Chapter 4: Requirement Specification for a Modular Web Shop App
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Order Total: $283.33

Go to Checkout
### Shipping Address

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Brooks</td>
</tr>
<tr>
<td>Last Name</td>
<td>Apple</td>
</tr>
<tr>
<td>Company</td>
<td>Foggles</td>
</tr>
<tr>
<td>Phone Number</td>
<td>385 31 155 05 52</td>
</tr>
</tbody>
</table>

### Order Summary

- **Total:** $283.33

### Shipping Methods

- Express Delivery
- Standard Delivery
- Same Day Delivery
Thank you!

Your order number is: XXXXXXXXX.

We'll email you an order confirmation with details.

Continue Shopping
Symfony is a set of reusable PHP components...

The standard foundation on which the best PHP applications are built. Choose any of the 30 stand-alone components available for your own applications.

What is Symfony →

...and a PHP framework for web projects

Speed up the creation and maintenance of your PHP web applications. End repetitive coding tasks and enjoy the power of controlling your code.

Projects using Symfony

Drupal  phpBB  laravel  SiteGround  OROCRM  PIWIK

Foundation
The most advanced responsive front-end framework in the world.

Download Foundation 6
Chapter 5: Symfony at a Glance

Symfony Installer (1.5.1)

This is the official installer to start new projects based on the Symfony full-stack framework.

To create a new project called blog in the current directory using the latest stable version of Symfony, execute the following command:

```
$ symfony new blog
```

Create a project based on the Symfony Long Term Support version (LTS):
```
$ symfony new blog lts
```

Create a project based on a specific Symfony branch:
```
$ symfony new blog 2.3
```

Create a project based on a specific Symfony version:
```
$ symfony new blog 2.3.6
```

Create a demo application to learn how a Symfony application works:
```
$ symfony demo
```

Updating the Symfony Installer

New versions of the Symfony Installer are released regularly. To update your installer version, execute the following command:

```
$ symfony self-update
```

```
Brankos-MacBook-Pro:test-app branko$ symfony new test-app

Downloading Symfony...
4.98 MB/4.98 MB 100%

Preparing project...

✔ Symfony 3.4.6 was successfully installed. Now you can:

* Change your current directory to /Users/branko/www/test-app
* Configure your application in app/config/parameters.yaml file.
* Run your application:
  1. Execute the php bin/console server:run command.
  2. Browse to the http://localhost:8000 URL.
* Read the documentation at http://symfony.com/doc

Brankos-MacBook-Pro:www branko$ 
```
Welcome to the Doctrine2 entity generator.

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like Acme\BlogBundle:Post.

The Entity shortcut name: AppBundle:Customer

Determine the format to use for the mapping information.

(Configuration format: yml, xml, php, or annotation) [annotation]:

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object, boolean, integer, smallint, bigint, string, text, datetime, datetimez, date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields):

Field type [string]: string
Field length [155]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

Field type [string]:
Field length [155]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

Field type [string]:
Field length [155]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

Field type [string]:
Field length [155]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

> Generating entity class src/AppBundle/Entity/Customer.php: OK!
> Generating repository class src/AppBundle/Repository/CustomerRepository.php: OK!

Everything is OK! Now get to work :)!

Welcome to the Doctrine2 CRUD generator.

This command helps you generate CRUD controllers and templates.

First, give the name of the existing entity for which you want to generate a CRUD (use the shortcut notation like Acme\BlogBundle:Post)

(The Entity shortcut name: AppBundle:Customer)
By default, the generator creates two actions: list and show. You can also ask it to generate "write" actions: new, update, and delete.

Do you want to generate the "write" actions [no]? yes

Determine the format to use for the generated CRUD.

Configuration format (yml, xml, php, or annotation) [annotation]:

Determine the routes prefix (all the routes will be "mounted" under this prefix: /prefix/, /prefix/new, ...).

Routes prefix [/customer]:

Summary before generation

You are going to generate a CRUD controller for "AppBundle:Customer" using the "annotation" format.

Do you confirm generation [yes]?

CRUD generation

Generating the CRUD code: OK
Generating the Form code: OK
Updating the routing: OK

Everything is OK! Now get to work :).

Customer list

<table>
<thead>
<tr>
<th>Id</th>
<th>Firstname</th>
<th>Lastname</th>
<th>Email</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Create a new entry</td>
</tr>
</tbody>
</table>

Customer creation

- Firstname
- Lastname
- Email
- Create

- Back to the list
Customer list

<table>
<thead>
<tr>
<th>Id</th>
<th>Firstname</th>
<th>Lastname</th>
<th>Email</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Branko</td>
<td>Ajzele</td>
<td><a href="mailto:ajzele@gmail.com">ajzele@gmail.com</a></td>
<td>show</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>edit</td>
</tr>
<tr>
<td>2</td>
<td>Joha</td>
<td>Doe</td>
<td><a href="mailto:joh.doe@dummy.mail">joh.doe@dummy.mail</a></td>
<td>show</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>edit</td>
</tr>
</tbody>
</table>

- Create a new entry

Customer edit

- Firstname: John
- Lastname: Doe
- Email: joh.doe@dummy.mail

- Back to the list
- Delete

Oops! An Error Occurred

The server returned a "500 Internal Server Error".

Something is broken. Please let us know what you were doing when this error occurred. We will fix it as soon as possible. Sorry for any inconvenience caused.
Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is hosted under a namespace (like Acme/MyBlogBundle).
The namespace should begin with a "vendor" name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it should end with the bundle name itself (which must have Bundle as a suffix).


Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace [Foggyline/TestBundle]:

In your code, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it's really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest FoggylineTestCase.

Bundle name [FoggylineTestBundle]:

Bundles are usually generated into the src/ directory. Unless you're doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?

[Configuration format (annotation, yml, xml, php) [xml]]:

Bundle generation

> Generating a sample bundle skeleton into src/Foggyline/TestBundle OK!
> Checking that the bundle is autoloaded: OK
> Enabling the bundle inside app/AppKernel.php: OK
> Importing the bundle's routes from the app/config/routing.yml file: OK

Everything is OK! Now get to work :)
Brankos-MacBook-Pro:/test-app branko$ phpunit
PHPUnit 4.7.6 by Sebastian Bergmann and contributors.

Time: 545 ms, Memory: 17.00Mb
OK (1 test, 2 assertions)
Chapter 6: Building the Core Module

Welcome to Symfony 3.1.0

Your application is now ready. You can start working on it at:
/Usere/branko/www/shop/

What’s next?
Read the documentation to learn
How to create your first page in Symfony

- web
  - bundles
  -cgi-bin
- css
  - app.css
  - foundation.css
  - foundation.min.css
- img
- js
  - vendor
    - foundation.js
    - foundation.min.js
    - jquery.js
    - what-input.js
    - app.js
Contact Us

Name

Email

Message

Best Sellers

missing image

missing image

missing image

missing image

missing image

iPhone  $49.99

Add to Cart

LG  $19.99

Add to Cart

Samsung  $79.99

Add to Cart

Luna  $19.99

Add to Cart

Edge  $39.99

Add to Cart

On Sale

missing image

missing image

missing image

missing image

missing image

iPhone  $19.99

Add to Cart

LG  $29.99

Add to Cart

Samsung  $39.99

Add to Cart

Luna  $49.99

Add to Cart

Edge  $69.99

Add to Cart

Symfony Password Encoder Utility

Type in your password to be encoded:

[OK] Password encoding succeeded
Authentication Required

http://shop.app requires a username and password. Your connection to this site is not private.

User Name: 
Password: 

[Buttons: Cancel, Log In]
Chapter 7: Building the Catalog Module

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is hosted under a namespace (like Acme\BlogBundle).

The namespace should begin with a "vendor" name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it should end with the bundle name itself (which must have Bundle as a suffix).


Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace [Foggyline\CatalogBundle]:

In your code, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it's really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest FoggylineCatalogBundle.

Bundle name [FoggylineCatalogBundle]:

Bundles are usually generated into the src directory. Unless you're doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?

Configuration format (annotation, yml, xml, php) [xml]:

Bundle generation

> Generating a sample bundle skeleton into src/Foggyline/CatalogBundle OK!
> Checking that the bundle is autoloaded: OK
> Enabling the bundle inside app/AppKernel.php: OK
> Importing the bundle's routes from the app/config/routing.yml file: OK

Everything is OK! Now get to work :)
Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like `AcmeBlogBundle:Post`.

[The Entity shortcut name: FoggylineCatalogBundle:Category]

Determine the format to use for the mapping information.

[Configuration format (yml, xml, php, or annotation) [annotation]]

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named \$id).

Available types: array, simple_array, json_array, object,
boolean, integer, smallint, bigint, string, text, datetime, datetimez,
date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): title
[Field type [string]: Field length [255]:]
[Is nullable [false]:]
[Unique [false]:]

New field name (press <return> to stop adding fields): url_key
[Field type [string]: Field length [255]:]
[Is nullable [false]:]
[Unique [false]: true]

New field name (press <return> to stop adding fields): description
[Field type [string]: text]
[Is nullable [false]: true]
[Unique [false]:]

New field name (press <return> to stop adding fields): image
[Field type [string]: Field length [255]:]
[Is nullable [false]: true]
[Unique [false]:]

New field name (press <return> to stop adding fields):

Entity generation

> Generating entity class src/Foggyline/CatalogBundle/Entity/Category.php: OK!
> Generating repository class src/Foggyline/CatalogBundle/Repository/CategoryRepository.php: OK!

Everything is OK! Now get to work :).

[Brankos-MacBook-Pro:shop branko$ php bin/console doctrine:schema:update --force
Updating database schema...
Database schema updated successfully! "i" query was executed
Brankos-MacBook-Pro:shop branko$]
Welcome to the Doctrine2 CRUD generator

This command helps you generate CRUD controllers and templates.

First, give the name of the existing entity for which you want to generate a CRUD
(use the shortcut notation like AcmeBlogBundle:Post)

The Entity shortcut name: foggylineCatalogBundle:Category

By default, the generator creates two actions: list and show.
You can also ask it to generate "write" actions: new, update, and delete.

Do you want to generate the "write" actions [no]? yes

Determine the format to use for the generated CRUD.

Configuration format (yml, xml, php, or annotation) [annotation]:

Determine the routes prefix (all the routes will be "mounted" under this
prefix: /prefix/, /prefix/new, ...).

Routes prefix [/category]:

Summary before generation

You are going to generate a CRUD controller for "FoggylineCatalogBundle:Category"
using the "annotation" format.

Do you confirm generation [yes]? yes

CRUD generation

Generating the CRUD code: OK
Generating the Form code: OK
Updating the routing: OK

Everything is OK! Now get to work :).
Welcome to the Doctrine2 CRUD generator

This command helps you generate CRUD controllers and templates.
First, give the name of the existing entity for which you want to generate a CRUD (use the shortcut notation like Acme\BlogBundle:Post)

The Entity shortcut name: FoggylineCatalogBundle:Product

By default, the generator creates two actions: list and show.
You can also ask it to generate "write" actions: new, update, and delete.

Do you want to generate the "write" actions [no]? yes

Determine the format to use for the generated CRUD.

Configuration format [yaml, xml, php, or annotation] [annotation]:

Determine the routes prefix (all the routes will be "mounted" under this prefix: /prefix/, /prefix/new, ...).

Routes prefix [/product]:

Summary before generation

You are going to generate a CRUD controller for "FoggylineCatalogBundle:Product" using the "annotation" format.

Do you confirm generation [yes]?

CRUD generation

Generating the CRUD code: OK
Generating the Form code: OK
Updating the routing: OK

Everything is OK! Now get to work :).
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>3</td>
</tr>
<tr>
<td>Title</td>
<td>Blue dress</td>
</tr>
<tr>
<td>Price</td>
<td>$30.00</td>
</tr>
<tr>
<td>SKU</td>
<td>Blue-dress</td>
</tr>
<tr>
<td>Description</td>
<td>Just a description of a sample product.</td>
</tr>
<tr>
<td>Qty</td>
<td>1</td>
</tr>
<tr>
<td>Image</td>
<td>dsda88568b2f49ba474150dd33648086.jpg</td>
</tr>
</tbody>
</table>

- Back to the list
- Edit
- Delete
Chapter 8: Building the Customer Module

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is hosted under a namespace (like Acme/BlogBundle). The namespace should begin with a “vendor” name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it should end with the bundle name itself (which must have Bundle as a suffix).


Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace [Foggyline/CustomerBundle]:

In your code, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it's really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest FoggylineCustomerBundle.

Bundle name [FoggylineCustomerBundle]:

Bundles are usually generated into the src directory. Unless you're doing something custom, hit enter to keep this default!

Target Directory [src]:

What format do you want to use for your generated configuration?

Configuration format (annotation, yml, xml, php) [xml]:

Bundle generation

> Generating a sample bundle skeleton into src/Foggyline/CustomerBundle OK!
> Checking that the bundle is auto-loaded: OK
> Enabling the bundle inside app/AppKernel.php: OK
> Importing the bundle's routes from the app/config/routing.yml file: OK

Everything is OK! Now get to work :)
Branko@MacBook-Pro:shop bran# $ php bin/console doctrine:schema:update --force
Updating database schema...
Database schema updated successfully! "1" query was executed
Branko@MacBook-Pro:shop bran#
Welcome to the Doctrine2 CRUD generator

This command helps you generate CRUD controllers and templates.

First, give the name of the existing entity for which you want to generate a CRUD
(use the shortcut notation like AcmeBlogBundle:Post)

| The Entity shortcut name: FoggylneCustomerBundle:Customer |

By default, the generator creates two actions: list and show.
You can also ask it to generate "write" actions: new, update, and delete.

Do you want to generate the "write" actions [no]? yes

Determine the format to use for the generated CRUD.

| Configuration format (yaml, xml, php, or annotation) [annotation]: |

Determine the routes prefix (all the routes will be "mounted" under this
prefix: /prefix/, /prefix/new, ...).

| Routes prefix [/customer]: |

Summary before generation

You are going to generate a CRUD controller for "FoggylneCustomerBundle:Customer"
using the "annotation" format.

Do you confirm generation [yes]?

CRUD generation

Generating the CRUD code: OK
Generating the Form code: OK
Updating the routing: OK

Everything is OK! Now get to work :).
Chapter 9: Building the Payment Module

Brankoo-MacBook-Pro:step brankoo$ php bin/console generate:bundle --namespace=Foggylime/PaymentBundle

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [n]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is hosted under a namespace (like App/MyBundle). The namespace should begin with a "vendor" name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it should end with the bundle name itself (which must have Bundle as a suffix).


Use / instead of \\ for the namespace delimiter to avoid any problem.

Bundle namespace [Foggylime/PaymentBundle]:

In your case, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it’s really up to you to come up with a unique name (a good practice is to start with the vendor name).

Based on the namespace, we suggest FoggylimePaymentBundle.

Bundle name [FoggylimePaymentBundle]:

Bundles are usually generated into the src/ directory, unless you’re doing something custom, hit enter to keep this default!

Target directory [src/]:

What format do you want to use for your generated configuration?

Configuration format (annotation, yml, xml, php) [xml]:

Bundle generation

> Generating a sample bundle skeleton into src/Foggylime/PaymentBundle OK!
> Checking that the bundle is autoloaded: OK
> Enabling the bundle inside app/AppKernel.php OK
> Exporting the bundle's routes from the appConfig/routing.yml file: OK

Everything is OK! Now get to work :)
Chapter 10: Building the Shipment Module

Brankoo-Macbook-Pro:shop brankoo$ php bin/console generate:bundle --namespace=Foggyline/ShipmentBundle

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [n]:
Your application code must be written in bundles. This command helps you generate them easily.

Give your bundle a descriptive name, like BlogBundle.
Bundle name [Foggyline/ShipmentBundle]:

In your case, a bundle is often referenced by its name. It can be the concatenation of all namespaces but it’s really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest Foggyline/ShipmentBundle.

Bundle name [Foggyline/ShipmentBundle]:

Bundles are usually generated into the src/ directory. Unless you’re doing something custom, hit enter to keep this default!

Target directory [src/]:

What format do you want to use for your generated configuration?
Configuration format (annotation, yml, xml, php) [annotation]:

Bundle generation

• Generating a sample bundle skeleton into src/Foggyline/ShipmentBundle OK!
• Checking that the bundle is autoloaded: OK
• Enabling the bundle inside app/kernel.php OK
• Importing the bundle’s routes from the module/config/routing.yml file: OK
• Importing the bundle’s services.yml from the app/config/services.yml file: OK

Everything is OK! Now get to work!}
Chapter 11: Building the Sales Module

Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is hosted under a namespace (like Acme/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it should end with the bundle name itself (which must have Bundle as a suffix).


Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace [Foggyline/SalesBundle]:

In your code, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it's really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest FoggylineSalesBundle.

Bundle name [FoggylineSalesBundle]:

Bundles are usually generated into the src/ directory. Unless you're doing something custom, hit enter to keep this default!

Target Directory [src/:] :

What format do you want to use for your generated configuration?

[Configuration format (annotation, yml, xml, php) [xml]:

> Bundle generation

> Generating a sample bundle skeleton into src/Foggyline/SalesBundle OK!
> Checking that the bundle is autoloaded: OK
> Enabling the bundle inside app/AppKernel.php: OK
> Importing the bundle's routes from the app/config/routing.yml file: OK

Everything is OK! Now get to work :)
Before completing this step, make sure you've created a symfony bundle named "SalesBundle" in your project.

Welcome to the Symfony bundle generator.

Are you planning on sharing this bundle across multiple applications? [no]: yes

Your application code must be written in bundles. This command helps you generate them easily.

Each bundle is nested under a namespace like `App\Bundle\MyBundle`. The namespace should begin with a "vendor" name like your company name, your project name, or your client name, followed by one or more optional category sub-namespaces, and it would end with the bundle name itself (which must have `Bundle` as a suffix).


Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace [App\Bundle\Salesbundle]:

In your code, a bundle is often referenced by its name. It can be the concatenation of all namespace parts but it's really up to you to come up with a unique name (a good practice is to start with the vendor name). Based on the namespace, we suggest `App\Bundle\Salesbundle`.

Bundle name [App\Bundle\Salesbundle]:

Bundles are usually generated into the src/ directory, unless you're doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?

Configuration format (annotation, xml, yml, php) [yml]:

Bundle generation

- Generating a sample bundle skeleton into App\Bundle\Salesbundle OK
- Checking that the bundle is auto-loaded OK
- Enabling the bundle into `composer.json` OK
- Importing the bundle's routes from the app/config/routing.yml file OK

Everything is OK! Now get to work :).
You have new mail in /var/mail/branko
[Brankos-MacBook-Pro:shop branko]$ php bin/console generate:doctrine:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate.
You must use the shortcut notation like AcmeBlogBundle:Post.

[The Entity shortcut name: FoggyleSalesBundle:Cart]

Determine the format to use for the mapping information.

[Configuration format (xml, php, or annotation) [annotation]:]

Instead of starting with a blank entity, you can add some fields now.
Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object,
boolean, integer, smallint, bigint, string, text, datetime, datetimetz,
date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): customer
[Field type [string]: integer
[Is nullable [false]:
[Unique [false]:

New field name (press <return> to stop adding fields): created_at
[Field type [datetime]:
[Is nullable [false]:
[Unique [false]:

New field name (press <return> to stop adding fields): modified_at
[Field type [datetime]:
[Is nullable [false]:
[Unique [false]:

New field name (press <return> to stop adding fields):

Entity generation

> Generating entity class src/FoggyleSalesBundle/Entity/Cart.php: OK!
> Generating repository class src/FoggyleSalesBundle/Repository/CartRepository.php: OK!
Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like AcmeBlogBundle:Post.

Entity shortcut name: FoggylineSalesBundle:Cart

Determine the format to use for the mapping information.

Configuration format (xml, xml, php, or annotation) [annotation]:

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object, boolean, integer, smallint, bigint, string, text, datetime, datetimetz, date, time, decimal, float, binary, blob, guid.

New field name [press <return> to stop adding fields]: customer
[Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name [press <return> to stop adding fields]: created_at
[Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name [press <return> to stop adding fields]: modified_at
[Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name [press <return> to stop adding fields]:

Entity generation

> Generating entity class src/Foggyline/SalesBundle/Entity/Cart.php: OK!
> Generating repository class src/Foggyline/SalesBundle/Repository/CartRepository.php: OK!

Everything is OK! Now get to work :)
brankos-mbp:shop brankos$ php bin/console generate:doctrine:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like *AcmeBlogBundle:Post*.

The Entity shortcut name: *FoggylineSalesBundle:CartItem*

Determine the format to use for the mapping information.

Configuration format (yml, xml, php, or annotation) [annotation]:

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named *id*).

Available types: array, simple_array, json_array, object, boolean, integer, smallint, bigint, string, text, datetime, datetimetz, date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): cart
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): qty
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): unit_price
Field type [string]: decimal
Precision [10]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): created_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): modified_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:
Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like Acme\BlogBundle\Entity\Post.

The Entity shortcut name: FoggyLineSalesBundle\CartItem

Determine the format to use for the mapping information.

Configuration format (yml, xml, php, or annotation) [annotation]:

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object, boolean, integer, smallint, bigint, string, text, datetime, datetimetz, date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): cart
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): qty
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): unit_price
Field type [string]: decimal
Precision [18]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): created_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): modified_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

Entity generation

> Generating entity class src/FoggyLine/SalesBundle/Entity/CartItem.php: OK!
> Generating repository class src/FoggyLine/SalesBundle/Repository/CartItemRepository.php: OK!

Everything is OK! Now get to work :)
brankos-mbp:shop branka$ php bin/console generate:doctrine:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.

First, you need to give the entity name you want to generate. You must use the shortcut notation like AcmeBlogBundle:Post.

The Entity shortcut name: FogylineSalesBundle:Order

"Order" is a reserved word.

The Entity shortcut name:
The Entity shortcut name: SensioGeneratorBundle:SalesOrder

Determine the format to use for the mapping information.

Configuration format (yml, xml, php. or annotation) [annotation]:

Instead of starting with a blank entity, you can add some fields now. Note that the primary key will be added automatically (named id).

Available types: array, simple_array, json_array, object, boolean, integer, smallint, bigint, string, text, datetime, datetimetz, date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): customer
  Field type [string]: integer
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): items_price
  Field type [string]: decimal
  Precision [10]:
  Scale: 4
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): shipment_price
  Field type [string]: decimal
  Precision [10]:
  Scale: 4
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): total_price
  Field type [string]: decimal
  Precision [10]:
  Scale: 4
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): status
  Field type [string]:
  Field length [255]:
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): payment_method
  Field type [string]:
  Field length [255]:
  Is nullable [false]:
  Unique [false]:

New field name (press <return> to stop adding fields): shipment_method
The Entity shortcut names: SensioGeneratorBundle\SalesOrder

Determine the format to use for the mapping information.

Configuration format [yaml, xml, php, or annotation] [annotation]

Instead of starting with a blank entity, you can add some fields now.
Note that the primary key will be added automatically named "id".

Available types: array, simple_array, json_array, object, string, integer, floating, blob, array, bool, integer, boolean, date, time, decimal, float, binary, blob, guid.

New field name (press <return> to stop adding fields): customer
Field type [string]: integer
Is nullable [true]:
Unique [false]:

New field name (press <return> to stop adding fields): items_price
Field type [string]: decimal
Precision [10]:
Scale [4]:
Is nullable [true]:
Unique [false]:

New field name (press <return> to stop adding fields): shipment_price
Field type [string]: decimal
Precision [10]:
Scale [4]:
Is nullable [true]:
Unique [false]:

New field name (press <return> to stop adding fields): total_price
Field type [string]: decimal
Precision [10]:
Scale [4]:
Is nullable [true]:
Unique [false]:

New field name (press <return> to stop adding fields): status
Field type [string]:
Field length [90]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): payment_method
Field type [string]:
Field length [90]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): shipment_method
Field type [string]:
Field length [90]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): created_at
Field type [datetime]:
Is nullable [true]:
Unique [false]:

New field name (press <return> to stop adding fields): modified_at
Field type [datetime]:
Is nullable [true]:
Unique [false]:
New field name (press <return> to stop adding fields): customer_email
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): customer_first_name
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): customer_last_name
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:
New field name (press <return> to stop adding fields): address_first_name
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_last_name
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_country
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_state
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_city
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_postcode
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_street
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): address_telephone
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):
New field name (press <return> to stop adding fields): address_first_name
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_last_name
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_country
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_state
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_city
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_postcode
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_street
Field type: string
Field length: 255
Is nullable: false
Unique: false

New field name (press <return> to stop adding fields): address_telephone
Field type: string
Field length: 255
Is nullable: false
Unique: false

Entity generation

> Generating entity class vendor/sensio/generator-bundle/Entity/SalesOrder.php: OK!
> Generating repository class SensioBundle/GeneratorBundle/Repository/SalesOrderRepository.php: OK!

Everything is OK! Now get to work :)
New field name (press <return> to stop adding fields): sales_order
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): product
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): title
Field type [string]:
Field length [255]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): qty
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): unit_price
Field type [string]: decimal
Precision [10]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): total_price
Field type [string]: decimal
Precision [10]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): created_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): modified_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):
New field name (press <return> to stop adding fields): sales_order
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): product
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): title
Field type [string]:
Field length [20]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): qty
Field type [string]: integer
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): unit_price
Field type [string]: decimal
Precision [2]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): total_price
Field type [string]: decimal
Precision [10]:
Scale: 4
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): created_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields): modified_at
Field type [datetime]:
Is nullable [false]:
Unique [false]:

New field name (press <return> to stop adding fields):

> Generating entity class src/Foggyline/SalesBundle/Entity/SalesOrderItem.php: OK!
> Generating repository class src/Foggyline/SalesBundle/Repository/SalesOrderItemRepository.php: OK!

Everything is OK! Now get to work :).
Chapter 12: Integrating and Distributing Modules

[Image of a web page showing a directory structure and a search form for submitting packages to a package repository.]

[Search form with a repository URL field and a check button.]

Submit package

The package name was not found in the composer.json, make sure there is a name present.
foggyline/catalogbundle

Just a test module for web shop application.