Hexagonal Architecture

(hek-sag-uh-nl)

Chris Fidao
Implementing Laravel

Real-world implementation of testable and maintainable code. (hopefully)
Vaprobash
Vagrant Provisioning Bash Scripts
Servers for Hackers.com
Why / What
Ports / Adapters
Boundary/Layers
WHY Architecture?
What is it?
So... What is it?
The Hexagon

Core Domain

Domain

Application

Framework

[Diagram showing the hexagon and its layers: Core Domain, Domain, Application, Framework, with arrows indicating interactions with external elements like a globe, envelope, SQL database, and cloud.]
(Core) Domain
Behavior
Constraints
Framework
Ports / Adapters
Ports & Adapters

Core

Domain

Application

Framework

Ports & Adapters diagram showing layers and interactions.
Inside/Outside

- Framework
- Database
- DBAL
- Application
- Use Case
- Core Domain
- Events
- dispatcher
- Service Impl
- CommandBus
- HTTP
interface Notifier {

    public function send(Message $message);
}

class SesNotifier implements Notifier {

    public function send(Message $message) {
        // Details
    }
}
Use-Case Driven Development
All the Contexts

- Web
- API
- CLI
- Queue
- Event Handler
Use Cases: CommandBus

CommandBus ➔ executes( Command )

Handler ➔ handles( Command )
// Class SimpleCommandBus

public function execute( $command )
{
    return $this->getHandler( $command )->handle( $command );
}
Boundaries
Domain/Application Boundary

- Use Case
- Events
- Repo
- Core Domain
- Application
- Framework
interface CommandBusInterface {
    public function execute( $command );
}

interface HandlerInterface {
    public function handle( $command );
}
interface TicketRepositoryInterface {

    public function getStaffOpenTickets(
        Staffer $staffer, $limit=10);

    public function save(Ticket $model);
}
The Application/External Boundary
interface Notifier {
    public function send(Message $message);
}

interface Validator {
    public function passes(Array $data);
    public function getErrors();
}

interface Dispatcher {
    public function dispatch(Array $events);
}
Framework

Core

Domain

Application

Framework

HTTP

Service Impl

Database

SQL
Identify the aspects that vary and separate them from what stays the same.
Layers
The Domain
class Ticket extends Model {

    public function assignStaffer(Staffer $staffer)
    {
        if( ! $staffer->categories->contains( $this->category ) )
        {
            throw new DomainException("Staffer can't be assigned to ". $this->category);
        }

        $this->staffer()->associate($staffer); // Set Relationship

        return $this;
    }

    public function setCategory(Category $category)
    {
        if( $this->staffer instanceof Staffer && ! $this->staffer->categories->contains( $category ) )
        {
            // Unset staffer if can't be assigned to set category
            $this->staffer = null;
        }

        $this->category()->associate($category); // Set Relationship

        return $this;
    }
}
class Ticket extends Model {

    /* ... Other logic ... */

    public function save(array $options = array()) {
        /* Integrity Checks, and then: */

        if (! $this->exists ) {
            $this->raise( new TicketCreatedEvent($this) );
        }

        return parent::save($options);
    }
}
class CreateTicketCommand {

    protected $data;

    public function __construct($data)
    {
        $this->data = $data;
    }

    public function __get($property)
    {
        // Simplified example
        return $this->data[$property];
    }
}
The Application

- CommandBus
- Dispatcher
- Domain
- Application
- Framework
- DBAL

- Framework
- Dispatcher
- Domain
- Application
- Framework
- DBAL
public function execute( $command )
{
    return $this->getHandler( $command )->handle( $command );
}
class CreateTicketHandler implements HandlerInterface {

    public function handle($command) {
        $this->validate($command); // Throw ValidationException
        $this->save($command);
    }

    protected function save($command) {
        $ticket = new Ticket;
        /* Some other setters... */
        $ticket->setCategory( $this->catRepo->find($command->category_id) );
        $ticket->setStaffer( $this->staffRepo->find($command->staffer_id) );
        $ticket->addMessage( $ticket->addMessage($command->message) );

        $this->ticketRepo->save($ticket); // Use Repositories

        $this->dispatcher->dispatch( $ticket->flushEvents() ); // Fire Events
    }
}
class DbTicketRepository implements RepositoryInterface {

    public function getStaffOpenTickets(Staffer $staffer, $limit=10) {
        return $this->ticket->where('staff_id', $staffer->id)
            ->take($limit)->get();
    }

    public function save(Ticket $ticket) {
        $ticket->save();
    }
}
class TicketController extends BaseController {

public function createTicket()
{
    $command = new CreateTicketCommand(Input::all());

    try {
        $this->bus->execute($command);
    } catch(ValidationException $e) {
        return Redirect::to('/tickets/new')->withErrors($e->getErrors());
    } catch(DomainException $e) {
        return Redirect::to('/tickets/new')->withErrors($e->getErrors());
    }

    return Redirect::to('/tickets');
}
}
class SesEmailNotifier implements NotifierInterface {

    public function __construct(SesClient $client)
    {
        $this->client = $client;
    }

    public function send(Message $message)
    {
        $to = [$message->to()];
        $message = ['Data' => $message->message()];

        $this->client->sendEmail(['
            'Destination' => ['ToAddresses' => $to],
            'Message' => ['Body' => ['Html' => $message]]
        ]);    
    }
}
use Illuminate\Events\Dispatcher as EventDispatcher;

class LaravelDispatcher implements Dispatcher {

    public function __construct(EventDispatcher $dispatcher)
    {
        $this->dispatcher = $dispatcher;
    }

    public function dispatch(Array $events)
    {
        foreach ( $events as $event )
        {
            $this->dispatcher->fire( $event->name(), $event );
        }
    }
}
TDD is DEAD
(and other myths)
Identify the aspects that vary and separate them from what stays the same.
Thanks