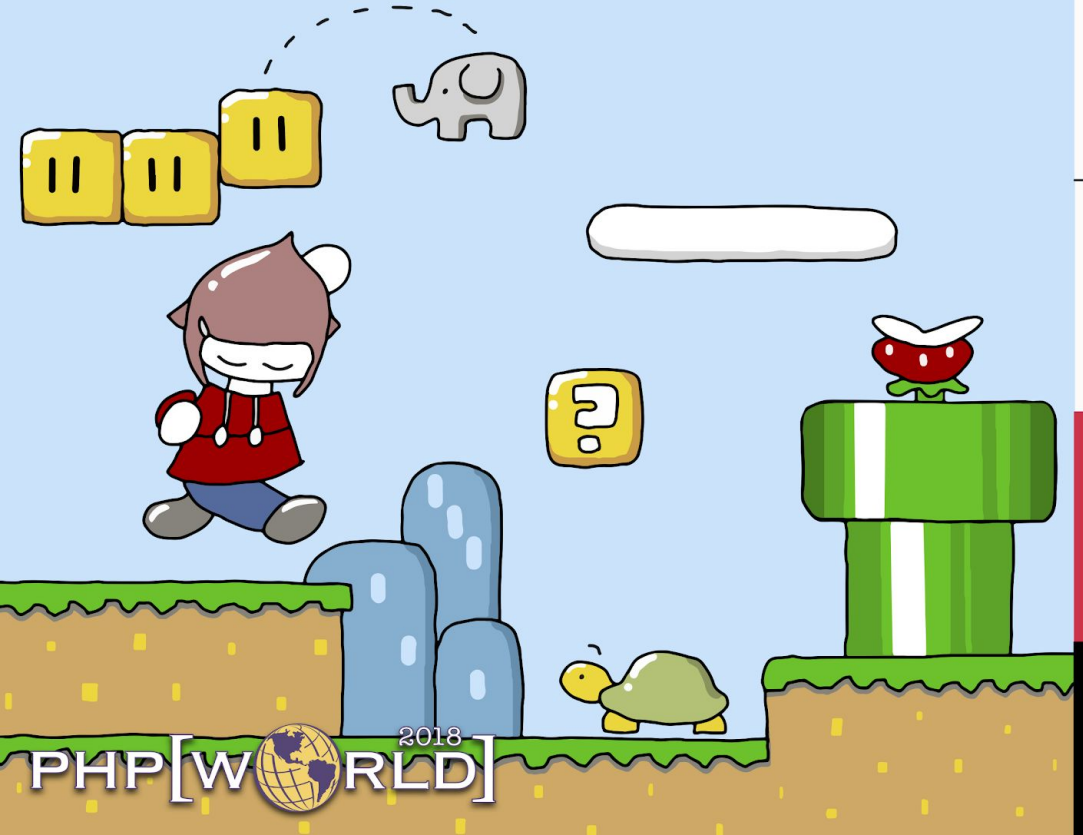
Thanks to Our Sponsors



AUTOMATTIC

Square

concrete5

Magento

twilio

OSMI

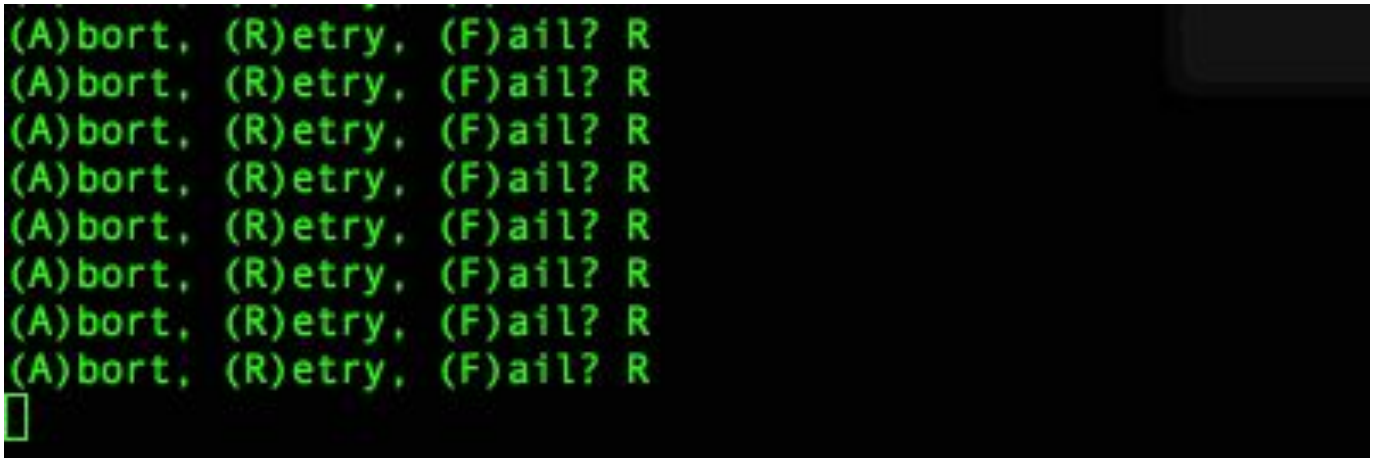
SOLUTIONSTREET
Quality Software Consulting

Resilient PHP

Brian Sanders php[world] 2018

http://tiny.cc/phpworld2018

"Resilient" Software?



~~"Resilient"~~ "Reliable" Software?

~~"Resilient"~~ "Reliable" Software?

"Resilient" Software

"Resilient" Software

A resilient PHP app has been designed to respond in a *consumer-friendly way* to atypical operating conditions.

"Resilient" Software

A resilient PHP app has been designed to respond in a *consumer-friendly way* to atypical operating conditions.

Resiliency implies reliability: the system should always be available, consistent, and accurate.

Resilient Availability

- The system can perform its functions properly under normal load
- **The system may *degrade* (in a controlled way) under extraordinary load or adverse operating conditions, but it returns to full functionality once the problems have been resolved**
- The performance of the system doesn't degrade with time

Controlled Degradation

- Raise a *specific* exception or error to the caller
- Return a stale response
- Return a partial response
- Return an asynchronous response

Scenario: User Details Controller

- PHP controller action to return user details
  - `GET /users/123`
- Dedicated application DB stores username, last login, social media handle
- Call to external web API provides list of most recent social media posts
- JSON output

```
{
  "id": 123,
  "username": "spartacus",
  "last_login": "2018-05-25T08:55:00.222Z",
  "handle": "@spartacus2018",
  "recent_posts": [
    { "uri": "http://soci.al/spartacus2018/post/5442a3",
      "text": "Does anyone know where I can get a cronut at this hour? Asking for a friend",
      "posted": "2018-05-29T14:04:19.811Z" }, ...
  ]
}
```

Scenario: User Details Controller

```php
class UserDetailsController extends Controller
{
    ...
    public function show($id): array
    {
        $user = \App\User::findOrFail($id);
        $user_attribs = $user->only(["id", "username", "last_login", "handle"]);
        if ($user->handle) {
            $posts = $this->social_site_client->recentPostsForUser($user->handle);
        } else {
            $posts = [];
        }
        return $user_attribs + ["recent_posts" => $posts];
    }
}
```

Scenario: User Details Controller

```php
class SocialSiteClient
{
    public function recentPostsForUser(string $handle, int $max = 5): array
    {
        $url = "http://soci.al/api/users/"-.urlencode($handle)."?include=posts&limit={$max}";
        $opts = [
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_HTTPHEADER => ["Content-Type: application/json"],
            CURLOPT_URL => $url
        ];
        $curl = curl_init();
        curl_setopt_array($curl, $opts);
        $response = curl_exec($curl);
        if ($response === false) {
            throw new \Exception("cURL error: " . curl_error($curl));
        }
        return json_decode($response, true)["posts"];
    }
}
```

Demo: User Details Controller

Scenario: User Details Controller

```
public function show($id): array
{
    $user = \App\User::findOrFail($id);
    $user_attribs = $user->only(["id", "username", "last_login", "handle"]);
    if ($user->handle) {
        $posts = $this->social_site_client->recentPostsForUser($user->handle);
    } else {
        $posts = [];
    }
    return $user_attribs + ["recent_posts" => $posts];
}
```

- The system may *degrade* (in a controlled way) under extraordinary load or adverse operating conditions

Scenario: User Details Controller

- PHP controller action to return user details
  - `GET /users/123`
- Dedicated application DB stores username, last login, social media handle
- Call to external web API provides list of most recent social media posts
  - If the social media posts are not available, we should degrade our response by omitting that detail.

Scenario: User Details Controller

- Call to external web API provides list of most recent social media posts
  - If the social media posts are not available, we should degrade our response by omitting that detail.
- JSON output


```
{
  "id": 123,
  "username": "spartacus",
  "last_login": "2018-05-25T08:55:00.222Z",
  "handle": "@spartacus2018",
  "recent_posts": [
    { "uri": "http://soci.al/spartacus2018/post/5442a3",
      "text": "Does anyone know where I can get a cronut at this hour? Asking for a friend",
      "posted": "2018-05-29T14:04:19.811Z" }, ...
  ]
}
```

Scenario: User Details Controller

- Call to external web API provides list of most recent social media posts
  - If the social media posts are not available, we should degrade our response by omitting that detail.
- JSON output

```
{
  "status": "complete",
  "details": {
    "id": 123,
    "username": "spartacus",
    "last_login": "2018-05-25T08:55:00.222Z",
    "handle": "@spartacus2018",
    "recent_posts": [ {
       "uri": "https://soci.al/post/5442a3",
       "text": "Still nothing better than a cronut",
       "posted": "2018-05-29T14:04:19.811Z" }
    ]
  }
}
```

```
{
  "status": "partial",
  "details": {
    "id": 123,
    "username": "spartacus",
    "last_login": "2018-05-25T08:55:00.222Z",
    "handle": "@spartacus2018",
    "recent_posts": []
  }
}
```

Scenario: User Details Controller

```
class UserDetailsController extends Controller
{
    ...
    public function show($id): array
    {
        $user = \App\User::findOrFail($id);
        $user_attribs = $user->only(["id", "username", "last_login", "handle"]);
        $result = [];
        try {
            $posts = $this->getRecentPosts($user);
            $result["status"] = "complete";
            $result["details"] = $user_attribs + ["recent_posts" => $posts];
        } catch (SocialSiteClientException $e) {
            $result["status"] = "partial";
            $result["details"] = $user_attribs + ["recent_posts" => []];
        }
        return $result;
    }
    ...
}
```

Scenario: User Details Controller

```
class UserDetailsController extends Controller
{
    ...
    private function getRecentPosts(\App\User $user): array
    {
        if ($user->handle === null) {
            return [];
        }
        try {
            return $this->social_site_client->recentPostsForUser($user->handle);
        } catch (\Exception $e) {
            throw new SocialSiteClientException("Error retrieving posts", 0, $e);
        }
    }
}
```

Demo: User Details Controller

Scenario: User Details Controller

```
class SocialSiteClient {
   public function recentPostsForUser(string $handle, int $max = 5): array
   {
       $url = "http://soci.al/api/users/".urlencode($handle)."?include=posts&limit={$max}";
       $opts = [
           CURLOPT_RETURNTRANSFER => true,
           CURLOPT_HTTPHEADER => ["Content-Type: application/json"],
           CURLOPT_URL => $url,
           CURLOPT_CONNECT_TO => ["soci.al:80:proxy:23080"],
           CURLOPT_TIMEOUT => 10
       ];
       $curl = curl_init();
       curl_setopt_array($curl, $opts);
       $response = curl_exec($curl);
       ...
```

Demo: User Details Controller

Modern Apps Are Servers and Clients

Modern Apps Are Servers and Clients

- Document and diagram your dependencies as part of your production runbook
- System diagrams communicate a general mental model
- Dependency diagrams communicate a specific risk model

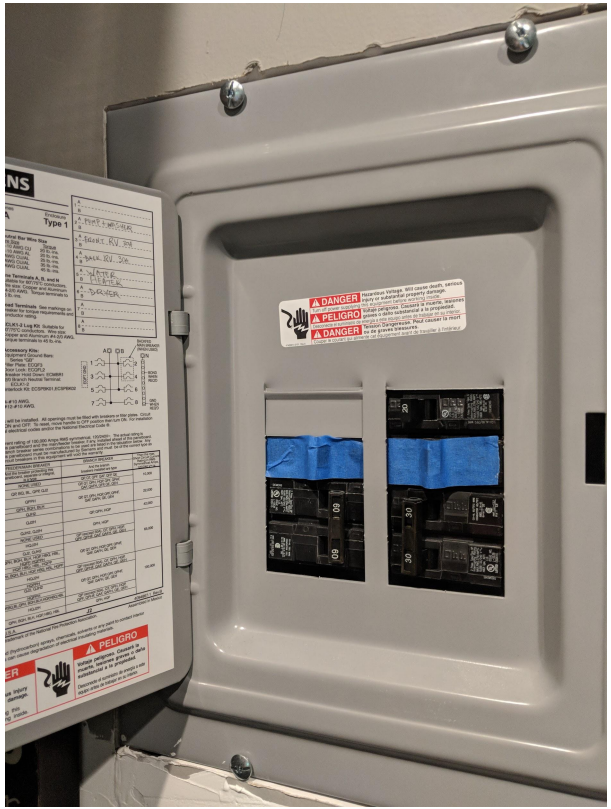
Modern Apps Are Servers and Clients

Resilience in server-side code comes from simplicity

Resilience in client code comes from sophistication

Release It! by Michael T. Nygard

(Physical) Circuit Breaker



(Software) Circuit Breaker Pattern

- A feature switch: code path differs whether breaker is enabled or disabled
- Includes failure detection algorithm

```
if <breaker is enabled> then
        … //normal behavior
else
        … //degraded behavior
on error
        <trip breaker>
```

Scenario: User Details Controller

```php
class UserDetailsController extends Controller {
    ...
    private function getRecentPosts(\App\User $user): array
    {
        if ($user->handle === null) {
            return [];
        }
        if ($this->social_circuit_breaker->isEnabled()) {
            try {
                return $this->social_site_client->recentPostsForUser($user->handle);
            } catch (\Exception $e) {
                error_log("Error retrieving posts: {$e->getMessage()}");
                throw new SocialSiteClientException("Error retrieving posts", 0, $e);
            }
        } else {
            throw new SocialSiteClientException("Posts service unavailable");
        }
    }
}
```

Circuit Breaker Implementation

```
class CircuitBreaker {
    private $name;
    private $redis;

    public function __construct(string $name) {
        $this->name = $name;
        $this->redis = Cache::store("redis");
    }

    public function isEnabled(): bool {
        $state_key = "circuit_breaker.{$this->name}.state";
        return $this->redis->get($state_key, "enabled") !== "disabled";
    }
    public function enable(): void { $this->setState("enabled"); }
    public function disable(): void { $this->setState("disabled"); }
    private function setState($state): void {
        $state_key = "circuit_breaker.{$this->name}.state";
        $this->redis->forever($state_key, $state);
    }
}
```

Circuit Breaker: Detecting Failure

```
class UserDetailsController extends Controller {
   ...
   private function getRecentPosts(\App\User $user): array
   {
      if ($user->handle === null) {
          return [];
      }
      if ($this->social_circuit_breaker->isEnabled()) {
         try {
             return $this->social_site_client->recentPostsForUser($user->handle);
         } catch (\Exception $e) {
             $this->social_circuit_breaker->disable();
             throw new SocialSiteClientException("Error retrieving posts", 0, $e);
         }
      } else {
         throw new SocialSiteClientException("Posts service unavailable");
      }
   }
}
```

Demo: Circuit Breaker

Circuit Breaker: Detecting Failure

- How sensitive should we be to failure?
  - An error is thrown
  - A call times out
  - 3 consecutive failures
  - At least 3 failures in the last minute
  - More than 25% failures over the last minute
  - Something else?

Scenario: User Details Controller

- Call to external web API provides list of most recent social media posts
  - We will consider the external web API unavailable if we get 3 failures within the last minute.
  - If the social media posts are not available, we should degrade our response by omitting that detail.

Circuit Breaker: Detecting Failure

```
class UserDetailsController extends Controller {
   ...
   private function getRecentPosts(\App\User $user): array
   {
      if ($user->handle === null) {
          return [];
      }
      if ($this->social_circuit_breaker->isEnabled()) {
         try {
             $posts = $this->social_site_client->recentPostsForUser($user->handle);
             $this->social_circuit_breaker->recordSuccess();
             return $posts;
         } catch (\Exception $e) {
             error_log("Error retrieving posts: {$e->getMessage()}");
             $this->social_circuit_breaker->recordFailure($e);
             throw new SocialSiteClientException("Error retrieving posts", 0, $e);
         }
      } ...
   }
```

Circuit Breaker: Detecting Failure

```php
interface FaultDetector
{
    public function recordSuccess(): void;
    public function recordFailure(\Exception $e = null): bool;
}


class AlwaysTripFaultDetector implements FaultDetector
{
    public function recordSuccess(): void {
        //NOOP
    }

    public function recordFailure(\Exception $e = null): bool {
        return true;
    }
}
```

Circuit Breaker: Detecting Failure

```
class CircuitBreaker {
    ...
    private $fault_detector;

    public function __construct(string $name) {
        ...
        $this->fault_detector = new AlwaysTripFaultDetector();
    }
    ...
    public function recordSuccess(): void {
        $this->fault_detector->recordSuccess();
    }

    public function recordFailure(\Exception $e = null): void {
        $should_trip = $this->fault_detector->recordFailure($e);
        if ($should_trip) {
            $this->disable();
        }
    }
}
```

Circuit Breaker: Detecting Failure

```
class FailureWindowFaultDetector implements FaultDetector
{
    private $name;
    private $max_failures;
    private $sampling_window;
    private $redis;

    public function __construct(string $name, int $max_failures, CarbonInterval $window)
    {
        $this->name = $name;
        $this->max_failures = $max_failures;
        $this->sampling_window = $window;
        …
    }
    …
```

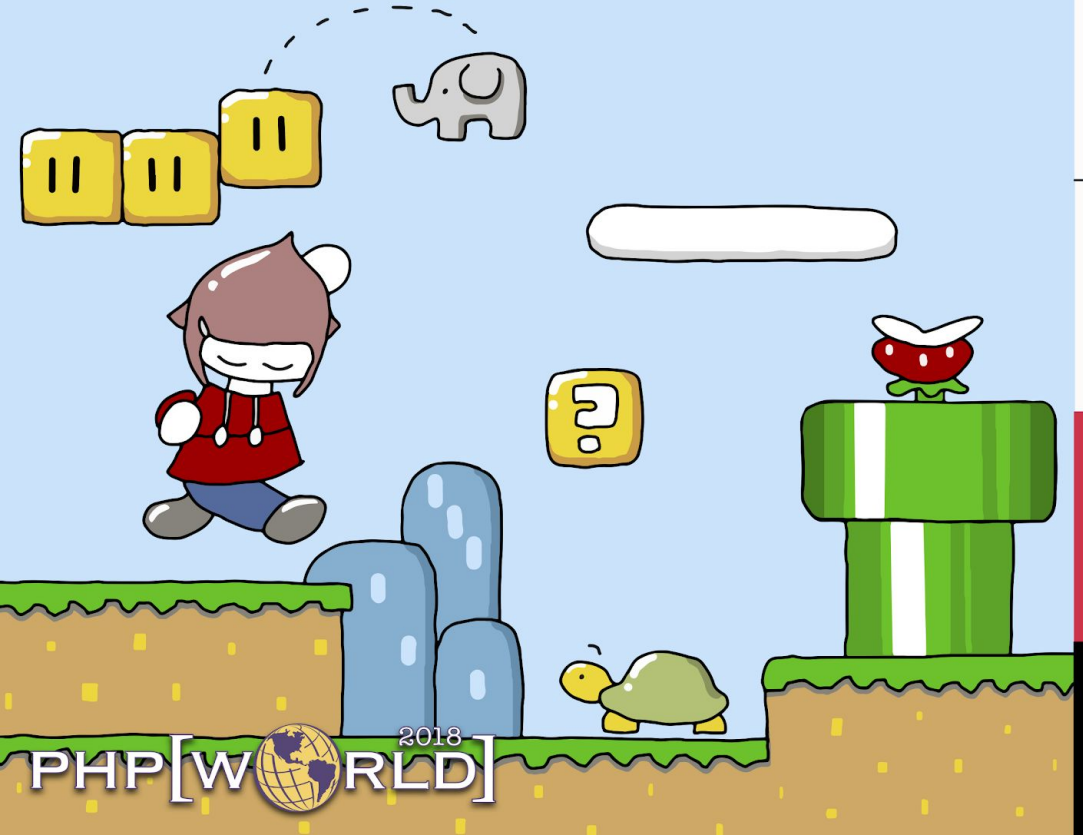
Circuit Breaker: Detecting Failure

```
  ...
  /**
   * For Redis geeks only: use a sorted set of failure timestamps for windowing
   */
  public function recordFailure(\Exception $e = null): bool {
     $key = "failure_window.{$this->name}";
     $now = time();
     $window_duration = $this->sampling_window->seconds;
     $beginning = $now - $window_duration;
     $redis_ops = function(Pipeline $pipe) use ($key, $now, $window_duration, $beginning) {
        $pipe->zremrangebyscore($key, 0, $beginning_of_window); //delete the old
        $pipe->zadd($key, [(string) $now => $now]);  //add this failure
        $pipe->expire($key, $window_duration_seconds); //update the TTL
        $pipe->zcard($key);  //see how many failures we have now
     };
     $batch_result = $this->redis->pipeline(["atomic" => true], $redis_ops);
     $failure_count = $batch_result[count($batch_result)-1];
     return $failure_count > $this->max_failures;
   }
}
```

Circuit Breaker Pattern: Next Steps

- Evolve your fault detectors
- Self-resetting == Self-healing system
- When a breaker trips, great time to trigger advanced diagnostics

Thanks to Our Sponsors



AUTOMATTIC

Square

concrete5

Magento

twilio

OSMI

SOLUTIONSTREET
Quality Software Consulting