General concepts

Goal

- To get familiar with Oro applications
- How they are organized
- Components that are used

Foundation

- Symfony Framework
  - No significant issues with adding symfony-specific components to ORO applications
- Doctrine ORM
  - Pay attention on the performance
- Some best practices aren't used
  - Eg. bundles.yml / routing.yml

https://symfony.com/projects/orocrm
https://symfony.com/projects/orocommerce

Oro applications

- OroCRM (Community and Enterprise)
- OroCommerce (Community and Enterprise)
- Akeneo PIM (external)
- Diamante Desk (external)
- Marello (external)
- All of them can* be used together, eg. crm+commerce
- Features and differences between CE and EE
  - Data volume, scalability, performance, support and more

Application environments

- dev
  - Symfony toolbar + ORO extensions
  - The slowest one
  - Good for debugging, mail catching etc.
  - System configuration
- prod
  - Performance optimized
  - Recommended for demo
- test
  - No demo data!

Demonstration >>>

Application environments

- Best practices from ORO team
  - Dev env for development
  - Use prod for demo for the clients
  - Check features in prod env before merging
  - Sometimes is worth to do clean install
  - In prod JS/assets are minified

Application directory structure

- Almost hard copy of Symfony
  - Show ORO customizations
- Customizations in /src or via composer
- /public
  - tracking.php

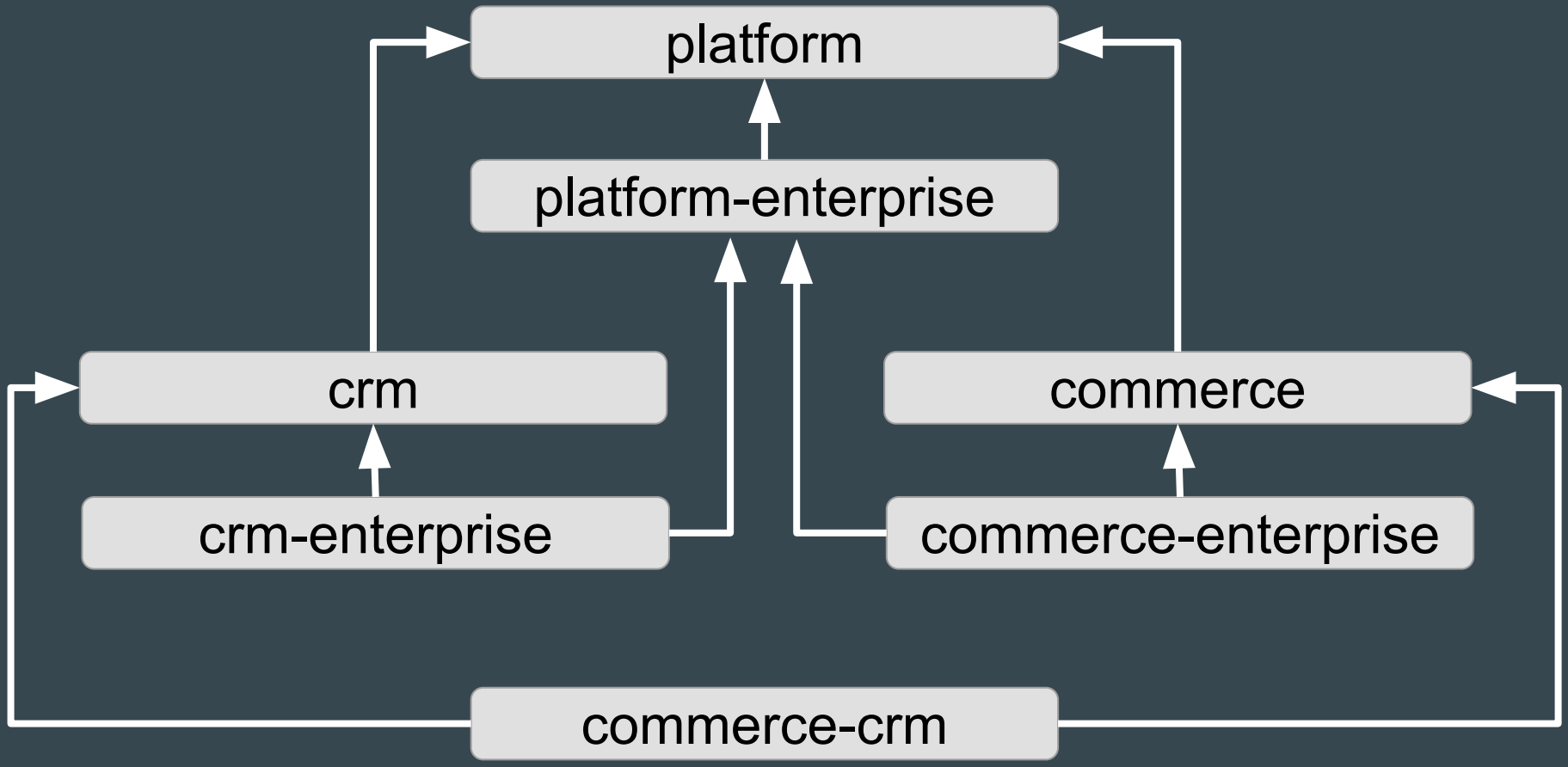
Demonstration >>>

Structure

Applications and packages

- Package contains finished functionality
- Application requires one or more packages
- Packages may require other packages
- Composer package manager

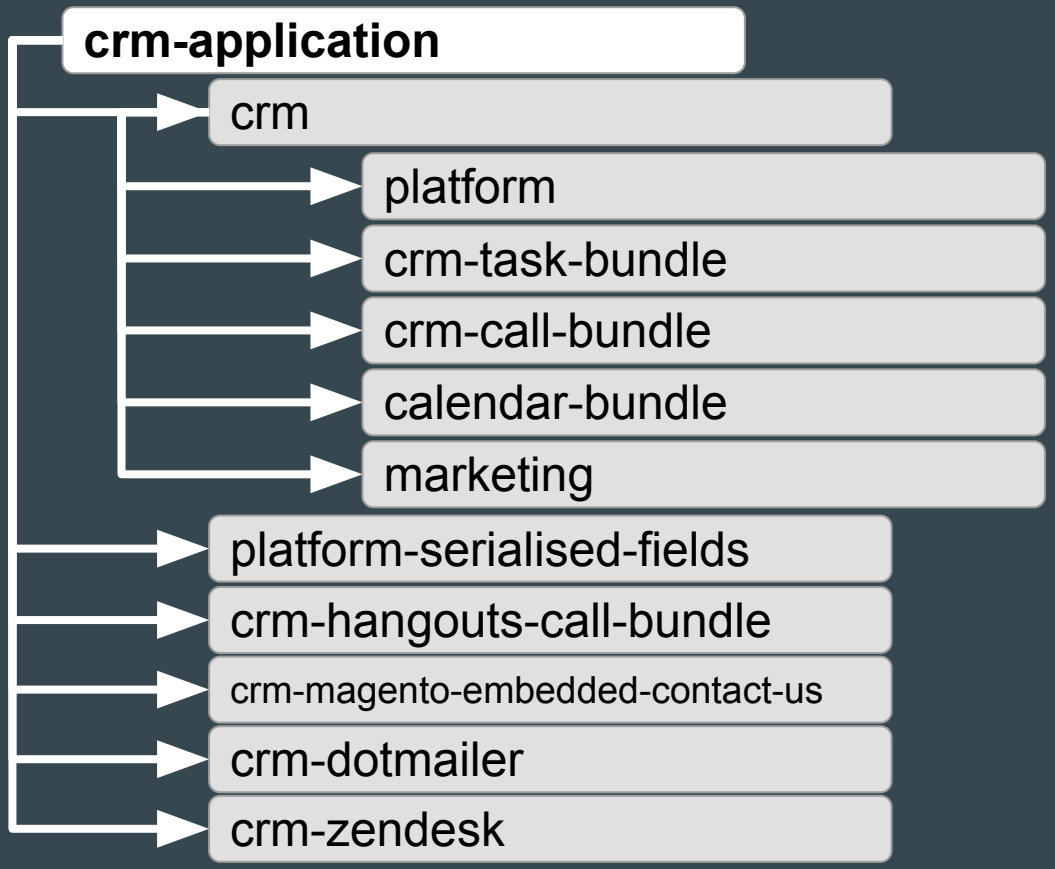
Main package dependencies



Additional packages

- Plugins
- Integrations
- Additional functionality
- Marketplace
  - https://marketplace.orocommerce.com/
  - How to Manage Extensions

Packages in OroCRM application



Dependencies of OroCRM application

- https://github.com/oroinc/crm-application/blob/3.1/composer.json#L22-L27
- https://github.com/oroinc/crm/blob/3.1/composer.json#L17-L21
- https://github.com/oroinc/platform/blob/3.1/composer.json#L13-L112

Packages and bundles

- Bundle contains finished code
- Package contains one or more bundles
  - eg. marketing
- No package manager for bundles

Demonstration >>>

Bundle directory structure

- Resources/config/oro
- Migrations
- .md files
- and more

Demonstration code >>>

Components and bridges

- Component
  - Abstract functionality (library)
  - No bundle dependencies
- Bridge
  - Connects several bundles

Demonstration >>>

Relational DBMSes

- MySQL
  - Not strict SQL
  - Suitable for small applications - up to 1M of entities
- PostgreSQL
  - Strict SQL
  - Suitable for big applications - more than 1M of entities
  - Supported only in Enterprise Edition

Search engines

- ORM
  - Suitable for small applications
  - Stores data in the application DB
  - Implements EAV
- Elasticsearch
  - Suitable for big applications
  - Allows to configure search behaviour
  - Document based engine (NoSQL)
  - Efficient aggregation, geo queries, REST, and more
  - Supported only in Enterprise Edition

[Admin guide]

Message queue engines

- DBAL
  - Suitable for small applications
  - Stores data in the application DB
  - Emulates queue
  - Better for debugging
- AMQP (RabbitMQ)
  - Suitable for big applications
  - RabbitMQ allows better scaling
  - Supported only in Enterprise Edition

[Docs] & [Admin guide]

Application configuration: Files

- Symfony
- config/config.yml ; parameters.yml
- config/security.yml
- config/routing.yml

- ORO
- <bundle>/Resources/config/oro/app.yml
- <bundle>/Resources/config/oro/routing.yml
- and others...

https://github.com/oroinc/platform/tree/master/src/Oro/Bundle/PlatformBundle
https://github.com/oroinc/platform/tree/master/src/Oro/Bundle/DistributionBundle

Application configuration: UI

- Global: System > Configuration
- Scoped:
  - User
  - Website (EE only)
  - Organization (EE only)
- Docs

Demonstration >>>

SOLID

- Single responsibility principle
- Open/closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

GRASP

General Responsibility Assignment Software Patterns

- Controller ⇒ entry point
- Creator ⇒ factory
- High cohesion ⇒ strict responsibility, focused
- Indirection ⇒ mediator
- Information expert ⇒ delegate responsibility
- Low coupling
- Polymorphism
- Protected variations ⇒ wrap instabilities
- Pure fabrication ⇒ service class

KISS

Keep
It
Simple,
Stupid

DRY

Don’t
Repeat
Yourself

YAGNI

You
Aren't
Gonna
Need
It

Application installation: CLI

- bin/console oro:install
- Additional parameters
- Might take up to 5 minutes
- Supports different environments
- Can be used in Continuous Integration

[Docs]

Demonstration >>>

Application upgrade

- Maintenance mode
- Turn off application processes (cron, MQ)
- Create backups of your db and source code
- Get new code
- Composer install
- Remove app/cache/<environment>
- bin/console oro:platform:update
- Return everything back

[Docs]

Cron

- bin/console oro:cron
- Triggers other commands
- Has to be executed every minute
- */1 * * * *
- It has to be prefixed with oro:cron

[Docs]

Demonstration UI + console >>>

Message queue consumer

- bin/console oro:message-queue:consume
- -vvv for more verbosity
- Supervisor
- Message limit
- Time limit
- Memory limit

Demonstration console >>>

Logger

- Use everywhere
- Use in critical places
- Use in the integrations
- Correct logging level
- Context

[Example]

How to find required code

- Search by labels
- Search by HTML (tags, classes)
- Use Symfony Profiler
- Set breakpoints
- Regex in Phpstorm

Demonstration >>>

Documentation

- Documentation in bundles
- https://www.orocommerce.com/documentation/current
- https://www.slideshare.net/OroCRM/presentations
- https://www.youtube.com/channel/UC9ougJgKzJd-ZxrLuvBqZLg
- Use Google