

A post-apocalyptic landscape featuring several large, rusted satellite dishes scattered across a field of tall, golden grass. In the distance, a lone figure is walking through the field. The sky is dark and filled with many birds flying. The overall atmosphere is desolate and haunting.

HOW TO SURVIVE:
THE ENDLESS FUN
THAT IS LEGACY
CODE



FREDERICK

VANBRABANT

@maybeFrederick

www.frederickvanbrabant.com

KEYBOARD FOR HIRE

**BRODY &
HOOPER**







**"INHERENT VALUE OF THE CODE
RULES EVERYTHING AROUND ME"**

- THE WU TANG CLAN

SO HOW DO WE GET STARTED?

**FIRST THINGS FIRST:
MENTALITY**

DON'T REWRITE

***“ MOVE FAST AND BREAK
THINGS. UNLESS YOU ARE
BREAKING STUFF, YOU ARE NOT
MOVING FAST ENOUGH.***

***“ MOVE FAST AND BREAK
THINGS. UNLESS YOU ARE
BREAKING STUFF, YOU ARE NOT
MOVING FAST ENOUGH.***

- Guy that never worked on legacy code

ACT LIKE IT IS OPEN-SOURCE

**AUTOMATE & ANALYSE
EVERYTHING**

**STOP THINKING ABOUT YOUR
FAVOURITE FRAMEWORK**

NEXT UP:
CODE MINDSET

**KEEP METHODS AND CLASSES
SMALL**

**KEEP PR'S AND COMMITS
SMALL**

DO CODE REVIEWS

PAIR PROGRAM AND MENTOR

**DOWN TO BUSINESS:
GETTING STARTED**

**GET YOUR BUILDS, GIT AND
DEPLOYS UNDER CONTROL**

USE A DEBUG TOOL

USE A FULL ON IDE,

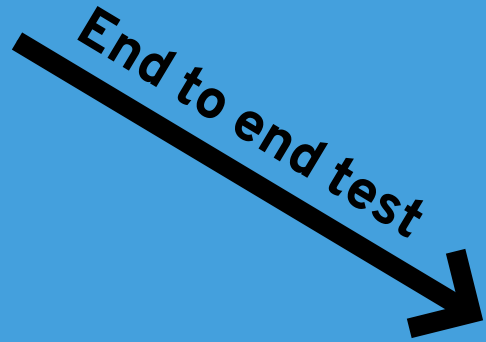
No ... seriously

**WRITE END TO END TESTS.
HEAPS OF THEM**

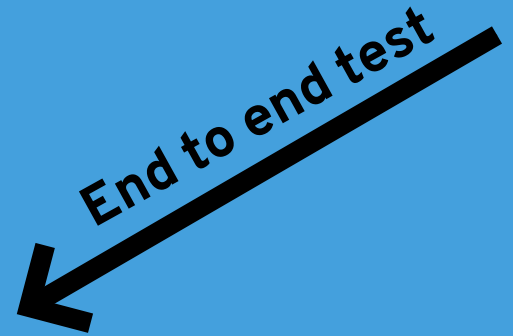
WORK OUTSIDE IN



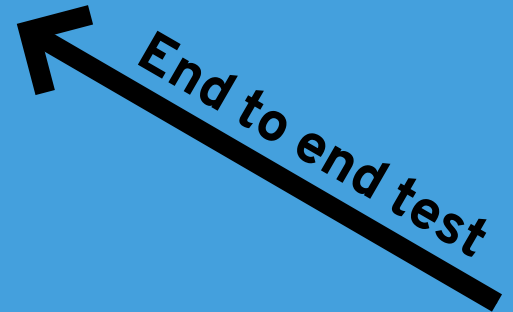
End to end test

A thick black arrow pointing from the top-left towards the center, with the text "End to end test" written above it.

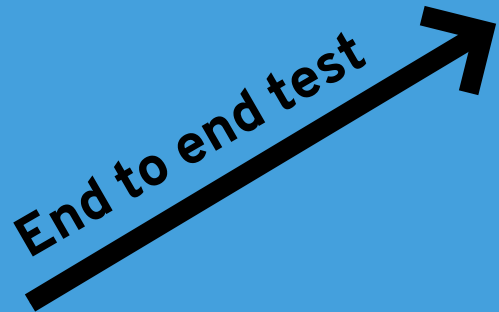
End to end test

A thick black arrow pointing from the top-right towards the center, with the text "End to end test" written above it.

End to end test

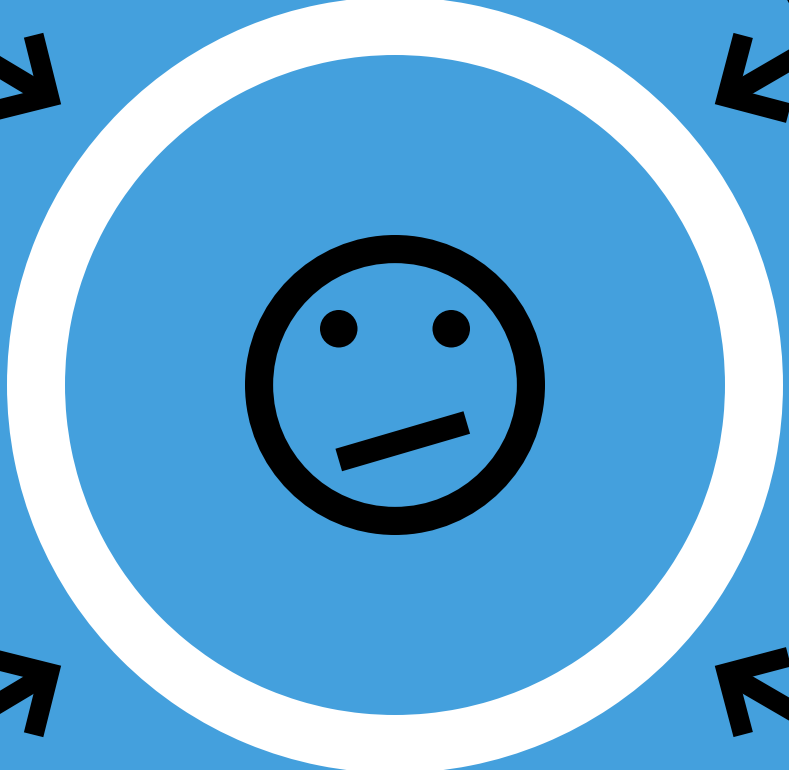
A thick black arrow pointing from the bottom-right towards the center, with the text "End to end test" written above it.

End to end test

A thick black arrow pointing from the bottom-left towards the center, with the text "End to end test" written above it.

End to end test

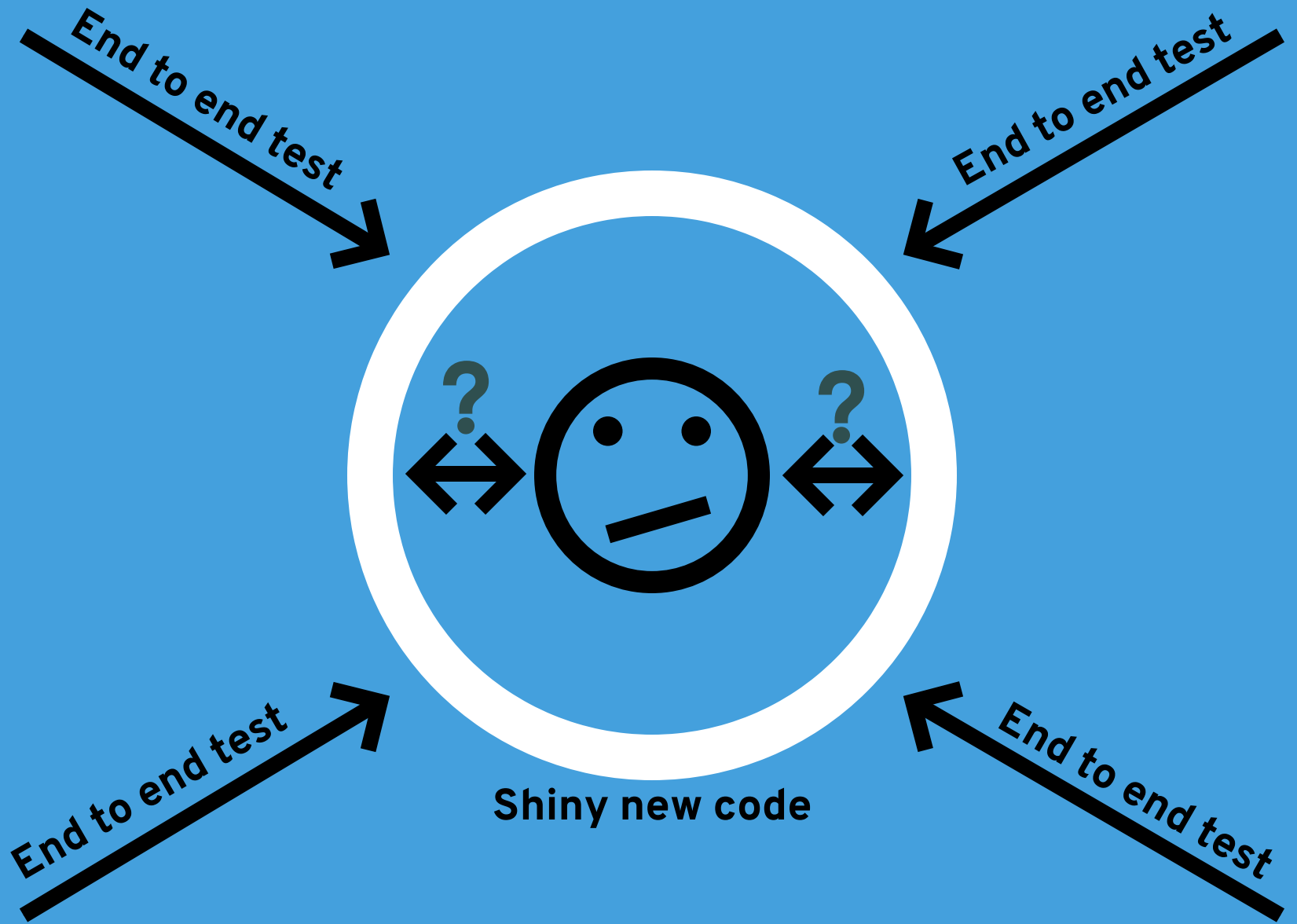
End to end test



Shiny new code

End to end test

End to end test



```
<?php
```

```
* A million things *
```

```
$conn = new mysqli('catInc', 'root', 'hunter2', 'main');
```

```
if ($conn->connect_error) {  
    die("Connection failed");  
}
```

```
$sql = "INSERT INTO Cats (name) VALUES ('Thoru)";  
$conn->query($sql)
```

```
$conn->close();
```

```
* A million things *
```

```
<?php
```

```
class KittenTranslationLayer {  
  
    public static function addCat($name)  
    {  
        $container = App::getContainer();  
        $catRepository = $container  
            ->get(CatRepository::class);  
  
        $catRepository->addCat(new CatName($name));  
    }  
}
```

```
<?php
```

```
class CatRepository {  
  
    private $connection;  
  
    public function __construct(Connection $connection)  
    {  
        $this->connection = $connection;  
    }  
  
    public function addCat(CatName $name)  
    {  
        $query = $this->connection  
            ->createQueryBuilder();  
  
        $query->insert('cats')  
            ->setValue('name', ':name')  
            ->setParameter('name', $name)  
            ->execute();  
    }  
}
```



```
<?php
```

```
* A million things *
```

```
$conn = new mysqli('catInc', 'root', 'hunter2', 'main');
```

```
if ($conn->connect_error) {  
    die("Connection failed");  
}
```

```
$sql = "INSERT INTO Cats (name) VALUES ('Thoru)";  
$conn->query($sql)
```

```
$conn->close();
```

```
* A million things *
```


**NOW ADD ALL THE OTHER
TESTS**

THE THING ABOUT MICROSERVICES ...

WE AREN'T DONE YET
THROW SOME DDD AT IT

**ORDER PER DOMAIN NOT PER
ACTION**

```
App/  
├── Controllers/  
│   ├── UserController.php  
│   └── PostController.php  
├── Models/  
│   ├── User.php  
│   └── Post.php  
├── Validators/  
│   ├── UserCreate.php  
│   ├── UserUpdate.php  
│   ├── PostCreate.php  
│   └── PostUpdate.php  
└── Transformers/  
    ├── User.php  
    └── Post.php
```


```
App/  
├── Controllers/  
│   ├── UserController.php  
│   └── PostController.php  
├── Models/  
│   ├── User.php  
│   └── Post.php  
├── Validators/  
│   ├── UserCreate.php  
│   ├── UserUpdate.php  
│   ├── PostCreate.php  
│   └── PostUpdate.php  
└── Transformers/  
    ├── User.php  
    └── Post.php
```

```
App/  
├── Http/  
│   ├── UserController.php  
│   └── PostController.php  
├── User/  
│   ├── Validators/  
│   │   ├── Create.php  
│   │   └── Update.php  
│   ├── Transformer.php  
│   └── User.php  
└── Post/  
    ├── Validators/  
    │   ├── Create.php  
    │   └── Update.php  
    ├── Transformer.php  
    └── Post.php
```


**ON THAT NOTE:
VALUE OBJECTS**


```
<?php
```

```
class UserRepository {  
    public function setAmountOfCats(  
        int $amount,  
        UserId $id  
    ) {  
        *code*  
    }  
}
```



**GUESS I'M NOW A
LEGACY CODE
DEVELOPER**

**YOU'RE GOING TO SEE SOME
FUNKY THINGS**

**TRY AND UNDERSTAND WHY
SOMETHING IS DONE**

LEARN MORE?

- Modernizing Legacy Applications In PHP
 - PM Jones
- Working Effectively with Legacy Code
 - Michael Feathers
- The League of Extraordinary Packages

THANKS

QUESTIONS



MAYBEFREDERICK