} ponemos todo el código de manejo de errores

Bloque catch

Method	Description
`getMessage()`	Returns the exception's message.
`getCode()`	Returns the exception's error code.
`getLine()`	Returns the line number from which the exception was thrown.
`getFile()`	Returns the name of the file from which the exception was thrown.,
`getTrace()`	Returns the exception's stack trace (runtime information about the context in which the error occurred) as an array which includes the name of the file that threw the exception, the line number, the function from which it was thrown, and the arguments that the function received.
`getTraceAsString()`	Returns same information `getTrace()` except the information is returned as a string rather than an array.

Ejemplo: echo e->getMessage();

Excepciones

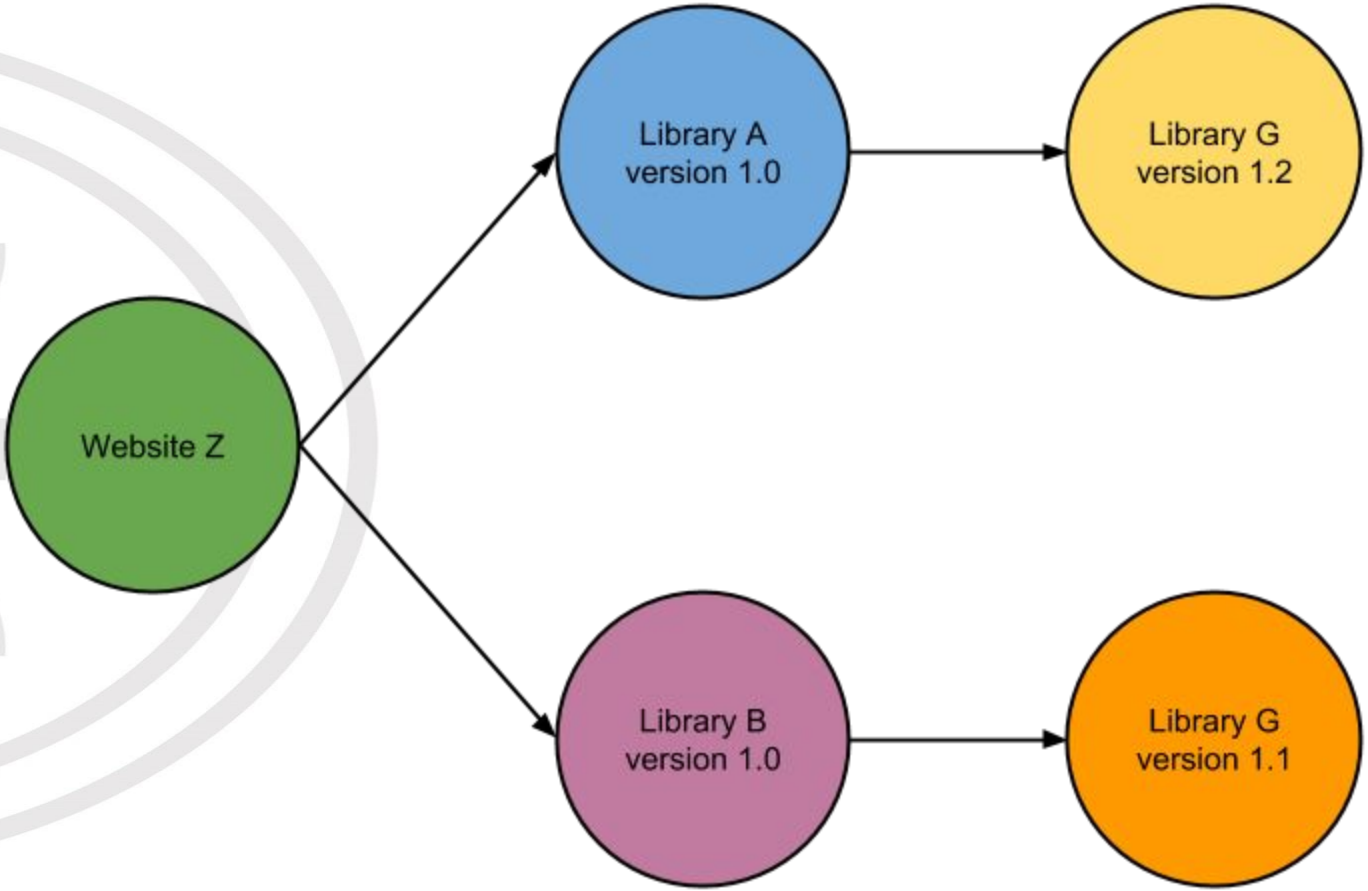
```
mysqli_report(MYSQLI_REPORT_STRICT);
try {
    $conexion = new mysqli("<host>","<usr>","<pwd>","<db>");
    $conexion->query("select * from autores");
    ...

} catch (Exception $e) {
  echo e->getMessage();
}
```

mysqli_report(MYSQLI_REPORT_STRICT) indica que si hay un error se emita una Exception

Composer

Composer

Es una herramienta para PHP que permite manejar dependencias de nuestro proyecto (usar librerias de terceros)

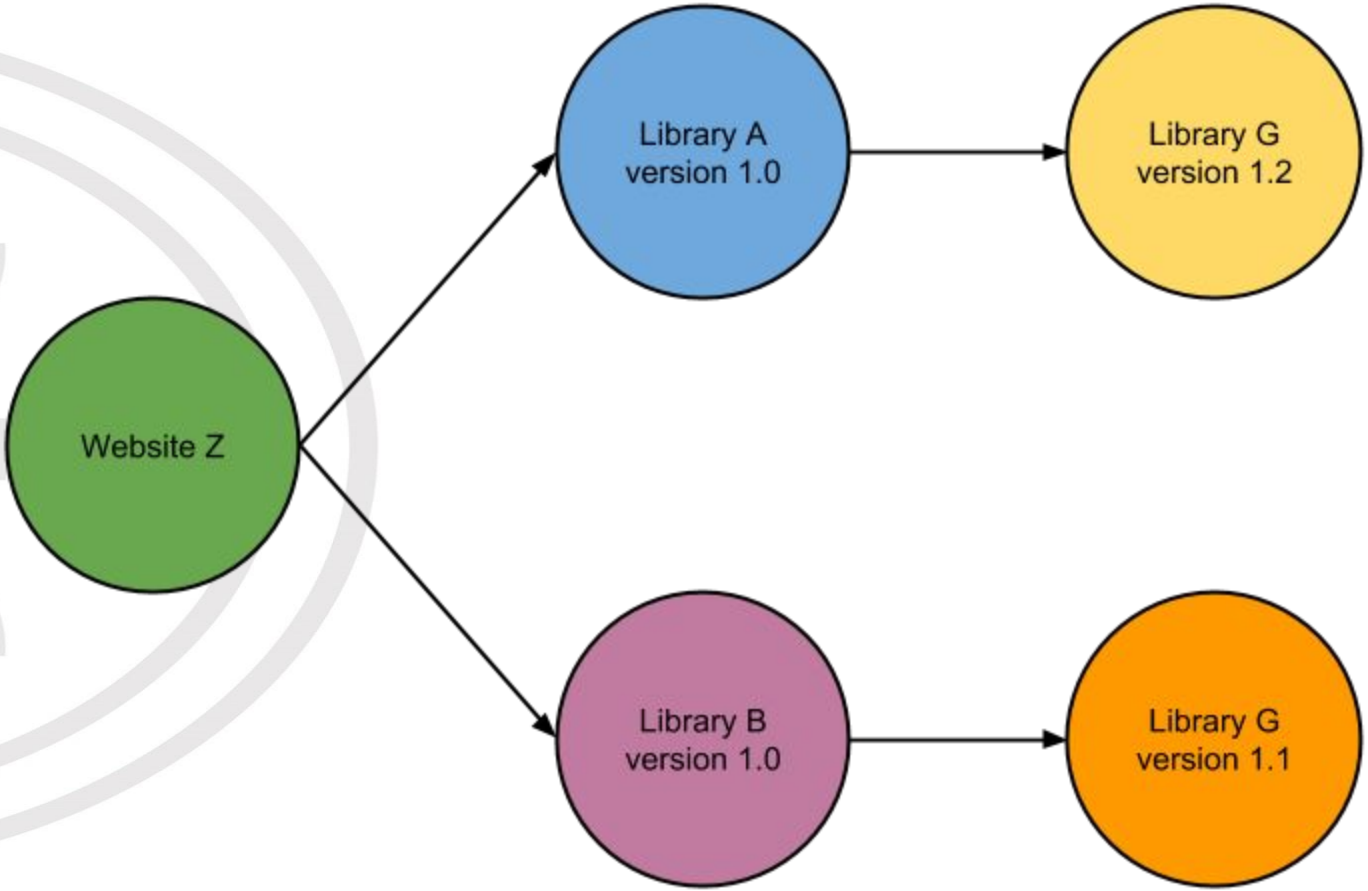

Website Z
Library A
version 1.0
Library G
version 1.2
Library B
version 1.0
Library G
version 1.1

Sin composer

Debemos descargar el código de terceros que necesitamos y todas las dependencias del mismo

Con composer

Solo indicamos que nuestro proyecto necesita X dependencia y automáticamente descarga todo lo necesario para que funcione


Website Z
Library A
version 1.0
Library G
version 1.2
Library B
version 1.0
Library G
version 1.1

Instalación Windows

Windows: Composer-Setup.exe
https://getcomposer.org/Composer-Setup.exe

Instalación Linux y MAC

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'a5c698ffe4b8e849a443b120cd5ba38043260d5c4023dbf93e1558871f1f07f58274fc6f4c93bcfd858c6bd0775cd8d
1') { echo 'Installer verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Espacio de nombres

Espacio de nombres

Al utilizar clases de diferentes componentes (y nuestro código) puede suceder que el nombre de una clase se repita.

Espacio de nombres

Si tenemos el archivo A que tiene la clase MiClase y el archivo B que también tiene una clase MiClase. Si hacemos referencia a MiClase… ¿a cuál de las 2 hacemos referencia?

Espacio de nombres

Para este tipo de inconvenientes, podemos incluir cada archivo en un espacio de nombre diferente. Ej:
A.php -> namespace Proyecto\A
B.php -> namespace Proyecto\B
Luego podemos hacer referencia que queremos crear una instancia de Proyecto\A\MiClase o de Proyecto\B\MiClase

Espacio de nombres

```php
namespace Proyecto\A;

class MiClase {

}
```

```php
namespace Proyecto\B;

class MiClase {

}
```

```php
$objA = new Proyecto\A\MiClase();
$objB = new Proyecto\B\MiClase();
```

PHPMailer

Instalación

Ejecutar en línea de comando (en la carpeta del proyecto)

```
composer require phpmailer/phpmailer
```

Uso

```php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;
require 'vendor/autoload.php';

$mail = new PHPMailer(true);
// Configuración del servidor
$mail->isSMTP();
$mail->SMTPAuth   = true;
$mail->Username   = 'user@example.com';
$mail->Password   = 'secret';
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port       = 587;

// Configuración envio y destinatario
$mail->setFrom('from@example.com', 'Mailer');
$mail->addAddress('joe@example.net', 'Joe User');
$mail->addAddress('ellen@example.com');
$mail->addReplyTo('info@example.com', 'Information');
$mail->addCC('cc@example.com');
$mail->addBCC('bcc@example.com');

$mail->isHTML(true);
$mail->Subject = 'Titulo del e-mail';
$mail->Body    = 'Cuerpo del e-mail <b>en negrita!</b>';

$mail->send();
echo 'E-mail enviado';
```

Práctica 01

Investigar que es herencia en programación orientada a objetos.
Crear un párrafo explicando con tus palabras el concepto de herencia.

Práctica 02

PHPMailer puede lanzar una excepción, ver en la documentación donde se puede producir la misma, y como debería ser el código para manejar la excepción

Práctica 03

Crea un método (notificarAlta) en tu clase de registración de usuarios

Práctica 03

Agrega la funcionalidad al método notificarAlta

- Envió de e-mail confirmando alta al usuario (con PHPMailer)

Recomendación: No uses tu cuenta de e-mail real, crea una nueva (en caso que no uses tu notebook)

Práctica 04 - Opcional

Agrega un método en tu clase de registración de usuario guardarUsuario, el cual guarde el usuario en la base de datos (debes refactorizar el código que actualmente estas utilizando)

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES