STATE IN THE STATELESS WORLD

Luka Mužinić
@lmuzinic

codementor

bit.ly/phpsrbija-codementor

WHY WE NEED STATE?

THE PROBLEM

STATUS OF ENTITIES

Our entities can have life of their own, they start out one way and then after series of events they end up different.

We want to store that information.

EASY PEASY

JUST ADD A PROPERTY

```php
/**
 * @var string
 */
private $status;

/**
 * @return string
 */
public function getStatus()
{
    return $this->status;
}

/**
 * @param string $status
 */
public function setStatus($status)
{
    $this->status = $status;
}
```

EASY PEASY

JUST CHANGE A PROPERTY AND SAVE IT

```
$em = $this->getDoctrine()->getManager();
$rep = $em->getRepository(Entity::class);

$entity = $rep->find(1);

$entity->setStatus('paid');

$entityManager->flush();
```

BUT THERE ARE ALSO SOME BUSINESS RULES

"I should be able to publish an article only after lector says it is OK"

"Warehouse should not ship customer orders until they have been fully paid"

"Support can promote users if they have verified their email address"

HOW TO ADD BUSINESS RULES?

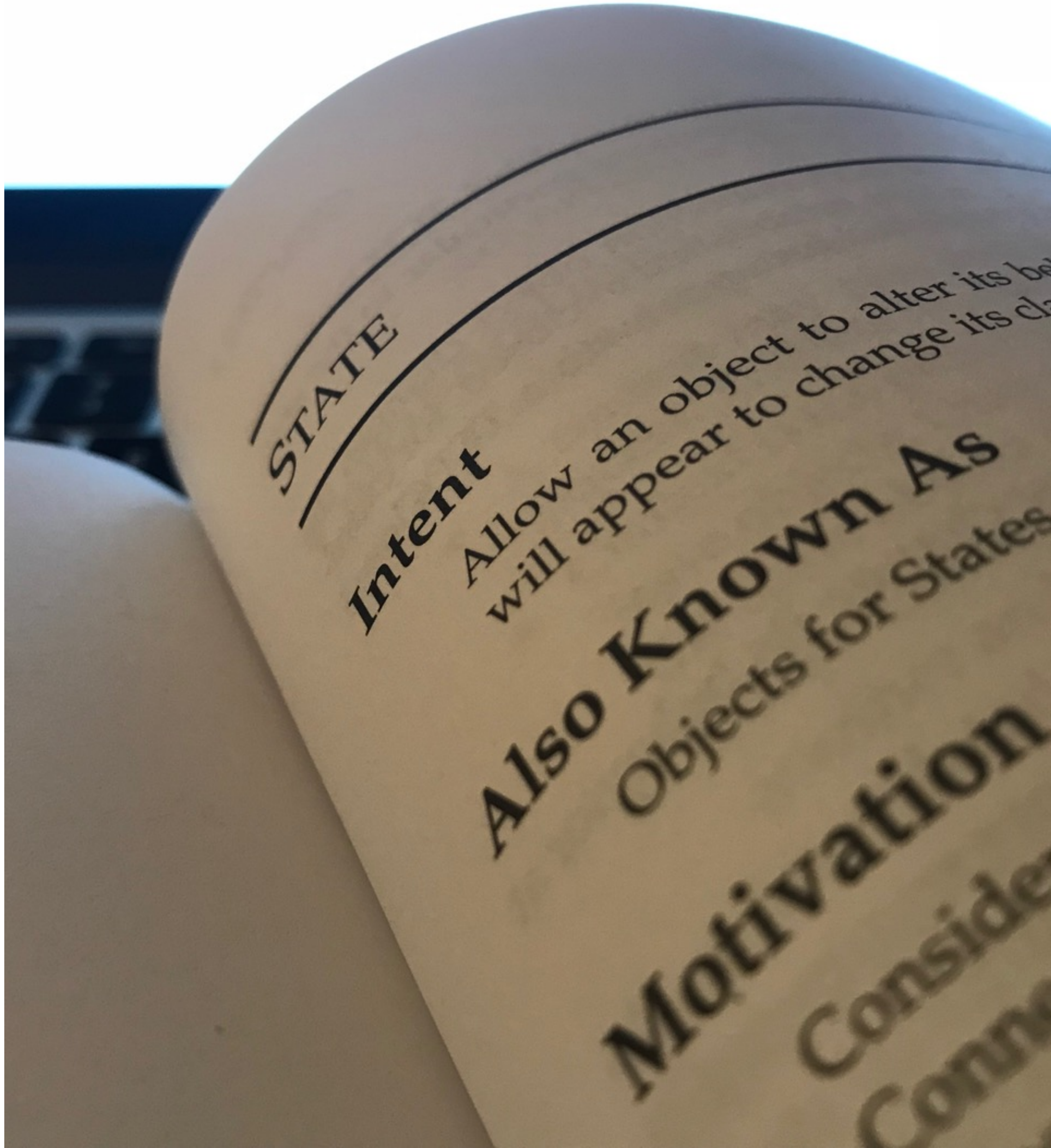
There must be a clever way to include business rules in our system.

STATE PATTERN

INTENT

Allow an object to alter its behaviour when its internal state changes. The object will appear to change its class.

http://wiki.c2.com/?StatePattern



STATE

Intent Allow an object to alter its be
will appear to change its cla

Also Known As
Objects for States

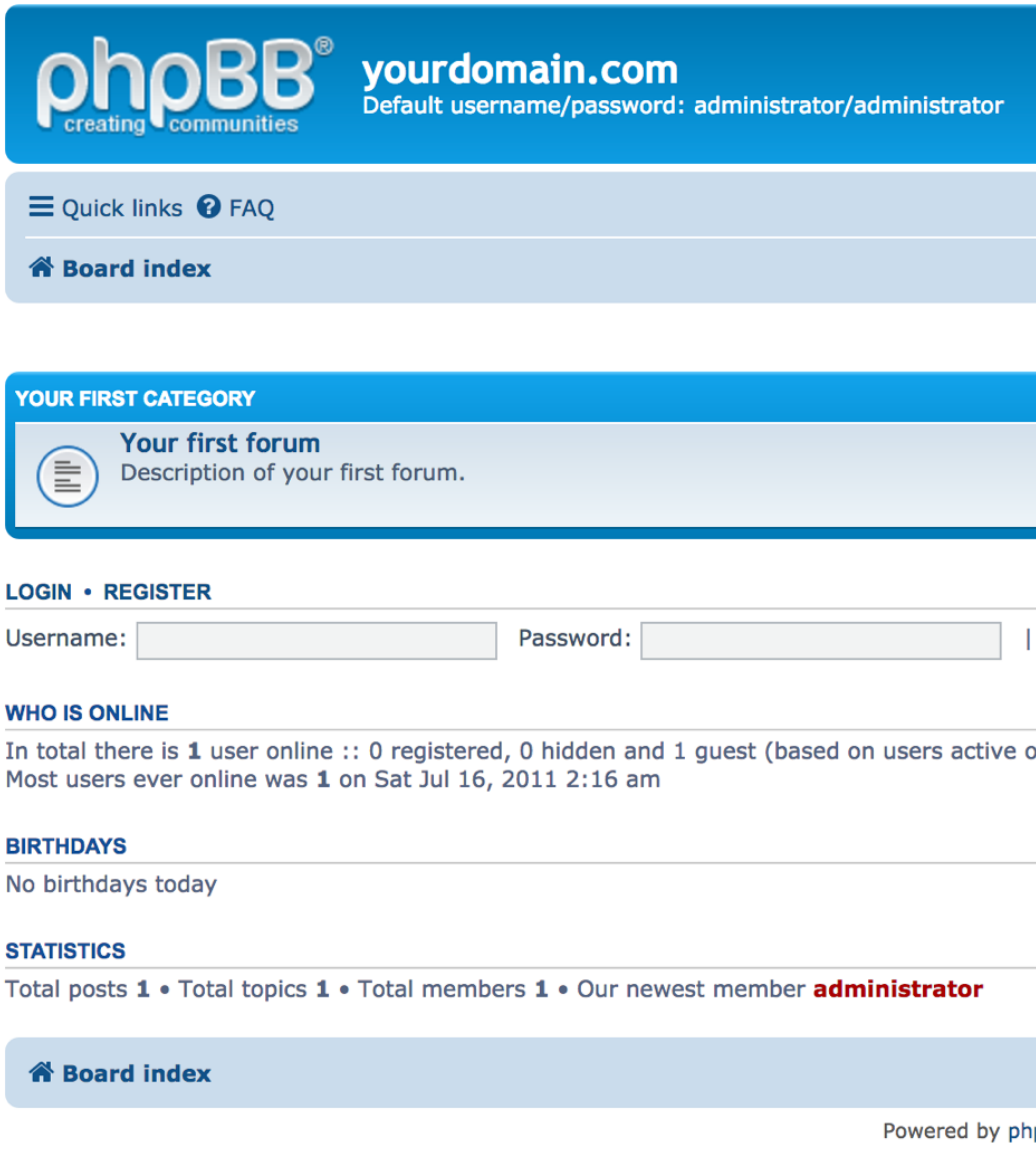
Motivation
Conside
Conn

STATE PATTERN

USAGE

The State pattern can be used, for instance, to implement a Finite State Machine efficiently and elegantly. This approach can be useful when implementing business processes or workflows.

https://github.com/sebastianbergmann/state



yourdomain.com
Default username/password: administrator/administrator

YOUR FIRST CATEGORY

Your first forum
Description of your first forum.

Username: Password:

WHO IS ONLINE

In total there is **1** user online :: 0 registered, 0 hidden and 1 guest (based on users active o
Most users ever online was **1** on Sat Jul 16, 2011 2:16 am

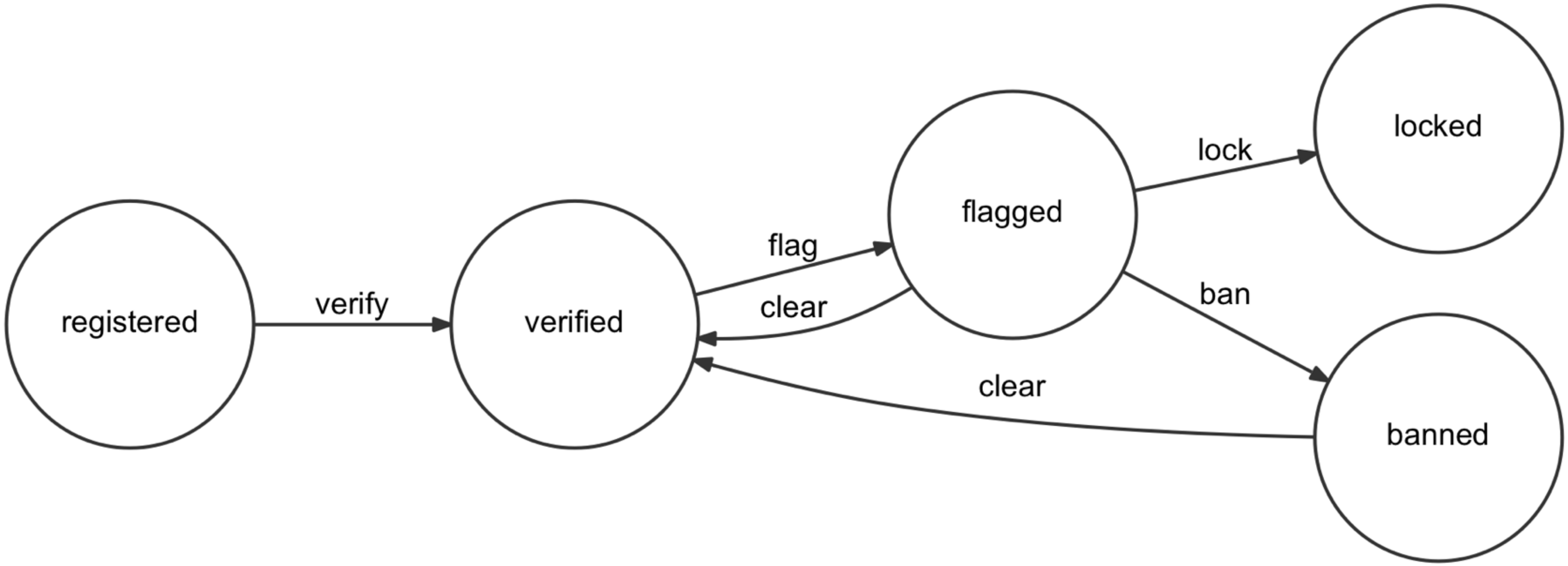
BIRTHDAYS

No birthdays today

STATISTICS

Total posts **1** • Total topics **1** • Total members **1** • Our newest member **administrator**

Powered by ph


registered
verify
verified
flag
clear
flagged
lock
locked
ban
banned
clear

STATE PATTERN

```
interface UserState
{
    public function verify();

    public function flag();

    public function ban();

    public function lock();

    public function clear();
}
```

Start off with a simple interface listing all possible transitions

STATE PATTERN

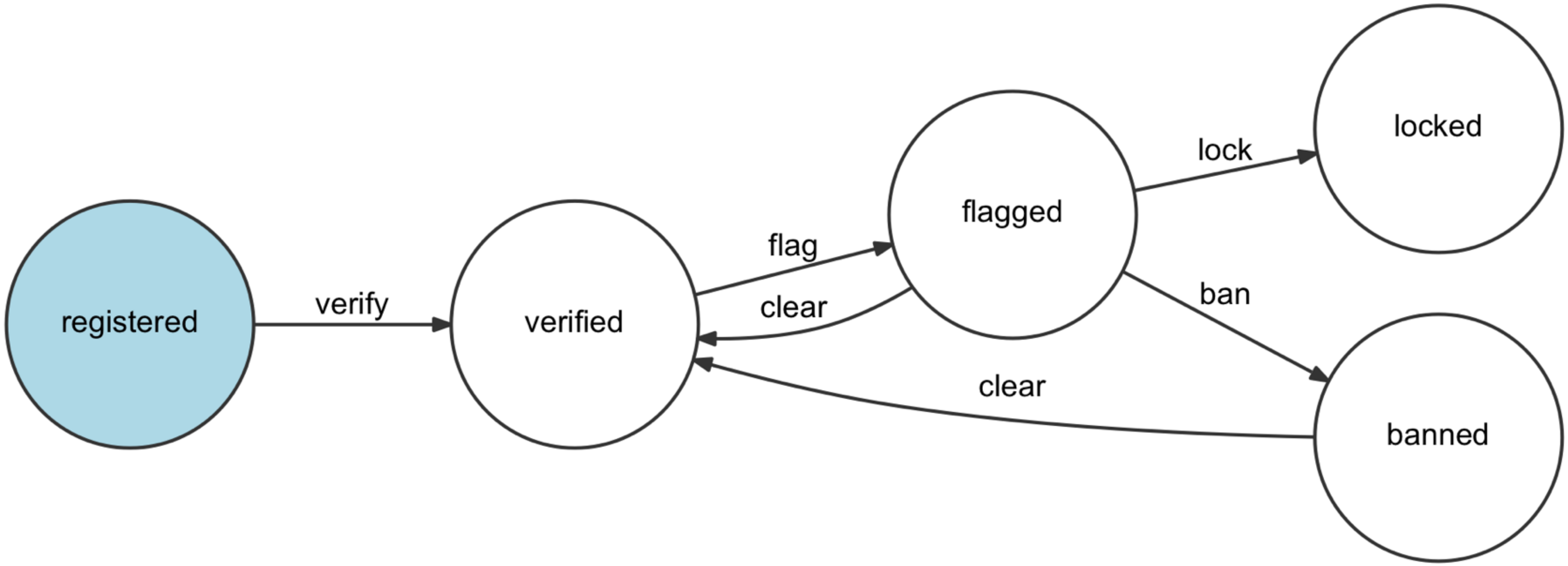
```php
abstract class AbstractUserState implements UserState
{
    public function verify()
    {
        throw new IllegalStateTransitionException;
    }

    public function flag()
    {
        throw new IllegalStateTransitionException;
    }

    public function ban()
    {
        throw new IllegalStateTransitionException;
    }

    public function lock()
    {
        throw new IllegalStateTransitionException;
    }

    public function clear()
    {
        throw new IllegalStateTransitionException;
    }
}
```

Make all transitions throw exceptions by default

STATE PATTERN

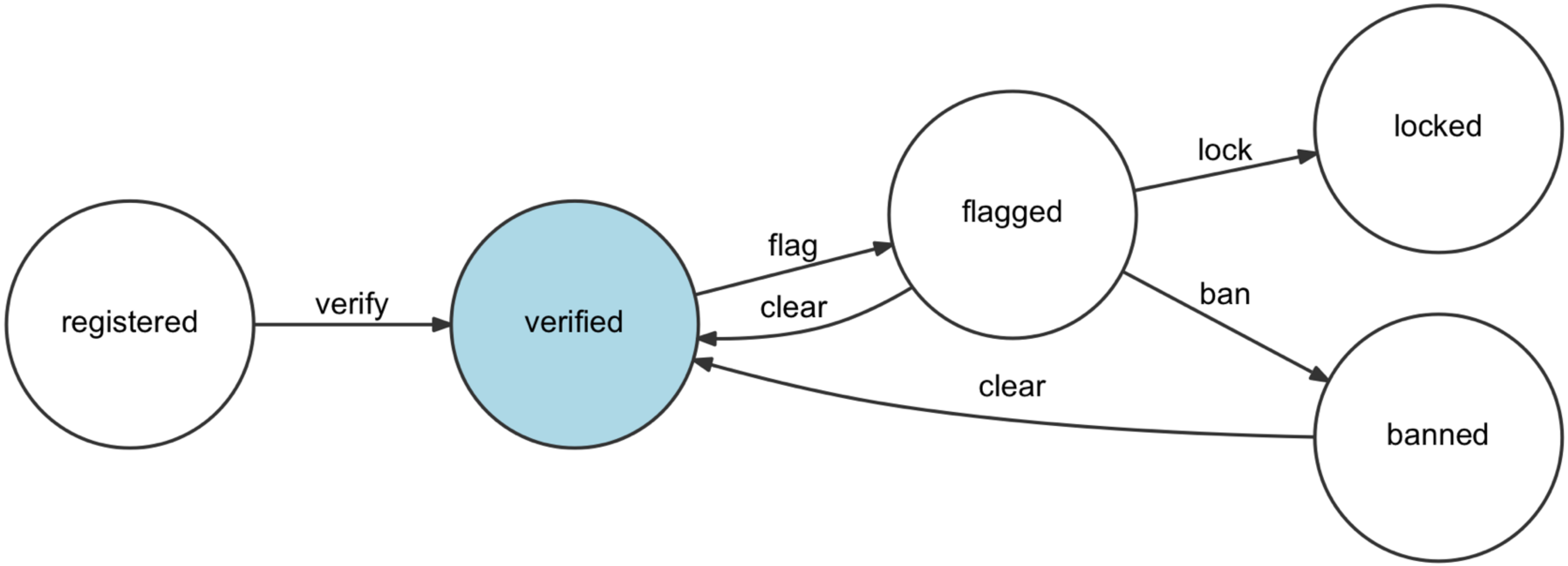
```
class RegisteredUserState extends AbstractUserState
{
    public function verify()
    {
        return new VerifiedUserState;
    }
}
```

Define states as factories to other states



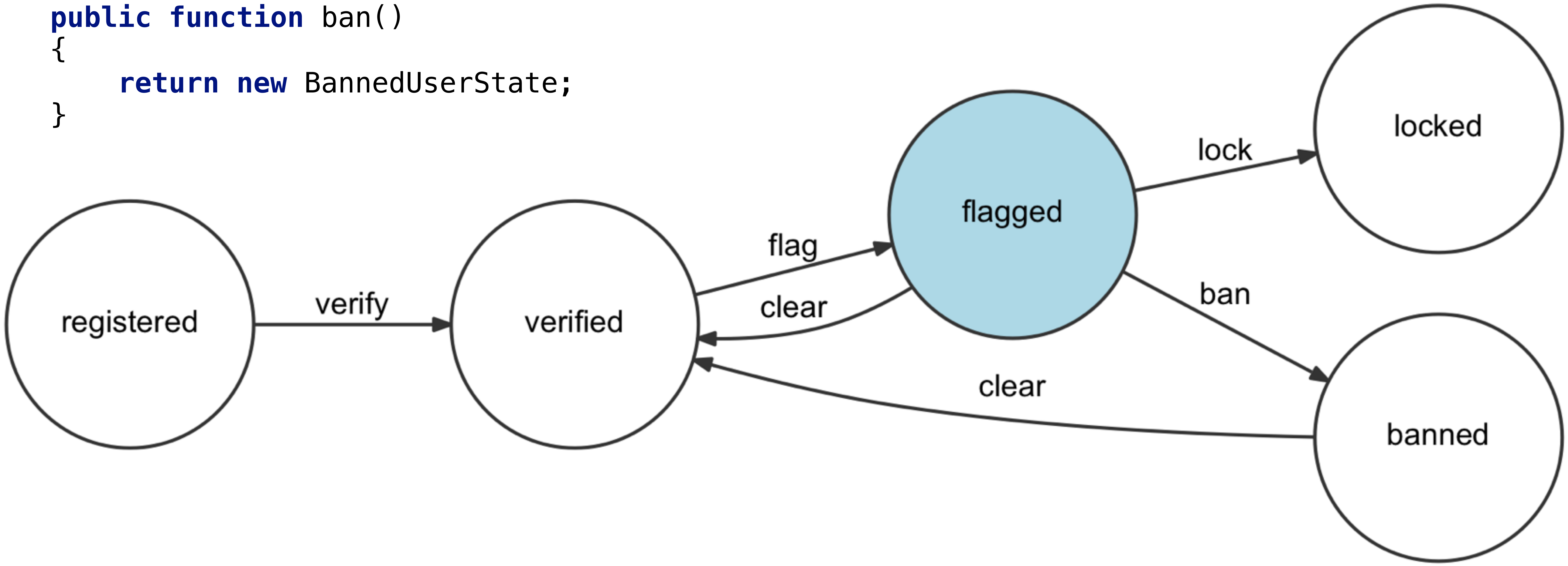
STATE PATTERN

```
class VerifiedUserState extends AbstractUserState
{
    public function flag()
    {
        return new FlaggedUserState;
    }
}
```



STATE PATTERN

```
class FlaggedUserState extends AbstractUserState
{
    public function lock()
    {
        return new LockedUserState;
    }

    public function ban()
    {
        return new BannedUserState;
    }
}
```



STATE PATTERN

```
class User
{
    private $state;

    public function __construct(UserState $state)
    {
        $this->setState($state);
    }

    private function setState(UserState $state)
    {
        $this->state = $state;
    }

    public function verify()
    {
        $this->setState($this->state->verify());
    }

    public function isVerified()
    {
        return $this->state instanceof VerifiedUserState;
    }
}
```

Add state to your entity

TESTING STATE PATTERN

```
class RegisteredUserTest extends TestCase
{
    /** @var \App\Entity\User */
    private $user;

    public function setUp()
    {
        $this->user = new User('Miro Svrtan');
        $this->user->setState(new RegisteredUserState());
    }
```

TESTING STATE PATTERN

```php
class RegisteredUserTest extends TestCase
{
    public function testBecomesVerified()
    {
        $this->user->verify();

        $this->assertTrue(
            $this->user->isVerified()
        );
    }

    public function testCanNotBeFlagged()
    {
        $this->expectException(IllegalStateTransitionException::class);

        $this->user->flag();
    }
}
```

TESTING STATE PATTERN

```
~ bin/phpunit --testdox
PHPUnit 6.5.8 by Sebastian Bergmann and contributors.

App\Tests\RegisteredUser
 [x] Becomes verified
 [x] Can not be flagged
 [x] Can not be locked
 [x] Can not be banned
```

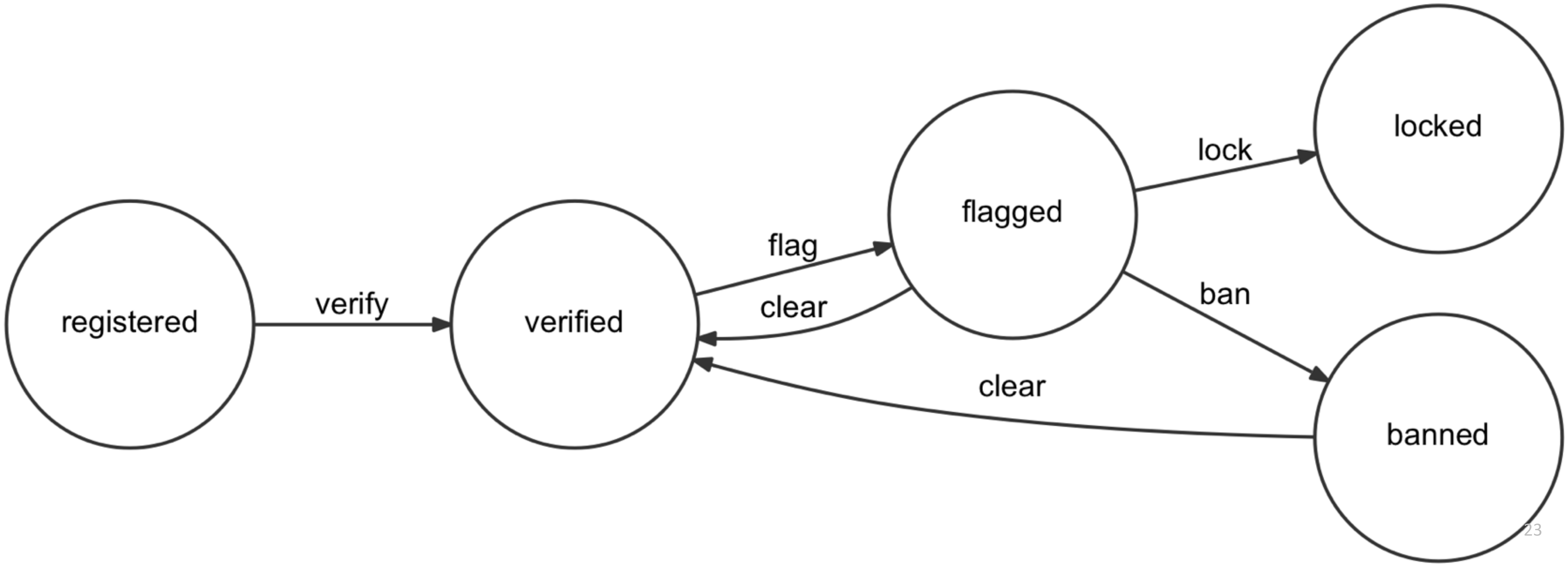
BUT MY ENTITIES ARE ALREADY BIG!

Shoving all that inside your entity makes them fat.

IS THERE A BETTER WAY?

STATE MACHINES

DEFINING STATES AND ALLOWED TRANSITIONS BETWEEN THEM



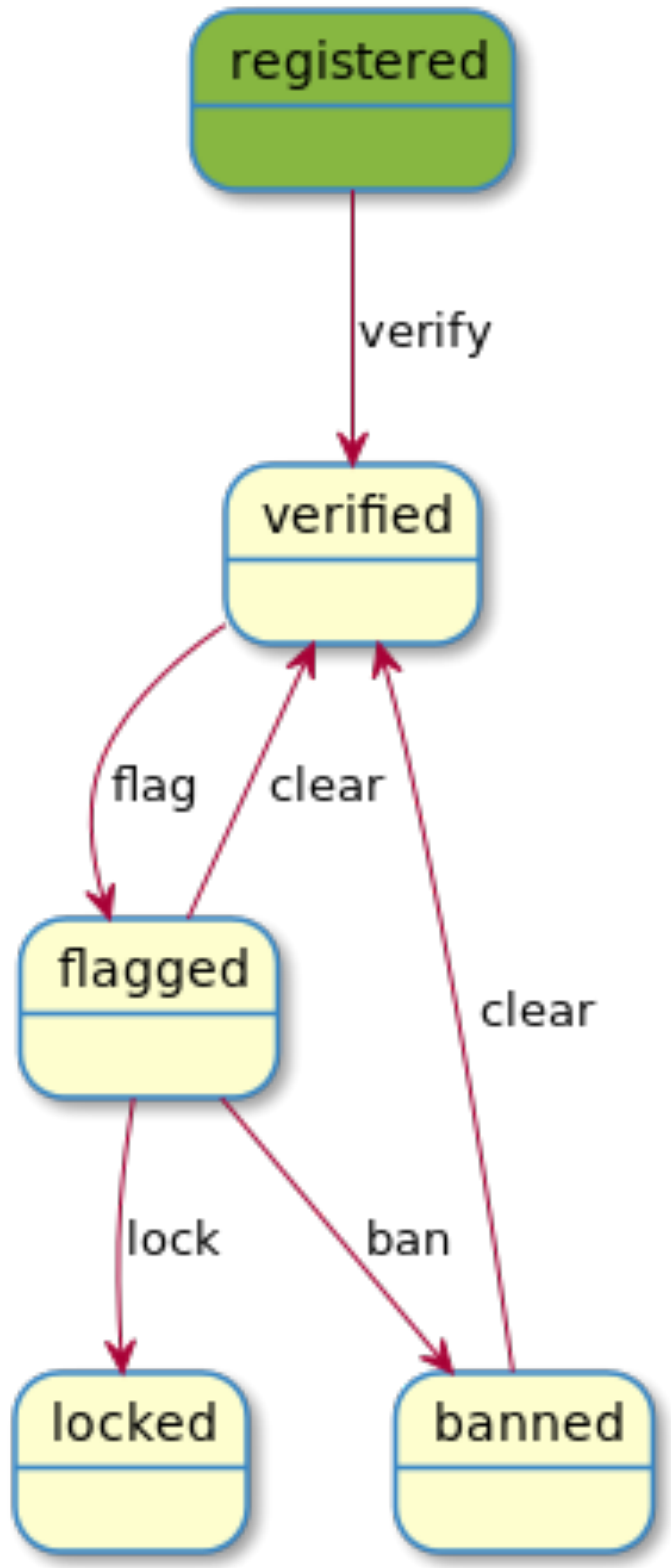
STATE MACHINES PACKAGES

```
composer require symfony/workflow

composer require winzou/state-machine

composer require yohang/finite
```

Pick one

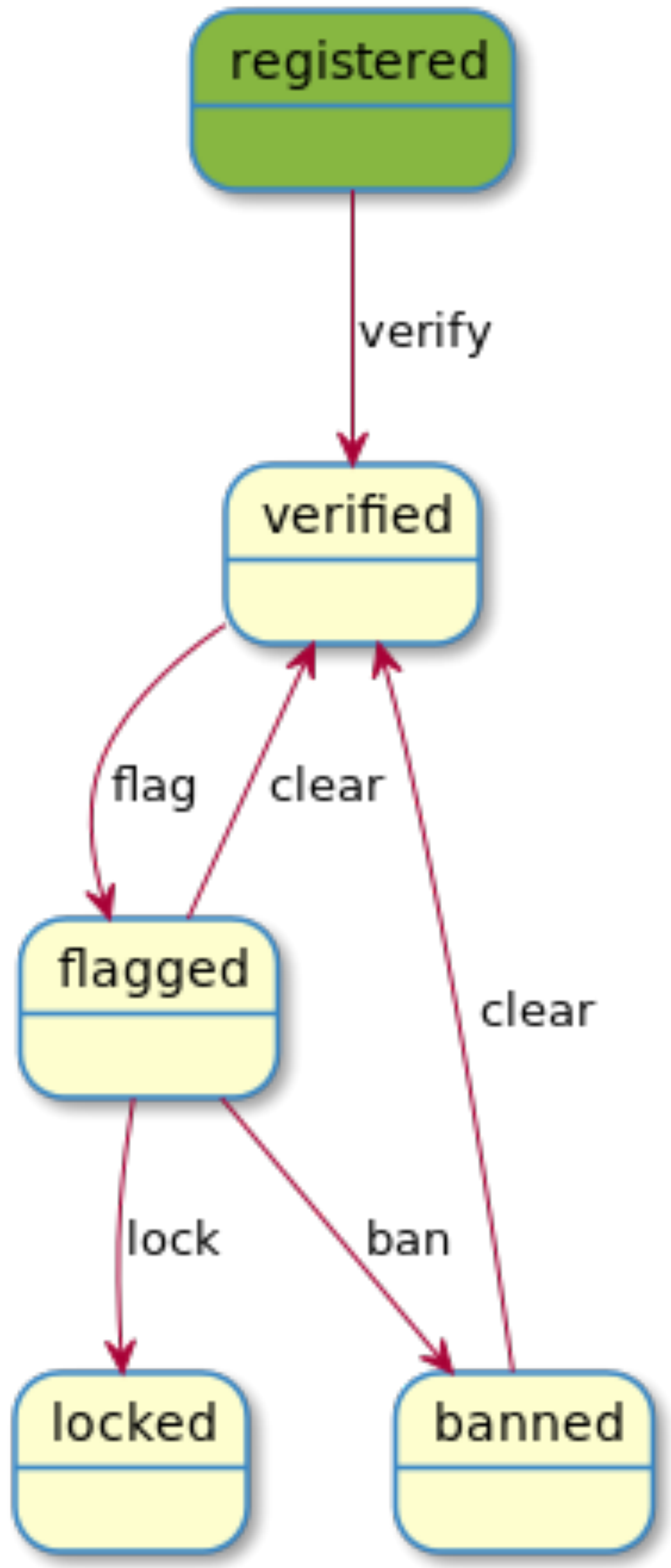
SYMFONY WORKFLOW CONFIG

```yaml
workflows:
    user:
        type: 'workflow'
        marking_store:
            type: 'single_state'
            arguments:
                - 'status'
        supports:
            - App\Entity\User
        places:
            - registered
            - verified
            - flagged
            - banned
            - locked
```



SYMFONY WORKFLOW CONFIG

```yaml
workflows:
    user:
        transitions:
            verify:
                from: registered
                to: verified
            flag:
                from: verified
                to: flagged
            ban:
                from: flagged
                to: banned
            lock:
                from: flagged
                to: locked
            clear:
                from: [banned, flagged]
                to: verified
```



SYMFONY WORKFLOW USAGE

```php
$user = new AppBundle\Entity\User();

$stateMachine = $this->container->get('state_machine.user');

if ($stateMachine->can($user, 'ban')) {
    $stateMachine->apply($user, 'ban');
}
```

Focus on what needs to be done, not if it can be done!

SYMFONY WORKFLOW EVENTS

```
class Workflow
{
    public function __construct(
        Definition $definition,
        MarkingStoreInterface $markingStore = null,
        EventDispatcherInterface $dispatcher = null,
        $name = 'unnamed'
    )

...
```

Cool, EventDispatcher is there!

SYMFONY WORKFLOW EVENTS

```php
$stateMachine->can($user, 'ban');
```

GUARD

workflow.guard
workflow.[workflow name].guard
workflow.[workflow name].guard.[transition name]

SYMFONY WORKFLOW EVENTS

```php
class UserSubscriber implements EventSubscriberInterface
{
    public function guardBan(GuardEvent $guardEvent)
    {
        /** @var \App\Entity\User $user */
        $user = $guardEvent->getSubject();

        if ($user->isModerator()) {
            $guardEvent->setBlocked(true);
        }
    }

    public static function getSubscribedEvents()
    {
        return [
            'workflow.user.guard.ban' => ['guardBan']
        ];
    }
}
```

Block it if necessary!

SYMFONY WORKFLOW EVENTS

```
$stateMachine->apply($user, 'ban');
```

LEAVE

workflow.leave
workflow.[workflow name].leave
workflow.[workflow name].leave.[transition name]

TRANSITION

workflow.transition
workflow.[workflow name].transition
workflow.[workflow name].transition.[transition name]

ENTER

workflow.enter
workflow.[workflow name].enter
workflow.[workflow name].enter.[transition name]

ENTERED

workflow.entered
workflow.[workflow name].entered
workflow.[workflow name].entered.[transition name]

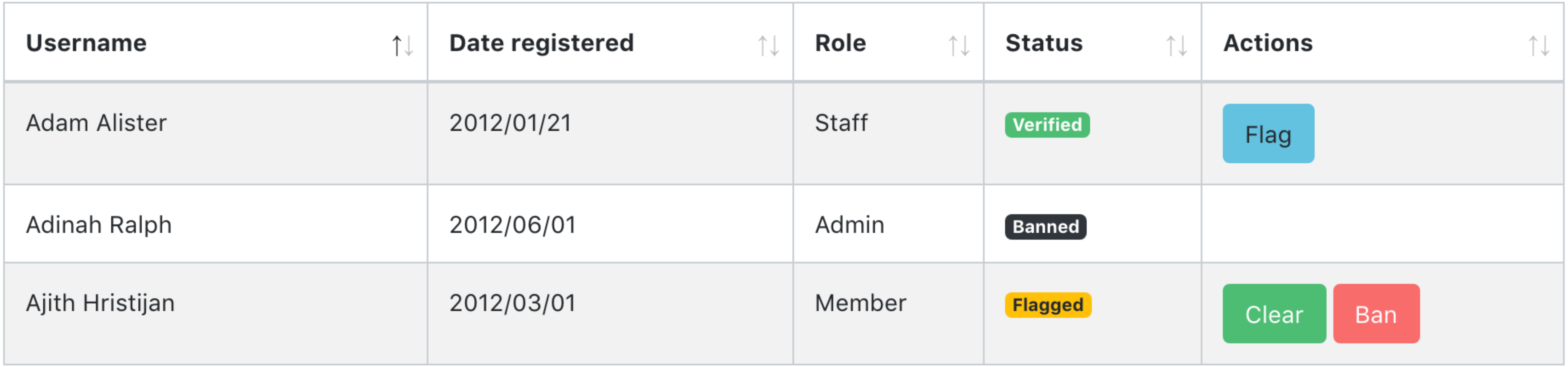
COMPLETED

workflow.completed
workflow.[workflow name].completed
workflow.[workflow name].completed.[transition name]

ANNOUNCE

workflow.announce
workflow.[workflow name].announce
workflow.[workflow name].announce.[transition name]

SYMFONY WORKFLOW IN TEMPLATES

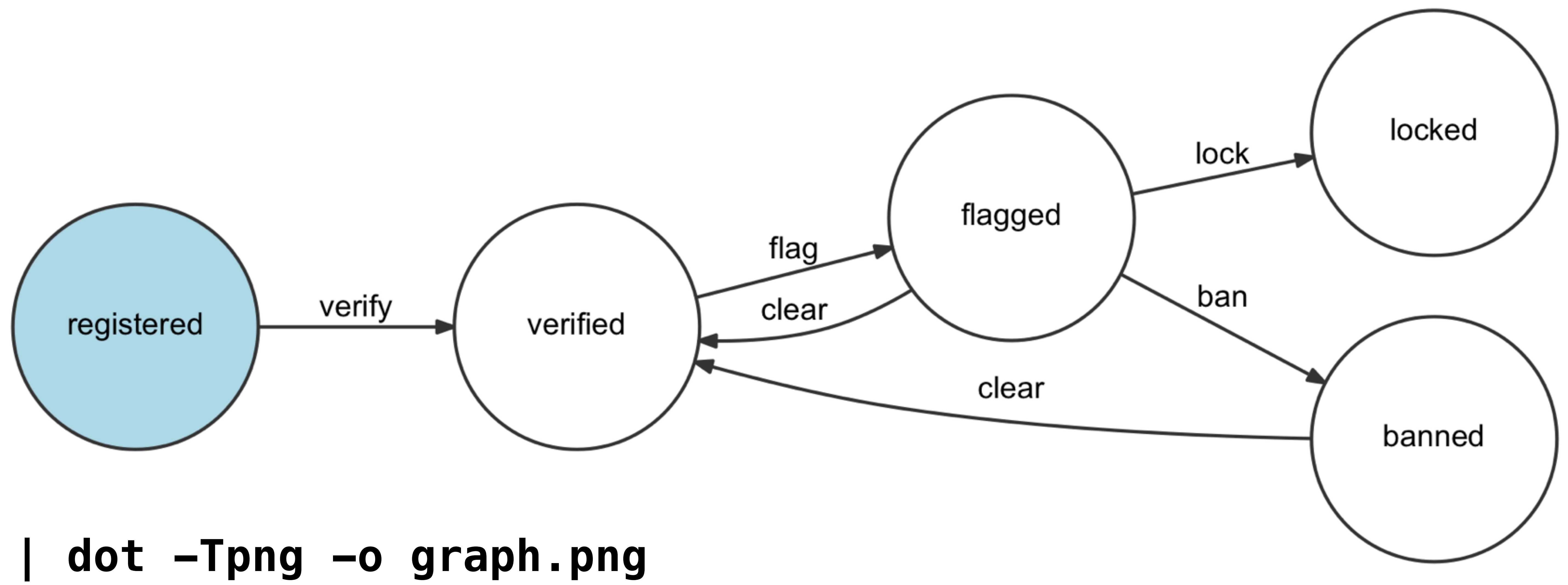


Username	Date registered	Role	Status	Actions
Adam Alister	2012/01/21	Staff	Verified	Flag
Adinah Ralph	2012/06/01	Admin	Banned	
Ajith Hristijan	2012/03/01	Member	Flagged	Clear Ban

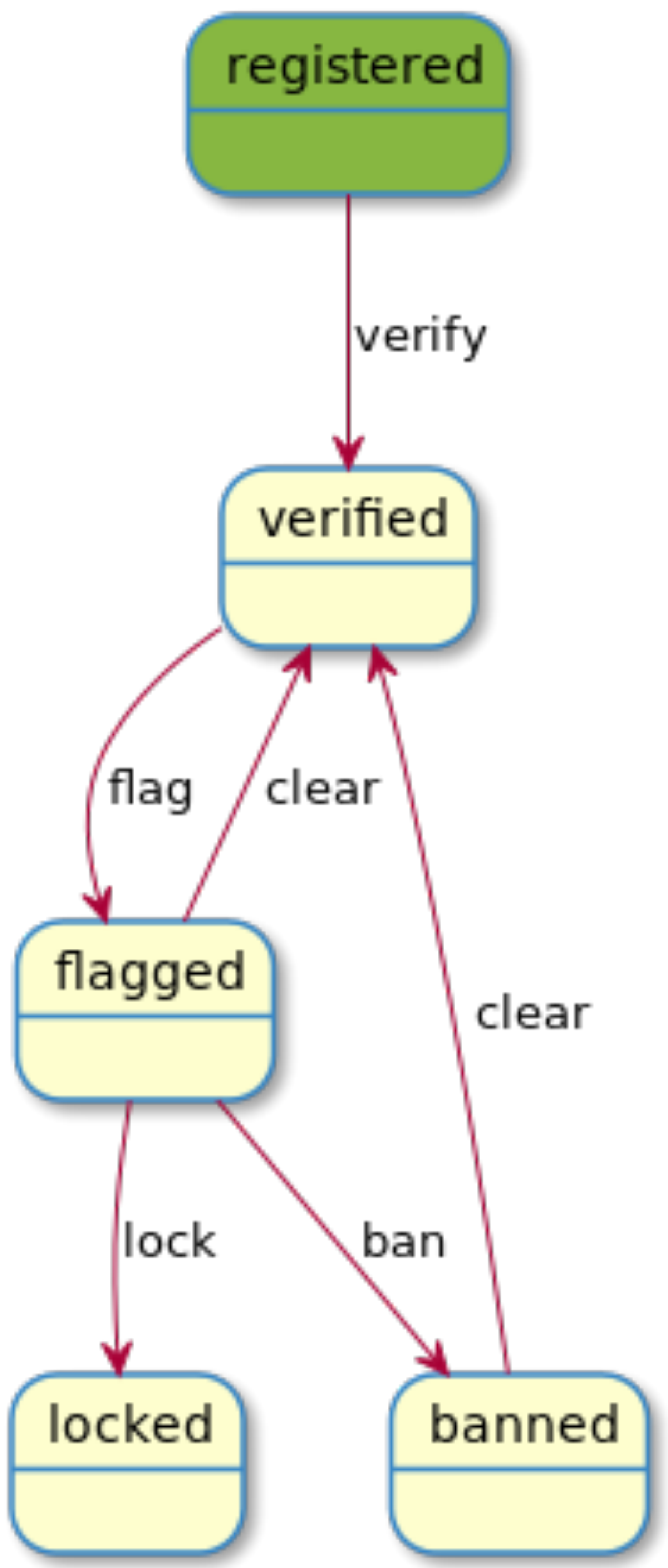
```
{% if workflow_can(user, 'ban') %}
    <a class="btn btn-danger" href=“#">ban</a>
{% endif %}
```

SYMFONY WORKFLOW DUMP

```
$ bin/console workflow:dump user
```



```
| dot -Tpng -o graph.png
```



```
| java -jar plantuml.jar -p > graph.png
```

SYMFONY WORKFLOW TEST COVERAGE

```php
class RegisteredUserTest extends KernelTestCase
{
    /** @var \Symfony\Component\Workflow\Workflow */
    private static $workflow;

    /** @var \App\Entity\User */
    private $user;

    public static function setUpBeforeClass()
    {
        static::bootKernel();
        static::$workflow = static::$kernel->getContainer()->get('workflow.user');
    }

    public function setUp()
    {
        $this->user = new User('Miro Svrtan');
        $this->user->setStatus('registered');
    }
```

SYMFONY WORKFLOW TEST COVERAGE

```php
public function testCanBeVerified()
{
    $this->assertTrue(
        static::$workflow->can($this->user, 'verify')
    );
}

public function testBecomesVerified()
{
    /** @var Marking $verified */
    $verified = static::$workflow->apply($this->user, 'verify');

    $this->assertTrue(
        $verified->has('verified')
    );
}

public function testCanNotBeFlagged()
{
    $this->expectException(NotEnabledTransitionException::class);

    static::$workflow->apply($this->user, 'flag');
}
```

SYMFONY WORKFLOW TEST COVERAGE

```
~ bin/phpunit --testdox
PHPUnit 6.5.8 by Sebastian Bergmann and contributors.

App\Tests\RegisteredUser
 [x] Can be verified
 [x] Becomes verified
 [x] Can not be flagged
 [x] Can not be locked
 [x] Can not be banned
```

POSSIBLE USAGES

BINARY

enable/disable
open/closed

PUBLISHING

articles (draft, approved, published, archived)

PAYMENTS

subscriptions (active, pending renewal, expired)
order (ordered, shipped, canceled, returned, refunded)

GAMES

character levels (peasant, beginner, warrior, lord)
action state (running, jumping)

USAGES IN THE WILD

`sylius/sylius` `using` **`winzou/state-machine`**

So far the list is short

TIPS AND TRICKS

Discuss it with your team/client

Treat them same as serialising

Leave a paper trail

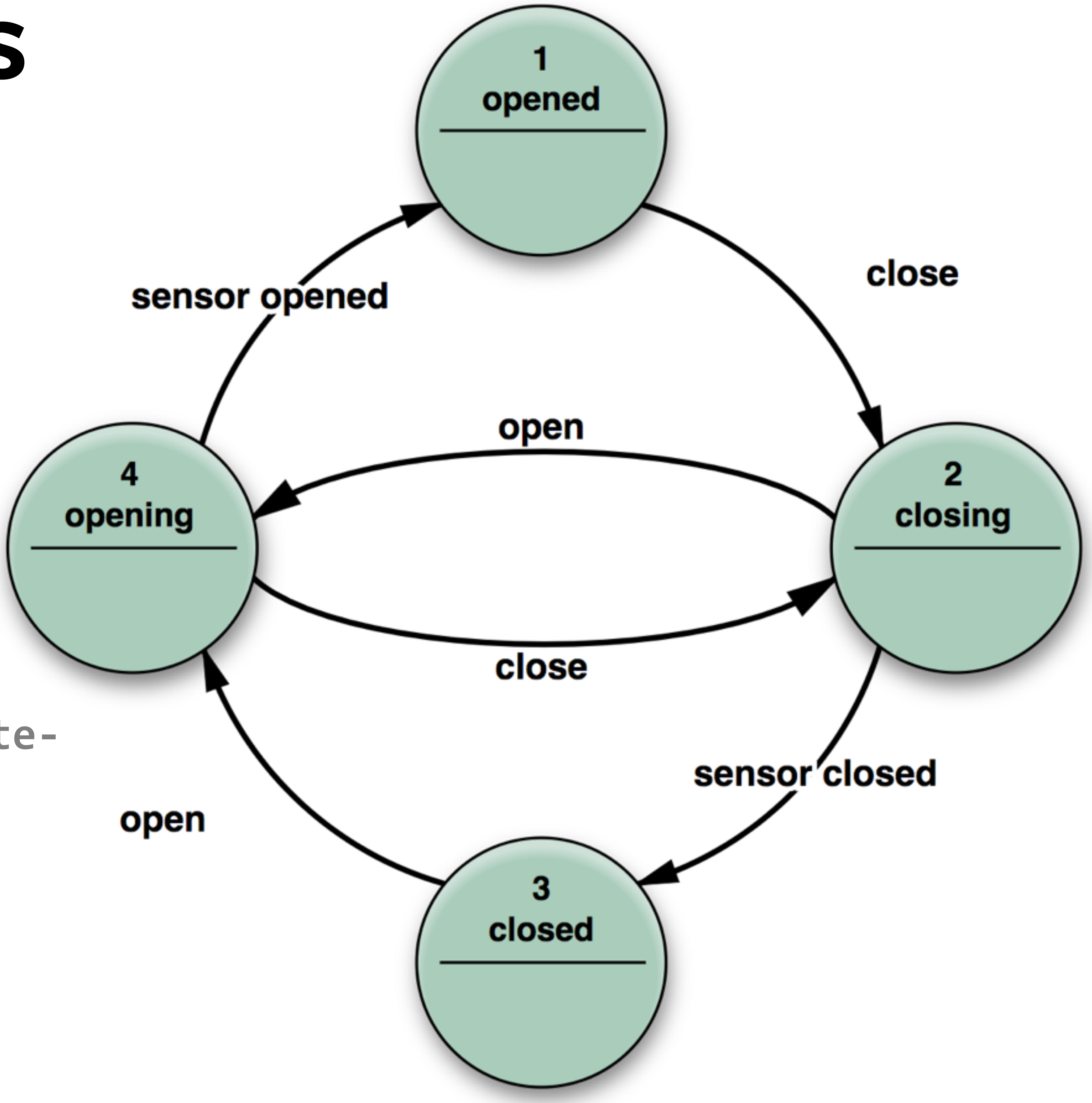
DOWNSIDES

Coupling with framework/ library

Testing becomes (little) complex

FINITE STATE MACHINES

```
# JS
npm install javascript-state-machine

# Python
pip install transitions

# Ruby
gem install state_machine

# PostgreSQL
http://felixge.de/2017/07/27/implementing-state-
machines-in-postgresql.html
```



WHAT DID WE LEARN?

RECAP

State machines are cool

STORE STATE SOMEWHERE

Models are good for storing state and bad for business logic how to transition between states.

Group that logic in one place, if possible. Preferably through state machines.

STATE PATTERN

The object will appear to change its class.

Look at sebastianbergmann/state as an excellent example of state pattern.

STATE MACHINES

Do not reinvent the wheel, use available packages if possible.

Test them as every other piece of code.

Enforce their usage throughout project.

QUESTIONS?

Luka Mužinić
@lmuzinic

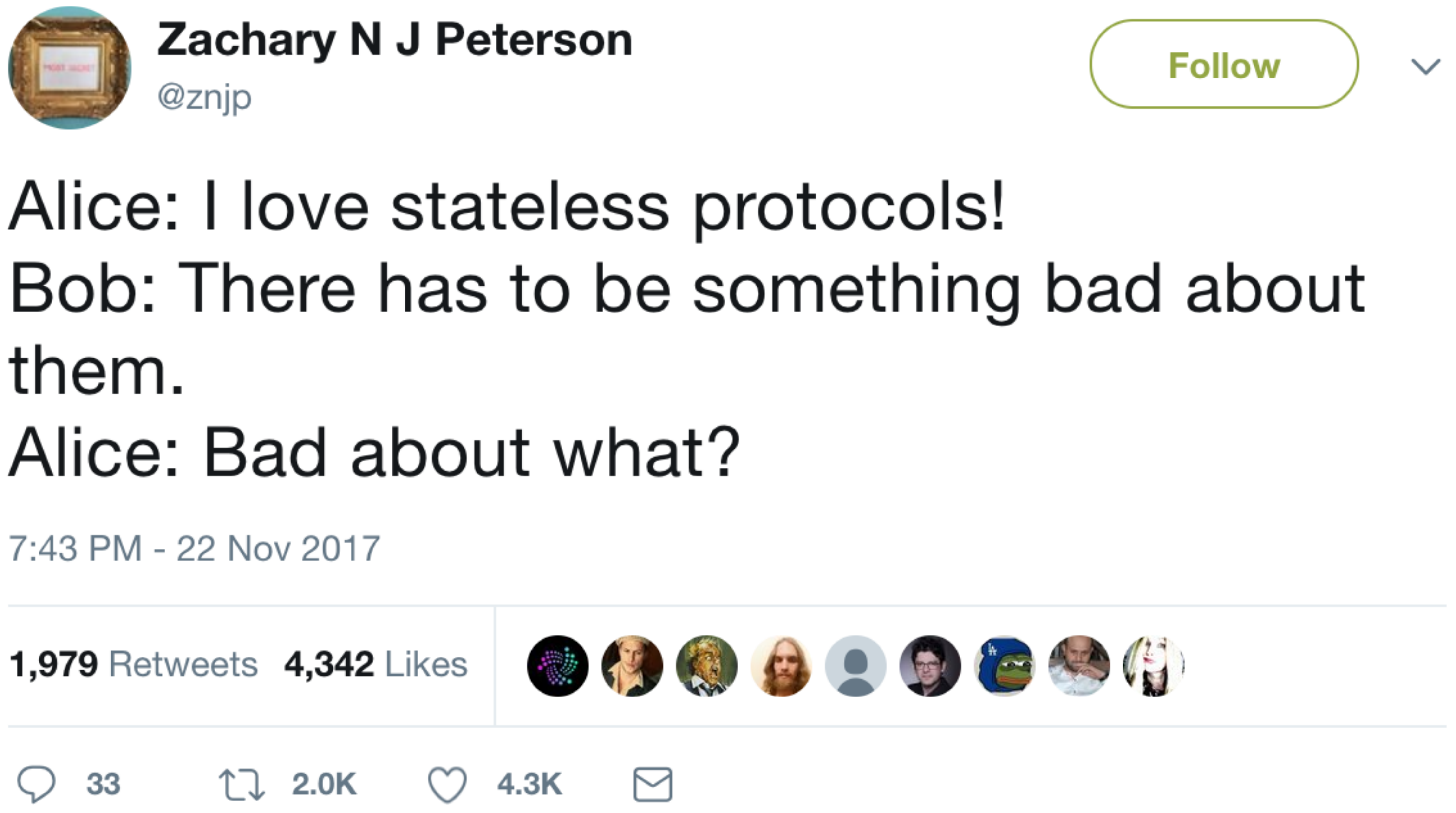
luka.muzinic.net/talks

bit.ly/phpsrbija-codementor

WEB CAMP
ZAGREB 2018.

October 5th & 6th 2018

2018.webcampzg.org



Zachary N J Peterson
@znjp

Follow

Alice: I love stateless protocols!
Bob: There has to be something bad about them.
Alice: Bad about what?

7:43 PM - 22 Nov 2017

1,979 Retweets 4,342 Likes

33 2.0K 4.3K