A progressive Dependency Injection Proposal for GYG

What DI is?

A form of inversion of control

The code receives dependencies, does not define or find them

Composition to the next level

Case 1 - No inversion of control

```
class UpdateRecommendations {
    public function process() {
        $files = (new S3Client('mysercret', 'myregion'))->getFiles();
        $db = new Db('myhostname', 'mypassowrd');
        foreach($files as $file) {
            $db->import($file);
        }
    }
}
```

Case 2 - Service Locator

```
class UpdateRecommendations implements ServiceLocatorAware
{

    public function process() {
        $files = $this->serviceLocator->get('s3Client')->getFiles();
        foreach($files as $file) {
            $this->serviceLocator->get('db')->import($file);
        }
    }

    public function setServiceLocator($serviceLocator) {
        $this->serviceLocator = $serviceLocator;
    }
}
```

Case 3 - Dependency Injection

```
$di[UpdateRecommendations::class] = function($di) {
    return new UpdateRecommendations($di['db'], $di['s3Client'])
};

class UpdateRecommendations
{
    public function __construct($db, $s3Client) {
        $this->db = $db;
        $this->s3Client = $s3Client;
    }

    public function process() {
        $files = $this->s3Client->getFiles();
        foreach($files as $file) {
            $this->db->import($file);
        }
    }
}
```

Benefits

- Single entry point for complexity
- Is obvious when something needs to be refactored
- Better testability
- Easy to change software parts

Which Lib?

Is possible to do with none

Which Lib?

- zend/service-locator
- pimple/pimple
- thephpleague/container
- phpdi/phpdi

PHP-DI

- PSR'll compatible
- Support inference
- Support annotation
- Support composing SL
- Support caching of closures

A progressive approach

- Use DI for new code
- SL is still useful
- Instantiate DTOs directly

Even better with

A little architecture

(Service, Gateway)

- Interface oriented software development
- Smaller API's ISP
- Test business rules not code!
- Even easier to change parts

References

- PSRII
- A little architecture