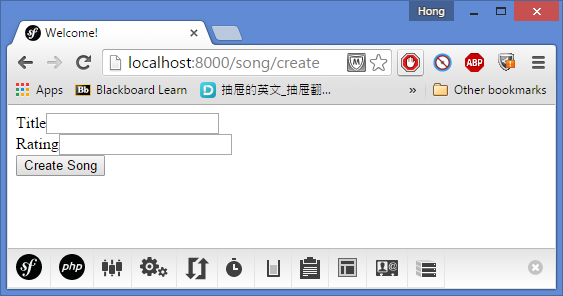
Book – form section

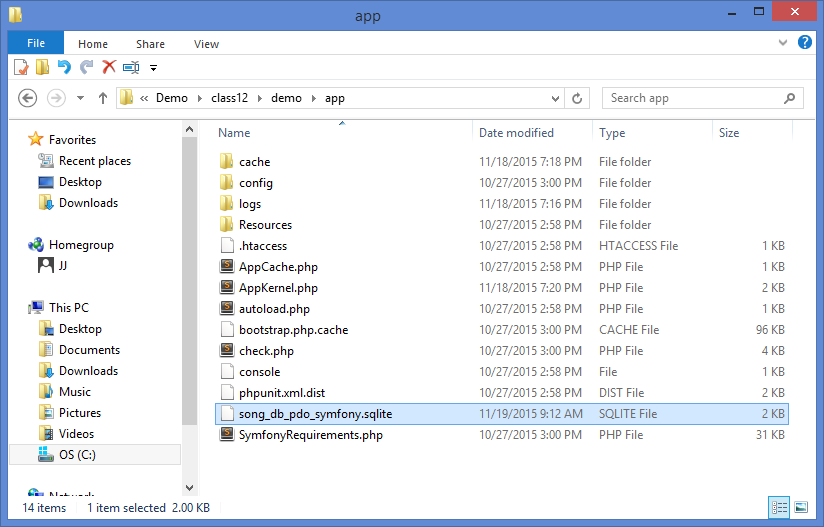
1. Create entity: Demo/src/AppBundle/Entity/song.php
2. Create controller: Demo/src/AppBundle/Controller/SongController.php
3. Create templet: Demo/ app/Resources/views/song/createsong.html.twig



Symphony’s default is required for every field is true.

1. Update the index: Demo/app/Resources/views/default/index.html.twig (change)
2. Save the data:

I have went ahead and reworked the demo code slightly to save the song into an sqlite database using the repository. Namespaces had to change and use statements had to be added to conform to symfony but for the most part it is the same repository. The other thing I noticed was it was creating the sqlite database file in the web directory. That is the public web directory so the database probably shouldn't be there and it should be in the app directory or somewhere else. In the song controller I used the service container to get the root directory from the kernel and then passed it to the new repository as a parameter. I then modified the repository constructor to take that parameter and use it to build the path I want the sqlite file saved in. In this case the app directory. The app directory is what is returned by thegetRootDir() kernel method you see on line 37 in the song controller. It would have been fine to leave the database in the web directory for this but I wanted to set it up the way I thought it should be. I also had to modify the Song entity to add an id property. Pay close attention to the namespaces and use statments in the new files. I only modified a few files vs what we did in the class demo. I added the ISongRepository.php and SqliteSongRepository.php files. I made some changes to the Song.php and SongController.php files. The rest is the same as the demo.



Analysis of symphony\_demo:

Data: final/symphony\_demo/app/data/blog.sqlite

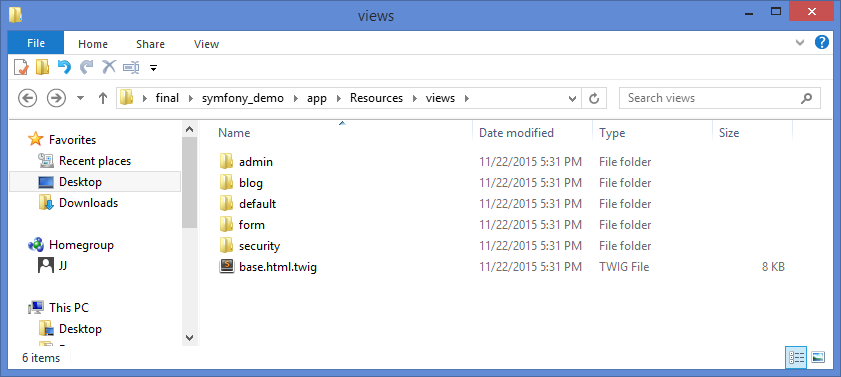
Css: final/symphony\_demo/app/Resources/assets/css

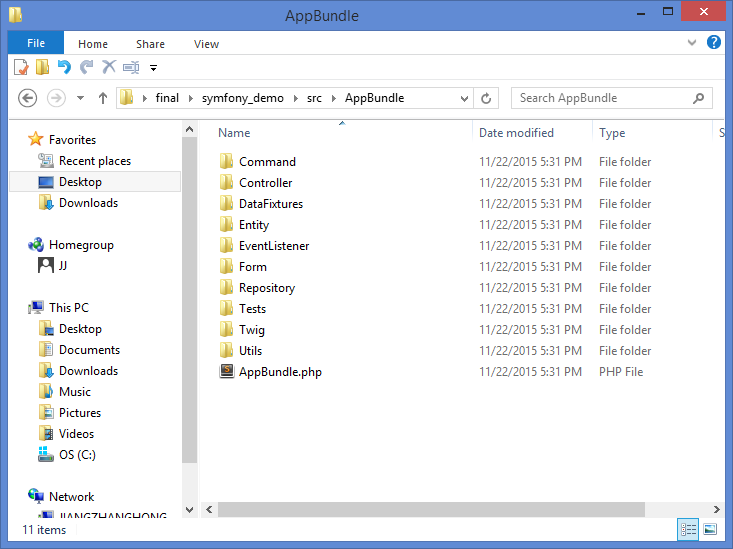
final/symphony\_demo/app/assets/js

final/symphony\_demo/app/Resources/assets/scss

catch error: final/symphony\_demo/app/Resources/TwigBundle

views:





My Final Guildline

1. Create entity: Src/AppBundle/Entity.

Notes: Post.php;

Users: User.php

1. Create form type: Src/AppBundle/Form.

Public Section: CommentType.php;

Edit Section: PostType.php;

Time type: /Type/ DateTimePickerType.php.

1. Create Repository: Src/AppBundle/ Repository.

Notes: PostRepository;

Users: UserRepository.

1. Create controller: Src/AppBundle/Controller.

Login: SecurityController.php;

Public Section: NoteController.php;

Edit Section: /Admin/ NoteController.php.

1. Create Template: app/Resources/views.

Basic Template: base.html.twig;

HomePage Template: /defalt/ homepage.html.twig;

Flash Message Template: /defalt/ \_flash\_messages.html.twig;

Time Template: /form/ fields.html.twig

Login Template: /security/ login.html.twig

Public Section Template: note/\_comment\_form.html.twig;

note/\_delete\_post\_confirmation.html.twig;

note/comment\_form\_error.html.twig;

note/index.html.twig;

note/post\_show.html.twig

Edit Section Template: basic template – layout.html.twig;

note/ \_form.html.twig

note/ edit.html.twig

note/ index.html.twig

note/ new.html.twig

note/ show.html.twig

1. Create SQLite: data/ note.sqlite
2. Configration Part: app/config. Need to configure routing, security and parameters. Update SQLite’s Path (database\_path: '%kernel.root\_dir%/data/note.sqlite'): app/config/ parameters.yml. app/config/config.yml (the main configuration file of your application. It stores all the common options for every execution environment ('prod', 'dev', 'test').)
3. Create new user (use cmd to create: $ php app/console app:add-user): Src/AppBundle/Command. Add user: AddUserCommand.php; Delete user: DeleteUserCommand.php; List user: ListUsersCommand.php.
4. Update autoload classes and namespaces: vendor/composer/ autoload\_classmap.php, autoload\_namespaces.php and ClassLoader.php.
5. Update CSS: web folder.

**Run the server: php app/console server:run**