



# Exceptions

[sd@freematiq.com](mailto:sd@freematiq.com)



# \_\_toString

```
class User
{
    private $name;
    private $password;

    public function __toString()
    {
        return "User. Name: {$this->name}" . PHP_EOL;
    }
}
```

```
$user = new \Freematiq\User();
echo $user;
```

Не используйте \_\_toString() в бизнес логике!  
Можно использовать для удобства



# Цепочка ВЫЗОВОВ

```
public function setName($name)
{
    $this->name = $name;
    return $this;
}
```

```
public function setPassword($password)
{
    $this->password = $password;
    return $this;
}
```

```
$user = new \Freematiq\User();
$user->setName('Ivan')->setPassword('USSR');
echo $user;
```



# Исключения

Исключение - некоторое критическое сообщение об ошибке

При генерации оно передаётся в ту часть программы, которая его обрабатывает


Исключение - это объект

Содержит текст + системную информацию



# Пример

```
try {  
    echo "В блоке исключения" . PHP_EOL;  
    if (true) {  
        throw new \Exception("Что-то пошло не так!");  
    }  
    echo "Всё хорошо!"; // уже не выполнится  
} catch (\Exception $e) {  
    echo "Oops!" . PHP_EOL;  
    echo "Problem: {" . $e->getMessage() . "}" . PHP_EOL;  
}
```



# Наследование ИСКЛЮЧЕНИЙ

```
class LearningException extends \Exception {}

try {
    echo "В блоке исключения" . PHP_EOL;
    if (true) {
        throw new LearningException("Что-то пошло не так!");
    }
    echo "Всё хорошо!"; // уже не выполнится
} catch (LearningException $e) {
    echo "Ой!" . PHP_EOL;
    echo "Наша учебная ошибка: {" . $e->getMessage() . "}" . PHP_EOL;
} catch (\Exception $e) {
    echo "Oops!" . PHP_EOL;
    echo "Problem: {" . $e->getMessage() . "}" . PHP_EOL;
}
```




# Когда и как ИСПОЛЬЗОВАТЬ

Все исключения состоят в иерархии  
Exception - самый высокий уровень (в  
php 5)

Использовать при нештатных  
ситуациях, когда их обработка  
переносится на более высокий уровень

if оставить там, где обработка  
ситуации решается “на месте”



# Exception src

```
public function __construct($message = "", $code = 0,  
Throwable $previous = null) { }
```

```
class Exception implements Throwable {  
    protected $message;  
    protected $code;  
    protected $file;  
    protected $line;
```

```
    final public function getTrace() { }  
    public function __toString() { }
```





# e-mail

<https://swiftmailer.symfony.com>

composer require swiftmailer/swiftmailer

Swift\_SmtpTransport - настройка  
отправки

Swift\_Mailer - отправка почты

Swift\_Message - сообщение



# e-mail

```
$transport = new Swift_SmtpTransport('smtp.mailtrap.io',  
25);
```

```
$transport  
->setUsername('194152d5330321233')  
->setPassword('3687a094fdbcb');
```

```
$swift = new Swift_Mailer($transport);
```

```
$message = new Swift_Message('Всем привет!', 'Это  
письмо отправлено из учебного класса');
```

```
$message->setFrom('sd@freematiq.com', 'Сергей');
```

```
$message->setTo('all@freematiq.com', 'Класс');
```

```
$result = $swift->send($message);
```

```
echo "Mail sent. Result = $result\n";
```



# DateTime

С php 5.2

- DateTime - время
- DateTimeZone - временная зона
- DateInterval - интервал между двумя датами



# DateTime

```
$date = new DateTime();  
echo $date->format('Y-m-d H:i:s'), PHP_EOL;  
echo $date->getTimestamp(), PHP_EOL;
```

```
$date = new DateTime('now', new DateTimeZone('Europe/Moscow'));  
echo $date->format('Y-m-d H:i:s'), PHP_EOL;
```

```
// public function setDate ($year, $month, $day) {}  
$date->modify('+3 days');  
echo $date->format('Y-m-d H:i:s'), PHP_EOL;
```

```
$date1 = new DateTime('2017-07-24 19:00');  
$date2 = new DateTime('2017-01-01 0:00');
```

```
$result = $date2->diff($date1);  
print_r($result);
```



# Итого

- Объект можно представить в виде строки `__toString()`
- Исключения - важная часть обработки ошибок
- Исключения составляют иерархию
- PHP содержит множество классов и методов работы



Конец