

Reactor Pattern & Event-Driven Programming

A scalable concurrent approach,
using `EventMachine` with `Thin` as an example

Reactor Pattern & Event-Driven Programming

<http://godfat.org/slide/2010-02-29-reactor-pattern-and.pdf>

Table of Content

Table of Content

- concurrency, why and how in network

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works
- how EventMachine works

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works
- how EventMachine works

concurrency, why and how in network

concurrency, why and how in network

- [network I/O] is slow, we shouldn't wait for [network I/O] while make [CPU] idle.

concurrency, why and how in network

- [network I/O] is slow, we shouldn't wait for [network I/O] while make [CPU] idle.
- you can replace [network I/O] and [CPU] with all other resources like [disc I/O], [memory I/O], etc.

concurrency, why and how in network

- [network I/O] is slow, we shouldn't wait for [network I/O] while make [CPU] idle.
- you can replace [network I/O] and [CPU] with all other resources like [disc I/O], [memory I/O], etc.
- each kernel process/thread for each client using a blocking I/O is easy to write but not scalable at all

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works
- how EventMachine works

Event-Driven Programming

to the rescue

Event-Driven Programming

- only one process/thread

Event-Driven Programming

- only one process/thread
- inversion of control

Event-Driven Programming

- only one process/thread
- inversion of control
- consists of an event loop and various event handlers

Event-Driven Programming

- inversion of control

Event-Driven Programming

```
loop{  
  # you control the flow  
  do_something  
}
```

- inversion of control

Event-Driven Programming

```
                                register method(:do_something)
loop{                             loop{
  # you control the flow          # event loop control the flow,
  do_something                   # later it calls your callback
}                                event = pop_event_queue
                                dispatch event if event
                                }
```

- inversion of control

Event-Driven Programming

```
register method(:do_something)
loop{
  # event loop control the flow,
  # later it calls your callback
  event = pop_event_queue
  dispatch event if event
}
```

- consists of an event loop and various event handlers

Event-Driven Programming

```
register method(:do_something)
```

```
loop{
```

```
# event loop control the flow,
```

```
# later it calls your callback
```

```
event = pop_event_queue
```

```
dispatch event if event
```

```
}
```

- consists of an event loop and various event handlers

Event-Driven Programming

```
register method(:do_something)
```

```
loop{
```

```
# event loop control the flow,
```

```
# later it calls your callback
```

```
event = pop_event_queue
```

```
dispatch event if event
```

```
}
```

- consists of an event loop and various event handlers

Event-Driven Programming in Flash with Ruby syntax

Event-Driven Programming in Flash with Ruby syntax

- game loop, an example of event loop

Event-Driven Programming in Flash with Ruby syntax

- game loop, an example of event loop
- Flash ActionScript, onEnterFrame

Event-Driven Programming in Flash with Ruby syntax

- game loop, an example of event loop
- Flash ActionScript, onEnterFrame
- frame by frame

Event-Driven Programming in Flash with Ruby syntax

```
# in each sprite thread
30.times{
  application.draw sprite
  sprite.x += 1
}
```

```
sprite.onEnterFrame = lambda{  
    sprite.x += 1  
}
```

```
# in each sprite thread  
30.times{  
    application.draw sprite  
    sprite.x += 1  
}
```

```
sprite.onEnterFrame = lambda{
  sprite.x += 1
}
application.register sprite
30.times{ # event loop, also called game loop
  events = application.pop_event_queue
  events.each{ |event|
    application.dispatch event
  }
  # model/view separation
  application.draw application.sprites
}
```

```
# in each sprite thread
30.times{
  application.draw sprite
  sprite.x += 1
}
```

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works
- how EventMachine works

Reactor Pattern

Reactor Pattern

```
loop{  
    data = read  
    handle data  
}
```

Reactor Pattern

```
loop{
    data = read
    handle data
}

register method(:handle)
loop{
    data = partial_read
    event = process data
    dispatch event if event
}
```

Event-Driven Programming

```
loop{
    # you control the flow
    do_something
}

register method(:do_something)
loop{
    # event loop control the flow,
    # later it calls your callback
    event = pop_event_queue
    dispatch event if event
}
```

Reactor Pattern

```
loop{
    data = read
    handle data
}

register method(:handle)
loop{
    data = partial_read
    event = process data
    dispatch event if event
}
```

Reactor Pattern

by wikipedia

Reactor Pattern

- resources # e.g. network I/O

by wikipedia

Reactor Pattern

- resources # e.g. network I/O
- synchronous event demultiplexer
i.e. the blocking event loop

by wikipedia

Reactor Pattern

- resources # e.g. network I/O
- synchronous event demultiplexer
i.e. the blocking event loop
- dispatcher
i.e. handler manager and event dispatcher

by wikipedia

Reactor Pattern

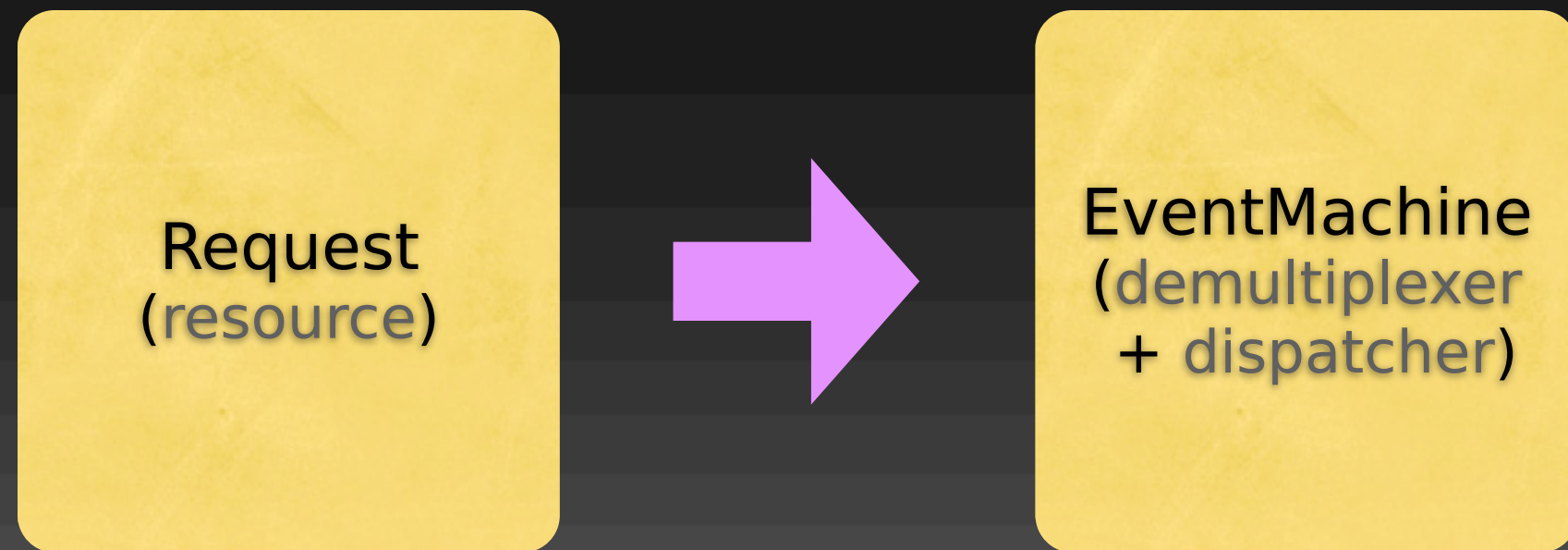
- resources # e.g. network I/O
- synchronous event demultiplexer
i.e. the blocking event loop
- dispatcher
i.e. handler manager and event dispatcher
- request handler # e.g. thin handler
by wikipedia

Reactor Pattern

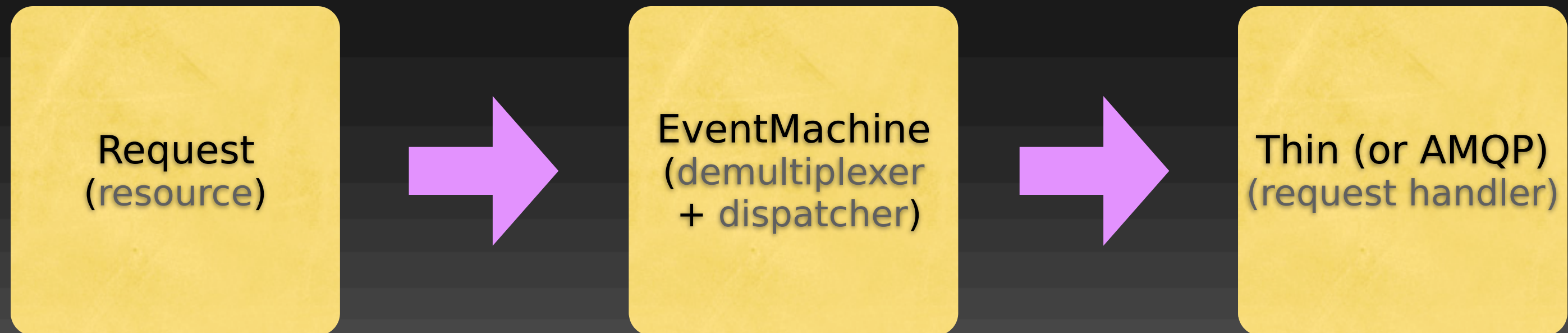


Request
(resource)

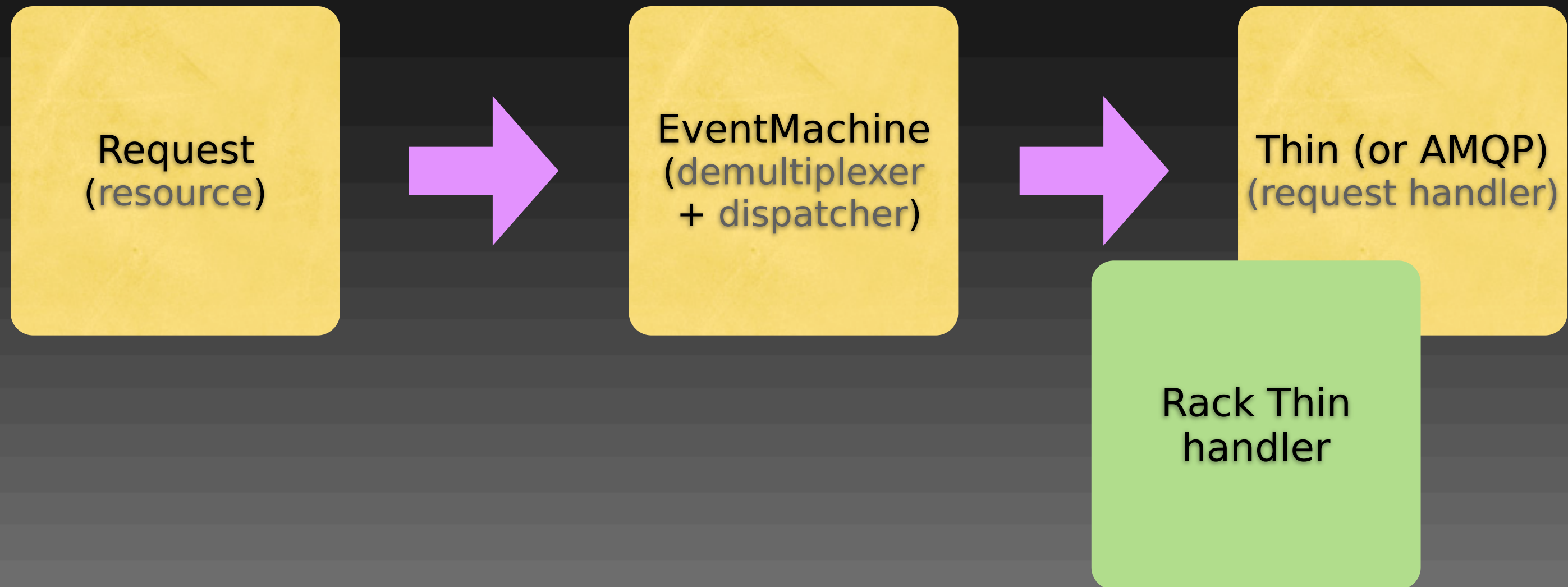
Reactor Pattern



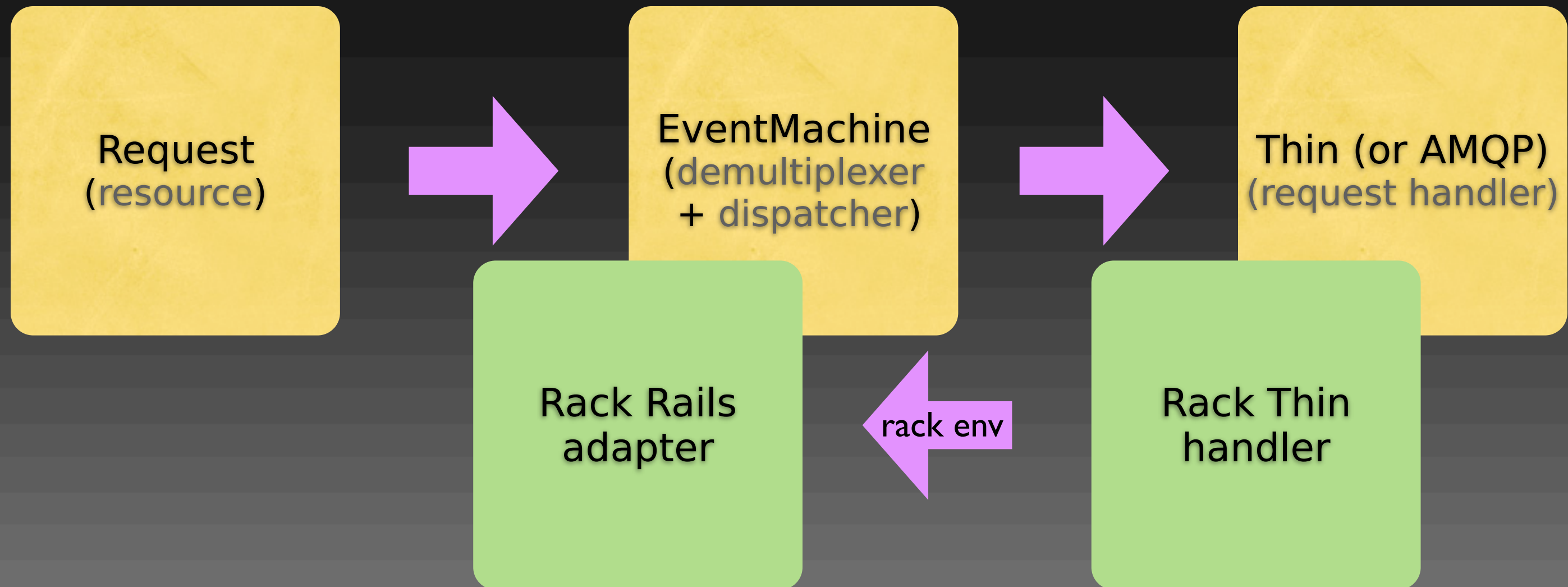
Reactor Pattern



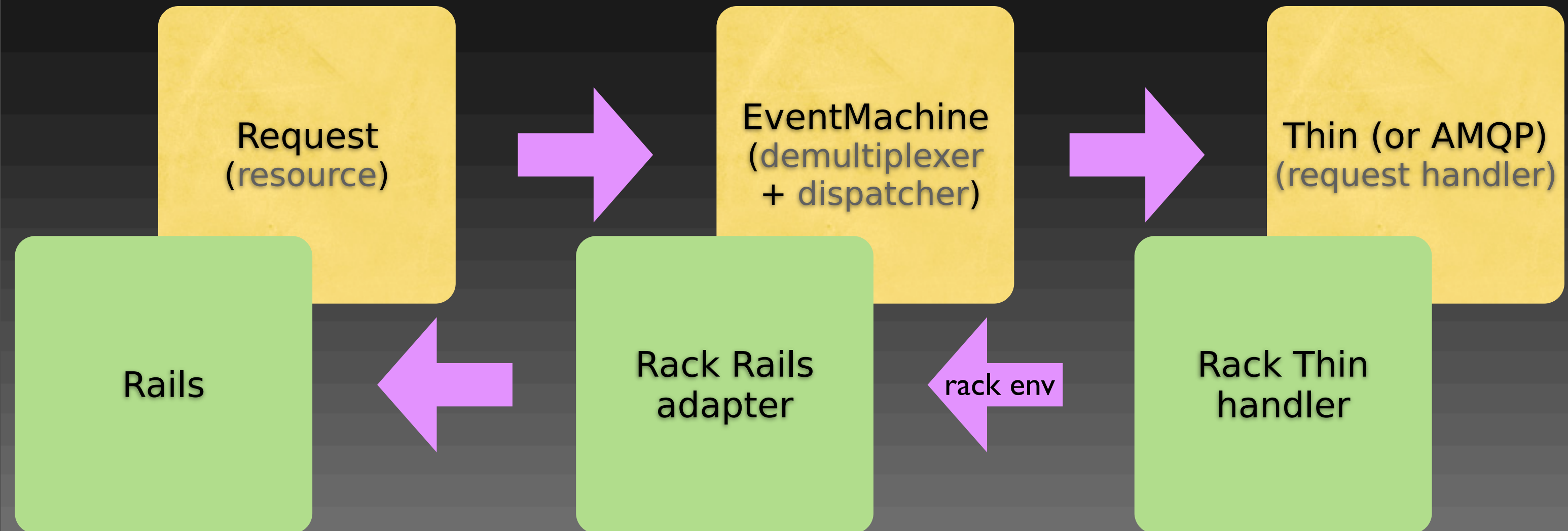
Reactor Pattern



Reactor Pattern



Reactor Pattern



Reactor Pattern

your rails
application

Request
(resource)

EventMachine
(demultiplexer
+ dispatcher)

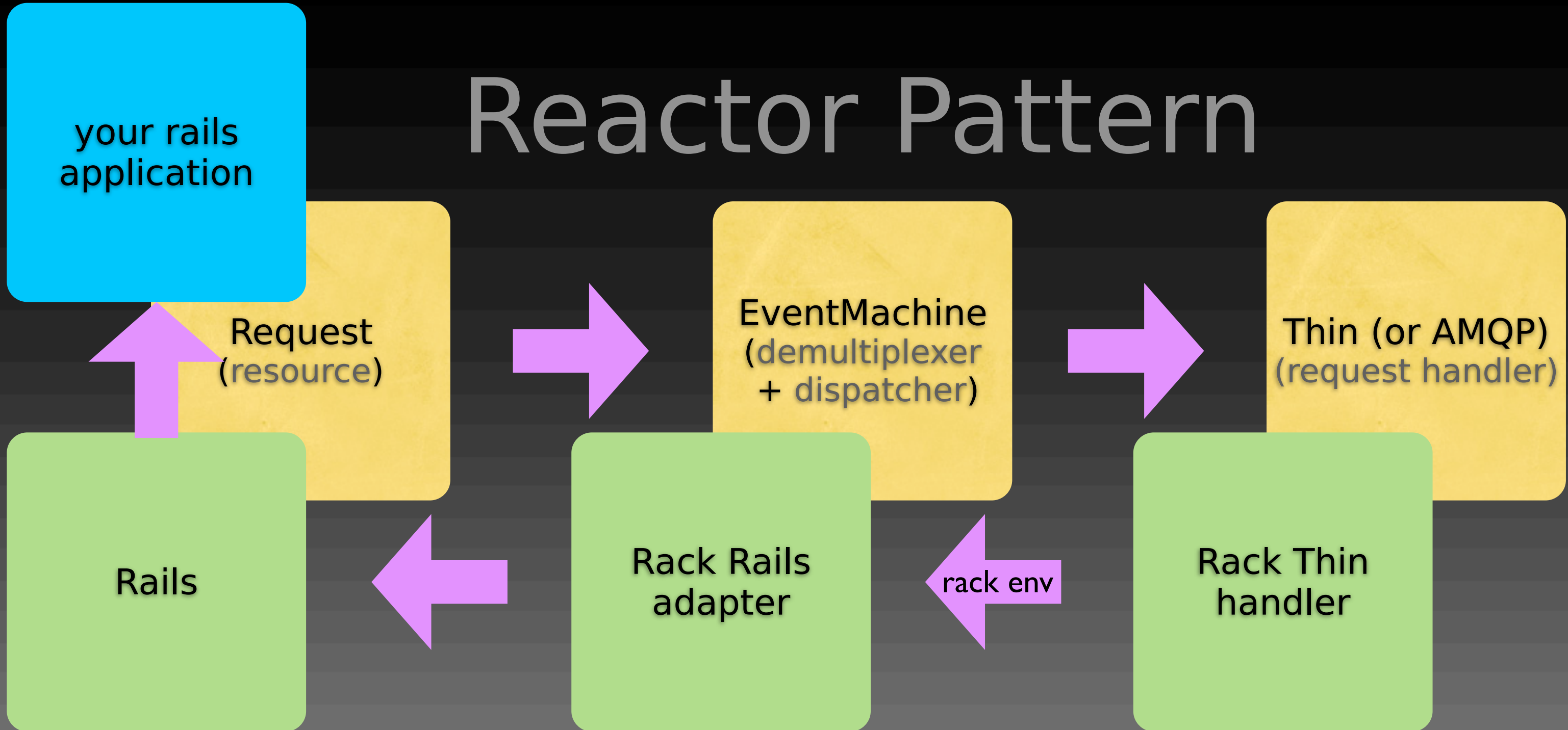
Thin (or AMQP)
(request handler)

Rails

Rack Rails
adapter

rack env

Rack Thin
handler



Reactor Pattern

EventMachine is a generic network I/O server/client library due to I/O and request handler separation in
Reactor Pattern

Reactor Pattern

- EventMachine (Ruby)

Reactor Pattern

- EventMachine (Ruby)
- Twisted (Python)

Reactor Pattern

- EventMachine (Ruby)
- Twisted (Python)
- nodejs (JavaScript in V8)

Reactor Pattern

- EventMachine (Ruby)
- Twisted (Python)
- nodejs (JavaScript in V8)
- libevent and libev (C)

Reactor Pattern

- select (POSIX)

Reactor Pattern

- select (POSIX)
- poll (POSIX)

Reactor Pattern

- select (POSIX)
- poll (POSIX)
- epoll (Linux)

Reactor Pattern

- select (POSIX)
- poll (POSIX)
- epoll (Linux)
- kqueue (BSD, Mac OS X (Darwin))

Table of Content

- concurrency, why and how in network
- Event-Driven Programming explained in Flash with Ruby syntax
- Reactor Pattern in EventMachine with Thin
- how Thin works
- how EventMachine works

how Thin work

- Thin::Server

how Thin work

- `Thin::Server`
- `Thin::Backends::TcpServer`
communicate with EventMachine

how Thin work

- `Thin::Server`
- `Thin::Backends::TcpServer`
communicate with EventMachine
- `Thin::Connection`
EventMachine event handler

how Thin work

- `Thin::Server`
- `Thin::Backends::TcpServer`
communicate with EventMachine
- `Thin::Connection`
EventMachine event handler
- `Thin::Request`
partial HTTP request parsing
Rack env builder

how Thin work

Sorry! To be continued.....

how Thin work

Sorry! To be continued.....

