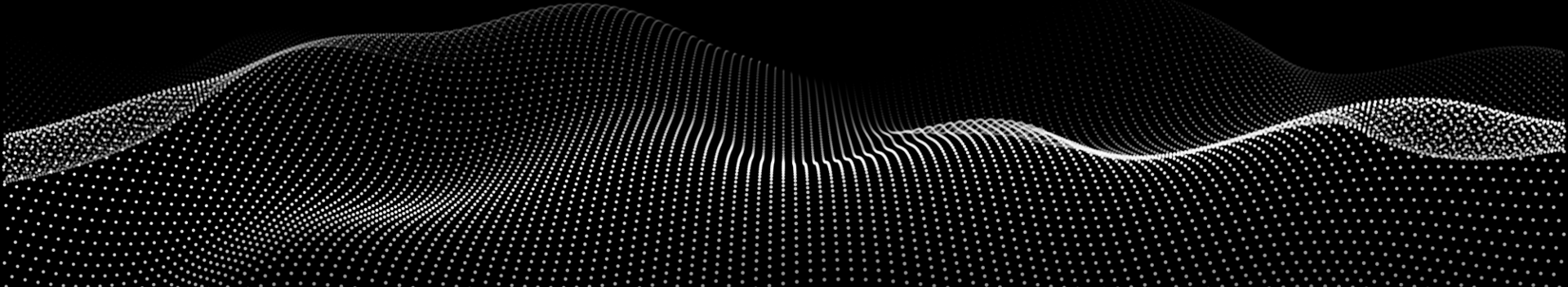


From Chaos to Control: The Concept of State Machines



Ben Kimim?

- Muhammet Us
- Karaca'da çalışıyorum
- Eticaret yazılım süreçlerinden sorumluyum
- Sektörde 15 yıl



 /ussmuhammet

 /muhammetus

State Machine Nedir?

Yazılımda durum makinesi, bir sistemin davranışını ve akışını modellemek için kullanılan bir tasarım kalıbıdır. Bu kalıp, sistemin her an hangi durumda olduğunu ve bu durumdan hangi olaylara tepki vererek hangi yeni duruma geçeceğini tanımlar.

Basitçe söylemek gerekirse, bir durum makinesi, bir sistemin farklı modları arasında geçiş yapmasını ve her modda farklı davranışlar sergilemesini kontrol eder. Bu sayede karmaşık sistemleri daha kolay yönetilebilir ve anlaşılabilir hale getirir.

Durum makineleri, gömülü sistemler, oyunlar, robotlar ve ağ protokolleri gibi birçok farklı alanda kullanılır. Özellikle olay odaklı ve tepkisel sistemler için ideal bir tasarım kalıbıdır.

Durum makinesinin temel bileşenleri şunlardır:

- **Durumlar:** Sistemin her an olabileceği farklı modları temsil eder.
- **Olaylar:** Sistemin durumunu değiştirmeye neden olan dış etkenleri veya iç değişiklikleri temsil eder.
- **Geçişler:** Bir durumdan diğerine geçişi tanımlayan kuralları temsil eder.
- **Eylemler:** Bir durumdayken gerçekleştirilen işlemleri temsil eder.

State Machine Avantajları

- Karmaşık sistemleri basitleştirir.
- Kodun anlaşılmasını ve bakımını kolaylaştırır.
- Hataları bulmayı ve düzeltmeyi kolaylaştırır.
- Sistemin davranışını test etmeyi kolaylaştırır.
- Tasarımı daha modüler hale getirir.



State Machine Dezavantajları

- Basit sistemler için fazla karmaşık olabilir.
- Tasarımı ve uygulaması zaman alabilir.
- Hatalı tasarım hatalara yol açabilir.
- Değişikliklere uyum sağlama zorluğu
- Büyük sistemlerde yönetim zorluğu

State Machine Uygulama Örnekleri

Trello

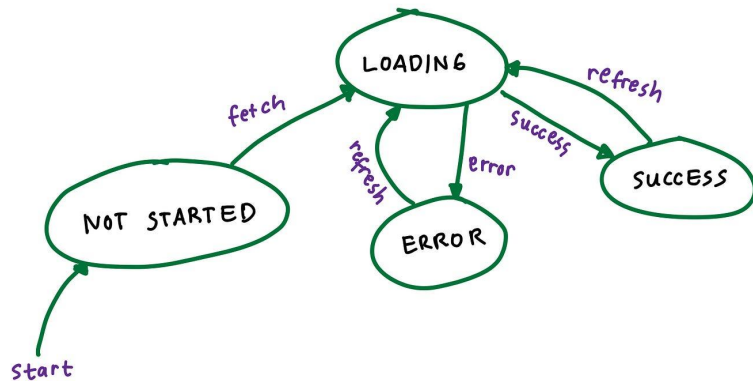
- **Durum 1:** Yeni Kart Oluşturuldu
- **Durum 2:** Kart Panoya Eklendi
- **Durum 3:** Karta Görev Eklendi
- **Durum 4:** Görev Tamamlandı
- **Durum 5:** Kart Arşivlendi

Youtube

- **Durum 1:** Video Yüklendi
- **Durum 2:** Video İşleniyor
- **Durum 3:** Video Yayınlandı

Amazon

- **Durum 1:** Ürün Detayları Görüntülendi
- **Durum 2:** Sepete Eklendi
- **Durum 3:** Sipariş Verildi
- **Durum 4:** Ödeme Onaylandı
- **Durum 5:** İlgili ERP aktarıldı



Gerçek Dünyada State Machine Kullanım Alanları

Gerçek dünyada her eylemimiz, bilinçli ya da bilinçsiz olarak, bir state machine'in adımlarını izler. Gündelik yaşamımızda basit gibi görünen eylemler bile, belirli bir sırayla gerçekleşir. Örneğin, markete giderken, önce evden çıkarız, ardından yolda ilerlerken karar veririz, alışveriş listemize bakarız, ürünleri seçeriz, ödeme yaparız ve en sonunda evimize geri döneriz. Bu süreçte her adım, bir öncekine bağlı olarak devam eder ve bize istediğimiz sonucu sağlar.

Aynı şekilde, benzin doldururken veya işe giderken de benzer bir model izlenir. Bu durumlar da belirli bir başlangıç noktasından, bir dizi adımı takip ederek bir sonuca ulaşmayı gerektirir. Her adımın birbirini izlemesi ve belirli bir sırayla gerçekleşmesi, günlük yaşamımızı yönetirken otomatik olarak işler. Dolayısıyla, state machine modeli, karmaşık gibi görünen birçok sürecin temelinde yatan düzenleyici prensiplerden biridir.

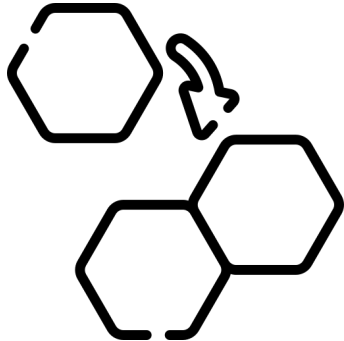
State Machine oluşturma için neler gerekli?

1. **Sistemin Tanımlanması:** İlk adım, sistemin işlevlerini ve davranışlarını net bir şekilde tanımlamaktır. Bu, sistemin farklı durumlarını, her durumdan geçebileceği geçişleri ve her geçişin tetikleyicilerini belirlemeyi içerir.
2. **Durum Makinesi Türünün Seçimi:** Farklı durum makinesi türleri farklı problemlere ve sistemlere uygundur. Sisteminizin karmaşıklığına ve gereksinimlerine göre en uygun türü seçmeniz gerekir.
3. **Durumların Tanımlanması:** Sistemin tüm olası durumlarını belirleyin. Her durum, sistemin belirli bir anlık görüntüsünü temsil eder.
4. **Geçişlerin Tanımlanması:** Her durumdan geçebileceği geçişleri belirleyin. Her geçiş, bir durumdan diğerine geçişi tetikleyen bir olayı veya koşulu temsil eder.
5. **Tetikleyicilerin Tanımlanması:** Her geçişi tetikleyen olayları veya koşulları belirleyin. Tetikleyiciler, sensörlerden gelen veriler, kullanıcı girdileri veya sistemdeki diğer değişiklikler gibi çeşitli kaynaklardan gelebilir.
6. **Eylemlerin Tanımlanması:** Her geçişle ilişkili eylemleri belirleyin. Eylemler, sistemin durumunu değiştiren veya sistemin işlevlerini gerçekleştiren işlemleri temsil eder.

State Machine oluşturma için neler gerekli?

1. **Durum Makinesi Tasarımı:** Seçtiğiniz durum makinesi türüne uygun bir şekilde durumları, geçişleri, tetikleyicileri ve eylemleri gösteren bir diyagram oluşturun.
2. **Durum Makinesi Uygulaması:** Seçtiğiniz programlama diline ve kütüphanesine göre durum makinesini kodlayın.
3. **Test ve Hata Ayıklama:** Durum makinesinin tüm olası senaryolar için doğru şekilde çalıştığından emin olmak için kapsamlı bir şekilde test edin ve hata ayıklayın.

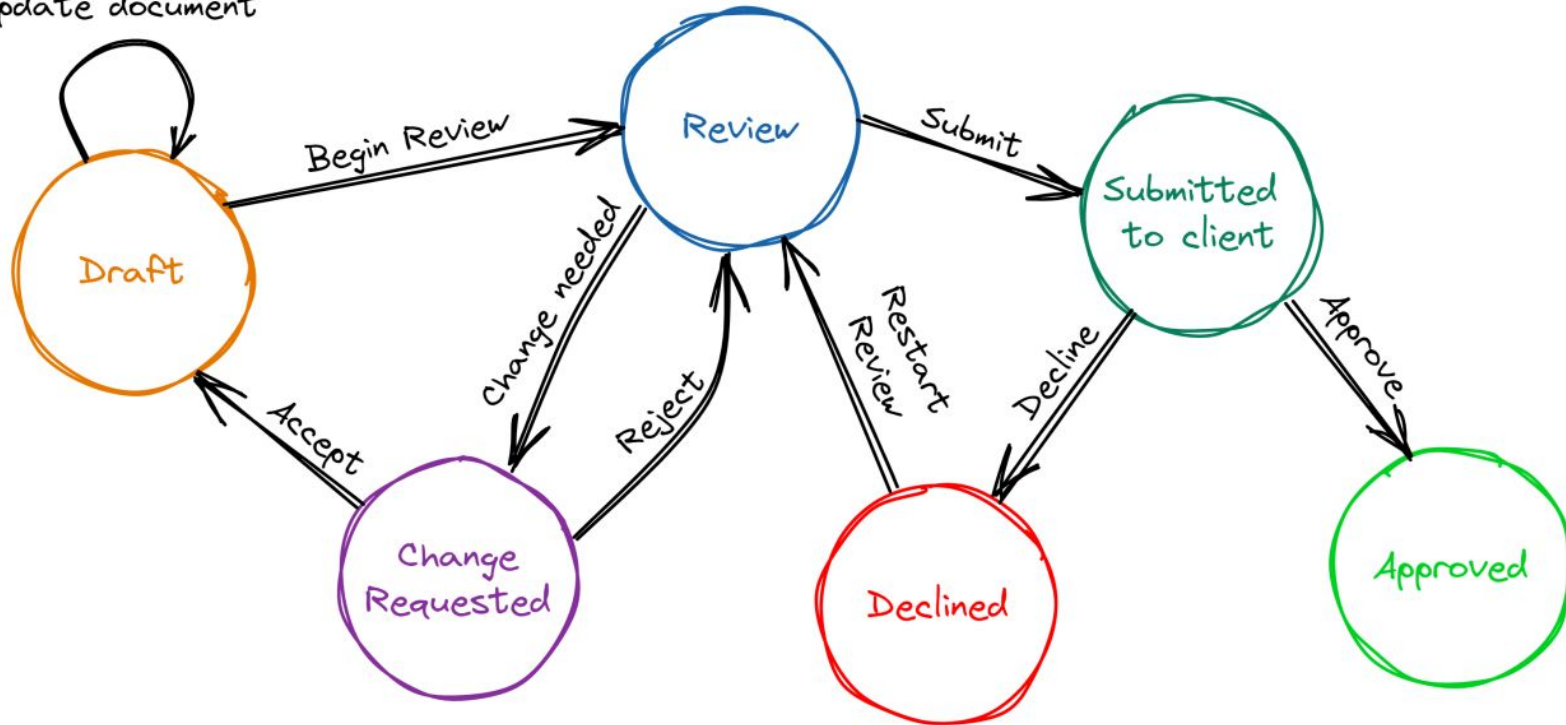
Durum makineleri oluştururken dikkat edilmesi gereken bazı önemli noktalar şunlardır:



- **Basitlik:** Durum makinesini mümkün olduğunca basit tutmaya çalışın. Karmaşık durum makineleri daha zor anlaşılır, yönetilir ve hata ayıklanır.
- **Modülerlik:** Büyük ve karmaşık durum makinelerini daha küçük, daha yönetilebilir alt makinesine bölün.
- **Açıklık:** Durum makinelerini açık ve belgelendirilmiş bir şekilde kodlayın, böylece diğer geliştiricilerin kolayca anlayabilmeleri ve sürdürebilmeleri.
- **Test:** Durum makinesinin tüm olası senaryolar için doğru şekilde çalıştığından emin olmak için kapsamlı bir şekilde test edin.

State Machine Tasarımı ve Geliştirme Süreci

Update document



Coding : Part 1



```
class ShoppingCartStates {
    private string $state = "EmptyCart";

    public function addProduct() {
        if ($this->state === "EmptyCart") {
            $this->state = "AddingProduct";
            echo "Ürün eklendi.\n";
        } elseif ($this->state === "RemoveProduct") {
            $this->state = "AddingProduct";
            echo "Ürün eklendi.\n";
        } elseif ($this->state === "AddingProduct") {
            echo "Ürün eklemek için uygun değil.\n";
        } elseif ($this->state === "CompleteShopping") {
            echo "Alışveriş zaten tamamlandı.\n";
        }
    }

    public function removeProduct() {
        if ($this->state === "AddingProduct") {
            $this->state = "RemoveProduct";
            echo "Ürün kaldırıldı.\n";
        } elseif ($this->state === "EmptyCart") {
            echo "Ürün kaldırmak için uygun değil.\n";
        } elseif ($this->state === "RemoveProduct") {
            echo "Ürün kaldırmak için uygun değil.\n";
        } elseif ($this->state === "CompleteShopping") {
            echo "Alışveriş zaten tamamlandı.\n";
        }
    }

    public function completeShopping() {
        if ($this->state === "AddingProduct") {
            $this->state = "CompleteShopping";
            echo "Alışveriş tamamlandı.\n";
        } elseif ($this->state === "RemoveProduct") {
            $this->state = "CompleteShopping";
            echo "Alışveriş tamamlandı.\n";
        } elseif ($this->state === "EmptyCart") {
            echo "Sepetinizde ürün bulunmamaktadır.\n";
        } elseif ($this->state === "CompleteShopping") {
            echo "Alışveriş zaten tamamlandı.\n";
        }
    }
}
```

Coding : Part 2

```
class ShoppingCartStates {
    private string $state = "EmptyCart";

    public function addProduct() {
        echo match ($this->state) {
            "EmptyCart", "RemoveProduct" => $this->state = "AddingProduct" and "Ürün eklendi.",
            "AddingProduct" => "Ürün eklemek için uygun değil.",
            "CompleteShopping" => "Alışveriş zaten tamamlandı.",
        };
    }

    public function removeProduct() {
        echo match ($this->state) {
            "AddingProduct" => $this->state = "RemoveProduct" and "Ürün kaldırıldı.",
            "EmptyCart", "RemoveProduct" => "Ürün kaldırmak için uygun değil.",
            "CompleteShopping" => "Alışveriş zaten tamamlandı.",
        };
    }

    public function completeShopping() {
        echo match ($this->state) {
            "AddingProduct", "RemoveProduct" => $this->state = "CompleteShopping" and "Alışveriş
tamamlandı."; "EmptyCart" => "Sepetinizde ürün bulunmamaktadır.",
            "CompleteShopping" => "Alışveriş zaten tamamlandı.",
        };
    }
}
```

Coding : Part 3

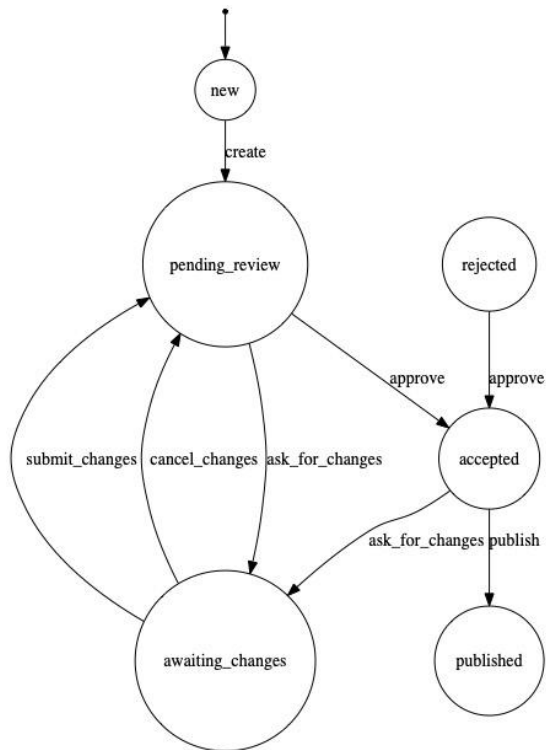
```
$sepet = new ShoppingCartStates();

// Durum: BoşSepet
echo "Durum: BoşSepet\n";
$sepet->addProduct(); // Ürün eklendi.
$sepet->removeProduct(); // Ürün kaldırmak için uygun değil. // [Hata]: Ürün kaldırabilmek için önce ürün eklemek gerekir.
$sepet->completeShopping(); // Sepetinizde ürün bulunmamaktadır. // [Hata]: Alışverişi tamamlayabilmek için önce ürün eklemek gerekir.

// Durum: ÜrünEkle
echo "Durum: ÜrünEkle";
$sepet->addProduct(); // Ürün eklemek için uygun değil. // [Hata]: Ürün eklemek için farklı bir durumda olmalısınız.
$sepet->removeProduct(); // Ürün kaldırıldı.
$sepet->completeShopping(); // Alışveriş tamamlandı.

// Durum: AlışverişiTamamla
echo "Durum: AlışverişiTamamla";
$sepet->addProduct(); // Alışveriş zaten tamamlandı. // [Hata]: Alışveriş tamamlandıktan sonra ürün ekleyemezsiniz.
$sepet->removeProduct(); // Alışveriş zaten tamamlandı. // [Hata]: Alışveriş tamamlandıktan sonra ürün çıkaramazsınız.
$sepet->completeShopping(); // Alışveriş zaten tamamlandı. // [Hata]: Alışveriş zaten tamamlanmış durumda.
```

Coding : Part 4



```
return [
  'transitions' => [
    'create' => [
      'from' => ['new'],
      'to' => 'pending_review',
    ],
    'ask_for_changes' => [
      'from' => ['pending_review', 'accepted'],
      'to' => 'awaiting_changes',
      'metadata' => ['title' => 'Ask for changes'],
    ],
    'cancel_changes' => [
      'from' => ['awaiting_changes'],
      'to' => 'pending_review',
    ],
    'submit_changes' => [
      'from' => ['awaiting_changes'],
      'to' => 'pending_review',
    ],
    'approve' => [
      'from' => ['pending_review', 'rejected'],
      'to' => 'accepted',
    ],
    'publish' => [
      'from' => ['accepted'],
      'to' => 'published',
    ],
  ],
];
```

Farklı State Machine Kütüphaneleri ve Çerçeveleri

<https://github.com/winzou/state-machine>

master	1 Branch	15 Tags	Go to file	Add file	Code
Ichrusciel	Merge pull request #58 from loic425/symfony-7	b7ced0a · 4 months ago	63 Commits		
.github/workflows	Add support for Symfony 7	4 months ago			
examples	Merge pull request #26 from winzou/callback	8 years ago			
spec/SM	Add Symfony5 support	4 years ago			
src/SM	Add Symfony5 support	4 years ago			
.gitignore	Initial commit	10 years ago			
LICENSE	Initial commit	10 years ago			
README.md	[README] Change badge from travis to GH	2 years ago			
composer.json	Add support for Symfony 7	4 months ago			

README MIT license

A very lightweight yet powerful PHP state machine

Define your states, define your transitions and your callbacks: we do the rest. The era of hard-coded states is over!

Build passing

Installation (via composer)

```
{
  "require": {
    "winzou/state-machine": "~0.1"
  }
}
```

<https://github.com/yohang/Finite>

Finite

Public

Watch

59

master

12 Branches

20 Tags

Go to file

t

Add file

Code

Yohan Giarelli

Removed keys from transition array

32cc53c · 2 years ago

225 Commits

docs

Added documentation for using events

9 years ago

examples

Merge pull request #51 from yohang/refactor-callback-ha...

9 years ago

src/Finite

Removed keys from transition array

2 years ago

tests

Added a way to retrieve all graphs for a class

2 years ago

.gitignore

Initied a Sphinx documentation

9 years ago

.gitmodules

Removed old docs submodule

9 years ago

.phpunit.result.cache

Added a way to retrieve all graphs for a class

2 years ago

.travis.yml

Removed PHP7.2 from build matrix

2 years ago

LICENSE

Added README and LICENSE

12 years ago

README.md

Updated README

9 years ago

composer.json

Added a way to retrieve all graphs for a class

2 years ago

phpunit.xml.dist

Updated for PHP8 & Symfony 6

2 years ago

README MIT license

Finite, A Simple PHP Finite State Machine

Finite is a Simple State Machine, written in PHP. It can manage any Stateful object by defining states and transitions between these states.

build canceled stable 1.3.5 downloads 2.81 M license MIT Scrutinizer 9.44 coverage 89 % Sencin alncinht

Sebdesign/State Machine - Laravel



sebdesign / laravel-state-machine

Issues 2 Pull requests 1 Actions Projects Security Insights

laravel-state-machine Public Watch 11

master 4 Branches 35 Tags Go to file Add file Code

sebdesign Merge pull request #71 from laravel-shift/l11-compatibility 9d4b37f · 2 months ago 133 Commits

.github	Fix tests	2 months ago
config	Add a MetadataStore to fetch metadata from graphs, stat...	6 years ago
src	compatibility update of Dispatcher with EventDispatcherI...	4 months ago
tests	Apply fixes from StyleCI	3 years ago
.editorconfig	Update test workflows	2 years ago
.gitattributes	Switch to GitHub Actions	4 years ago
.gitignore	Added command to generate image of graph	4 years ago
.scrutinizer.yml	Update .scrutinizer.yml	3 years ago
.styleci.yml	Add support for Laravel 8	4 years ago
CHANGELOG.md	Update changelog	2 months ago
LICENSE.md	Initial commit	8 years ago
README.md	Remove syntax highlight from debug command output	2 years ago
composer.json	Allow symfony/process v7	2 months ago
phpunit.xml.dist	Switch to GitHub Actions	4 years ago

README MIT license

on/sebdesign/laravel-state-machine

Basit Kullanım

```
<?php
```

```
// Herhangi bir eloquent modelimiz
```

```
$article = App\Article::find($id);
```

```
// Tanımlayacağımız graph
```

```
// Facade kullanabiliriz
```

```
$stateMachine = StateMachine::get($article, 'simple');
```

```
// Şuan bulunduğu state
```

```
$stateMachine->getState();
```

```
// Geçiş yapabileceği stateler
```

```
$stateMachine->getPossibleTransitions();
```

```
// Approve geçişine izin veriyor mu?
```

```
$stateMachine->can('approve');
```

```
// Approve geçişini yap eğer geçemez ise throws an SM\SMException
```

```
$stateMachine->apply('publish');
```

```
// Approve geçişini yap eğer geçemez ise false döner ve exception fırlatmaz
```

```
$stateMachine->apply('publish', true);
```

Callbacks

```

<?php

'callbacks' => [
    // Bir geçiş yaptığımızda çalışacak olaylar
    'guard' => [
        'guard_on_submitting' => [
            // Belirli bir geçişte geri çağırıcı çağırır
            'on' => 'submit_changes',

            // Bu sınıfın yöntemini çağırır
            'do' => ['MyService', 'handle'],

            // Geri çağrı için argümanlar
            'args' => ['object'],
        ],
    ],
],

class MyService {

    public function handle(Article $article){
        //....
    }
}
```

Gates and Policies

```
<?php
```

```
'callbacks' => [  
    'guard' => [  
        'guard_on_approving' => [  
            // Belirli geçişi tetikle  
            'on' => 'approve',  
            // bu şekilde çağırmanızla aynı işi görür; Gate:allows('approve', $article)  
            'can' => 'approve',  
        ],  
    ],  
],  
];
```

```
Gate::define('approve', function (User $user, Article $article) {  
    //  
});
```

Events

```
<?php

protected $listen = [
    SMEvents::TEST_TRANSITION => [
        \App\Listeners\CheckTransition::class,
    ],
    SMEvents::PRE_TRANSITION => [
        \App\Listeners\BeforeTransition::class,
    ],
    SMEvents::POST_TRANSITION => [
        \App\Listeners\AfterTransition::class,
    ],
    'winzou.state_machine.*' => [
        \App\Listeners\Transition::class,
    ],
];
```

```
<?php

// Onay Tarihini kontrol et
Event::listen(SMEvents::TEST_TRANSITION, function (TransitionEvent $event) {
    $context = $event->getContext();

    if ($context['approved_at']->isPast()) {
        $event->setRejected();
    }
});

// approve stateine geçiş yapılabilir mi kontrol et
$stateMachine->can('approve', ['approved_at' => now()]);
```

- Belirli durumlarda db güncellemeleri
- Loglama işlemleri
- Rulesetler
- Queue işlemleri

Meta Data

```
<?php
```

```
[
    'graphA' => [
        'class' => App\Article::class,

        'metadata' => [
            'title' => 'Article State Machine',
        ],

        'states' => [
            [
                'name' => 'pending_review',
                'metadata' => ['title' => 'Pending Review'],
            ],
        ],

        'transitions' => [
            'ask_for_changes' => [
                'from' => ['pending_review'],
                'to' => 'accepted',
                'metadata' => ['title' => 'Ask for changes'],
            ],
        ],
    ],
];
```

```
$ php artisan winzou:state-machine:debug simple
```

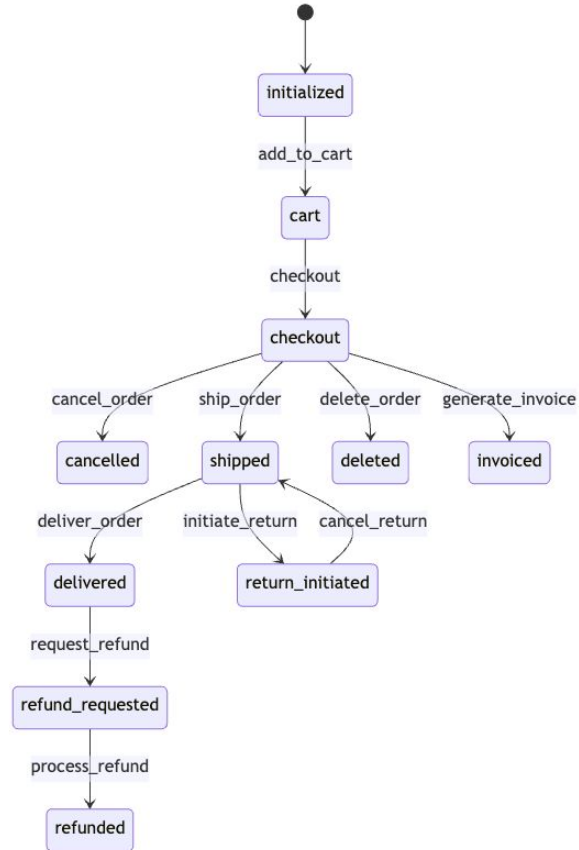
Configured States:		Metadata:
new		
pending_review		title: Pending Review
awaiting_changes		
accepted		
published		
rejected		

Transition	From(s)	To
create	new	pending_review
ask_for_changes	pending_review accepted	awaiting_changes
cancel_changes	awaiting_changes	pending_review
submit_changes	awaiting_changes	pending_review
approve	pending_review rejected	accepted
publish	accepted	published

Guard Callbacks	Satisfies	Do	Args
guard_on_submitting	On: submit_changes	MyClass::handle()	object
guard_on_approving	On: approve	Gate::check("approve")	

Before Callbacks	Satisfies	Do	Args
log_before_approval	On: approve	Log::info()	"approving article", object

Örnek State Tasarımları - Checkout



Örnek State Tasarımları - Coding

```
<?php
[
    'Basket' => [
        'class' => App\Basket::class,

        'metadata' => [
            'title' => 'Basket State Machine',
        ],
    ],

    'transitions' => [
        'AddRemoveQuantity' => [
            'from' => ['Adding', 'Removing'],
            'to' => 'Adding',
            'metadata' => ['title' => 'Add/Remove Quantity'],
        ],
        'DeleteItem' => [
            'from' => ['Adding', 'Removing', 'Deleting'],
            'to' => 'Deleting',
            'metadata' => ['title' => 'Delete Item'],
        ],
        'StartPayment' => [
            'from' => ['Adding', 'Removing', 'Deleting'],
            'to' => 'PaymentStart',
            'metadata' => ['title' => 'Start Payment'],
        ],
        'CancelPayment' => [
            'from' => ['PaymentStart'],
            'to' => 'Adding',
            'metadata' => ['title' => 'Cancel Payment'],
        ],
        'SuccessfulPayment' => [
            'from' => ['PaymentStart'],
            'to' => 'PaymentCompleted',
            'metadata' => ['title' => 'Successful Payment'],
        ],
        'CancelShopping' => [
            'from' => ['Adding', 'Removing', 'Deleting', 'PaymentStart', 'PaymentCompleted'],
            'to' => 'Cancelled',
            'metadata' => ['title' => 'Cancel Shopping'],
        ],
        'RequestRefund' => [
            'from' => ['PaymentCompleted'],
            'to' => 'Refunding',
            'metadata' => ['title' => 'Request Refund'],
        ],
    ],
],
];
```

```
<?php
'transitions' => [
    'AddToCart' => [
        'from' => ['initialized'],
        'to' => 'cart',
        'metadata' => ['title' => 'Add to Cart'],
    ],
    'Checkout' => [
        'from' => ['cart'],
        'to' => 'checkout',
        'metadata' => ['title' => 'Checkout'],
    ],
    'CancelOrder' => [
        'from' => ['checkout'],
        'to' => 'cancelled',
        'metadata' => ['title' => 'Cancel Order'],
    ],
    'ShipOrder' => [
        'from' => ['checkout'],
        'to' => 'shipped',
        'metadata' => ['title' => 'Ship Order'],
    ],
    'DeliverOrder' => [
        'from' => ['shipped'],
        'to' => 'delivered',
        'metadata' => ['title' => 'Deliver Order'],
    ],
    'RequestRefund' => [
        'from' => ['delivered'],
        'to' => 'refund_requested',
        'metadata' => ['title' => 'Request Refund'],
    ],
    'ProcessRefund' => [
        'from' => ['refund_requested'],
        'to' => 'refunded',
        'metadata' => ['title' => 'Process Refund'],
    ],
    'DeleteOrder' => [
        'from' => ['checkout'],
        'to' => 'deleted',
        'metadata' => ['title' => 'Delete Order'],
    ],
    'GenerateInvoice' => [
        'from' => ['checkout'],
        'to' => 'invoiced',
        'metadata' => ['title' => 'Generate Invoice'],
    ],
    'InitiateReturn' => [
        'from' => ['shipped'],
        'to' => 'return_initiated',
        'metadata' => ['title' => 'Initiate Return'],
    ],
    'CancelReturn' => [
        'from' => ['return_initiated'],
        'to' => 'shipped',
        'metadata' => ['title' => 'Cancel Return'],
    ],
],
]
```

Örnek State Tasarımları - Coding

```
$basket = new Basket();
$basket->name = 'My Basket';
$basket->save();

echo "Mevcut Sepet Durumu: " . $basket->getCurrentState() . "";

// Geçiş 1: initialized durumundan refund_request durumuna geçiş
$transition1 = 'refund_request';
if ($basket->canTransition($transition1)) {
    $basket->applyTransition($transition1);
    echo "Yeni Sepet Durumu (Geçiş 1): " . $basket->getCurrentState() . "";
} else {
    echo "Geçiş 1 ($transition1) uygulanamaz. Hata nedeni: Bu geçiş, initialized durumunda yapılamaz.";
}

// Geçiş 2: checkout durumundan shipping durumuna geçiş
$transition2 = 'shipping';
if ($basket->canTransition($transition2)) {
    $basket->applyTransition($transition2);
    echo "Yeni Sepet Durumu (Geçiş 2): " . $basket->getCurrentState() . "";
} else {
    echo "Geçiş 2 ($transition2) uygulanamaz. Hata nedeni: Sepetiniz ödeme yapılmamış durumda olduğu için shipping durumuna geçiş yapılamaz.";
}
```

Örnek State Tasarımları - Coding

```
// Geçiş 3: checkout durumundan invoiced durumuna geçiş
$transition3 = 'invoiced';
if ($basket->canTransition($transition3)) {
    $basket->applyTransition($transition3);
    echo "Yeni Sepet Durumu (Geçiş 3): " . $basket->getCurrentState() . " ";
} else {
    echo "Geçiş 3 ($transition3) uygulanamaz. Hata nedeni: Sepetiniz henüz ödeme yapılmamış durumda olduğu için invoiced durumuna geçiş yapılamaz.";
}

// Geçiş 4: invoiced durumundan checkout durumuna geçiş
$transition4 = 'checkout';
if ($basket->canTransition($transition4)) {
    $basket->applyTransition($transition4);
    echo "Yeni Sepet Durumu (Geçiş 4): " . $basket->getCurrentState() . " ";
} else {
    echo "Geçiş 4 ($transition4) uygulanamaz. Hata nedeni: Sepetinizin invoiced durumunda olduğu için tekrar checkout durumuna geçiş yapılamaz.";
}

// Geçiş 5: deleted durumundan shipped durumuna geçiş
$transition5 = 'shipped';
if ($basket->canTransition($transition5)) {
    $basket->applyTransition($transition5);
    echo "Yeni Sepet Durumu (Geçiş 5): " . $basket->getCurrentState() . " ";
} else {
    echo "Geçiş 5 ($transition5) uygulanamaz. Hata nedeni: Sepetiniz deleted durumunda olduğu için shipped durumuna geçiş yapılamaz.";
}
```

TEŞEKKÜRLER!