

Onion Architecture + Symfony Cheat Sheet by vikbert (vikbert) via cheatography.com/20854/cs/23882/

Onion Architecture

Layer: Application Core (application +

domain)

Layer: Presentation

Layer: Infrastructure

Layer: Tests

1. Application Core

Domain has no interaction direct with outerlayer. It represents the domain business and domain logic. It defines always the domain specific entities, value objects, events, exceptions, services, factories, interfaces.

Application Application layer manages the internal domain logic. It provides different application services, which enable the communication with presentation, tests and infrastructure.

1.1 Domain Layer

models	consist of entities, value
	objects, aggregates
repository	interfaces to access the
interface	business models, which are
	used by application and
	implemented by outer layer.
	For example: infrastructure

assertions

the business rules to adjust changes on business behavior and business models

services

domain services define the complex internal communication among the domain models. For example: apply some changes cross different domain models.

events

which can be used to track the state changes of domain

1.2 Application Layer

events / Event Subcriber	defines the events, which represent the state changes in business domain. for example:
services	theser services enable the interaction with internal domains by using the predefined interfaces in the domain layer.
query interface	These interfaces are defined for fetching the domain data. They are commonly used by presentation layer and implemented by infrastructure layer.
command	they are simple objects, which are used to change the state of business domain. For

example: confirmPayment

2. Presentation Layer

contro-	controllers are the typical
llers	gateways for interaction
	comming from end user. It can
	be a controller, that represents
	REST endpoint; or a controller,
	that renders the web page.
consoles	It enables the user to access
	and update the application
	core via console in terminal.
templates	provide the template to define
	how to represent the business
	data. for example: template of
	email, template of exports,
	template of preview
views/-	provide the UX interface to end

2. Presentation Layer (cont)

DTO Data Transfer Object, defines the view model of request and response

Presentation layer provides the interfaces how end user can drive the business logic

3. Infrastructure layer

doctrine	query implementations
mail	repository implementations
filesystem	exports
Queue	cron-jobs
SSO	logging

The infrastructure layer holds the most low level code. Anything in here should be easy to replace. Code here should never effect anything related to logic, or how your application behaves.

4. Tests Layer

unit tests	works well
integr- ation	test if the communication between application core and external services in infrastru- cture layer is possible
functional	test if the interaction between end user and the presentation layer work well

unit toota toot if internal application core

Tests layer test the functionality of application core and integation between application core and outer layer.

Remark 01:

Application core is the independent core, which defines the most of core logic and a couple of interfaces, that must be implemented and used by outer layer. The inner application core should be independent from outlayer, and should be always runable, if you change any part of the outer layer.



By **vikbert** (vikbert) cheatography.com/vikbert/

Published 31st July, 2020. Last updated 13th June, 2021. Page 1 of 2.

users

forms

Sponsored by Readable.com

Measure your website readability!

https://readable.com



Onion Architecture + Symfony Cheat Sheet by vikbert (vikbert) via cheatography.com/20854/cs/23882/

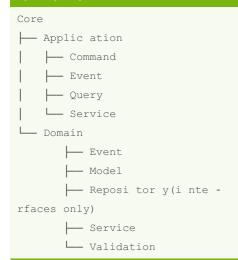
Key tenets of Onion

The big advantag of Onon Architecture is that business logic ends up coupled to ONLY application layer concerns, not to infrastructure layer anymore. The application is built around an independent object model. Inner layers define interfaces. Outer layers implement interfaces

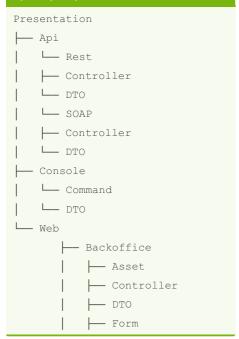
Direction of coupling is toward the center.

All application core code can be compiled and run separate from infrastructure

Symfony Project structure: Core



Symfony Project structure: Presentation



Symfony Project structure: Presentation (cont)



Symfony Project structure: Infrastructure

Infrastructure
— Mail
Persis tence
│
Migrations
☐ Repository
— Queue
L SSO

Symfony Project structure: Tests

Test	s
<u> </u>	functional
<u> </u>	integr ation
	unit

Remark 02

Application layer should never use the concret implementation from infrastructure layer or presentation layer. It defines the application interfaces and manages the domainer interfaces, so that the application core can work a wohle without outerlayer. By providing the different application services, the communication with tests, presentation and infrastructure is possible.



By **vikbert** (vikbert) cheatography.com/vikbert/

Published 31st July, 2020. Last updated 13th June, 2021. Page 2 of 2. Sponsored by **Readable.com**Measure your website readability!
https://readable.com