



Dingo

Dindo Document

DingoLab
June, 2016

Dingo Dindo Document

李约瀚

qinka@live.com

14130140331

June 1st 2016

Version 0.0.3.0

西安, Xi'an

DingoLab

前言

这个文档是 Dingo 后端 Dindo 的文档，包括后端的大体需求说明，宏观设计说明、详细设计说明、数据库设计与实现、软件源码说明、软件测试说明、软件部署说明件与软件使用说明。

后端 Dindo 使用 Haskell ¹，与 Yesod 框架 ² 编写的。同时整个后端代码中 Haskell 的部分是使用 Haskell 与 L^AT_EX 混排的文学编程。所以文档中有一部分为程序代码（及其说明）。

Dindo 的名称由来是在笔者（也是主要维护者）在数学建模的校赛是，使用 Lingo 是受到 Lingo 与 Lindo 的关系而起的名字。

这个后端依次将介绍需求、设计、数据库设计、软件部署、软件使用与维护、Dindo 代码及其说明等内容，以上是正文部分。附录中将会有部分术语表、维护的文档、索引、参考文档等。

¹Haskell 是一门纯函数式的编程语言。

²Yesod 是一个使用 Haskell 作为主要语言，的 RESTful API 的 WEB 应用框架。

目录

1 大体需求说明	1
2 Dindo 架构设计概论	1
3 均衡负载设计	1
4 弹性计算设计	2
5 微服务架构设计	2
6 业务流程说明	2
7 数据库设计	2
8 Dindo 部署说明	2
8.1 测试部署方式	2
8.1.1 原生运行	2
9 Dindo 软件使用与维护说明	4
10 Dindo 源码及说明	4
10.1 dindo-database	4
10.1.1 src/Dindo/Database.lhs	4
10.1.2 src/Import.lhs	8
10.2 dindo-common	8
10.2.1 src/Dindo/Import.lhs	8
10.2.2 src/Dindo/Import/Aeson.lhs	9
10.2.3 src/Dindo/Import/ByteString.lhs	9
10.2.4 src/Dindo/Import/Database.lhs	9
10.2.5 src/Dindo/Import/Digest.lhs	10
10.2.6 src/Dindo/Import/Rable.lhs	10
10.2.7 src/Dindo/Import/Text.lhs	11
10.2.8 src/Dindo/Import/TH.lhs	11
10.2.9 src/Dindo/Import/Yaml.lhs	11
10.2.10 src/Dindo/Import/Yesod.lhs	12

10.2.11	src/Dindo/Common.lhs	13
10.2.12	src/Dindo/Common/Auth.lhs	13
10.2.13	src/Dindo/Common/Rable.lhs	17
10.2.14	src/Dindo/Common/Yesod/Config.lhs	21
10.2.15	src/Dindo/Common/Yesod/Launch.lhs	25
10.2.16	src/Dindo/MicroFramework/API.lhs	26
10.2.17	src/Dindo/MicroFramework/Destory.lhs	26
10.2.18	src/Dindo/MicroFramework/Register.lhs	27
10.3	dindo-launch	29
10.3.1	src/Main.lhs	29
10.4	dindo-usrmanage	32
10.4.1	src/Dindo/Std.lhs	32
10.4.2	src/Dindo/UM.lhs	32
10.4.3	src/Dindo/UM/Data.lhs	33
10.4.4	src/Dindo/UM/Foundation.lhs	38
10.4.5	src/Dindo/UM/Handler.lhs	40
10.5	dindo-tools	49
10.5.1	src/pash/Main.lhs	49
11	Dindo 公共组件	51
12	Dindo 数据库	51
13	Dindo Launcher	51
14	Dindo 微服务组件——用户管理	51
15	DIndo 测试说明	51
15.1	如何测试	51
A	术语解释	52
B	Docker 中 Weave 的配置	52
C	后端附带工具使用说明	52
C.1	dindo-pash	52

目录	iii
D 发行（发布）的二进制文件镜像与包的命名规则	53

1 大体需求说明

2 Dindo 架构设计概论

Dindo 是 Dingo 的核心部分之一，负责客户端与后端的交互，同时负责客户端与数据库的、客户端之间的间接交互。此部分将有负载均衡的大致方法、弹性计算的解决方案、后端 API 与服务程序分割的内容。同时还将说明后端业务流程。

Dindo 是基于 Docker 容器上，采用微服务架构的一个后端。所有的组件将运行与 Docker 容器之中，且方便运行与公有云搭建的 Docker 中，同时价格相对比较便宜。按照灵雀云的收费标准 [1]，按照北京一区（AWS）来计算。当不使用弹性计算中的策略，即仅当容器的大小与数量时确定不变时。负载均衡负载的采用一个 M 级别的容器，运行 5 个 L 级别的容器作为数据库，运行 20 个的 M 级别的容器为处理业务的核心部分。数据库每个容器配置 100G 的挂载点用于存放数据，并计划每天下载数据量有 10G。按上述配置需要³

$$((20 + 1) * 0.329 + 5 * 0.658) * 24 * 30 + 10 * 30 * 0.93 + 0.75 * 100 * 5 = 7997.28$$

每个月大致需要不到 8000 元的成本⁴。

Dindo 开发过程依赖敏捷开发，并采用以持续集成为主的测试方式测试，同时采用持续交付的方式交付运营者。由于采用微服务架构、持续交付与 Docker 可以使得后端的版本升级处于“无痛”状态。微服务架构也能使的后端的业务逻辑分布在不同的程序（组件），也可使得后端分布上线。

3 均衡负载设计

均衡负载采用 Nginx 作负载均衡的软件，

³一个月按 30 天计算。

⁴当采用弹性计算时，这个成本将继续下降

4 弹性计算设计

5 微服务架构设计

6 业务流程说明

业务流程部分包括后端对事件驱动型的业务处理过程，每个 API 中业务处理过程等。这部分的主要内容将在 Dindo 源码及其结束的部分说明。

7 数据库设计

8 Dindo 部署说明

此部分主要说明 Dindo 的部署问题，包括测试、原型与最后实际运行是的部署。测试与原型的部署有两种方式，一种是直接运行，另一种是基于 Docker⁵。而最后运营是的部署，目前计划直接部署公有云之上，利用 CaaS 服务。

8.1 测试部署方式

测试的部署一般适用于调试与检测。调试一方面是指后端开发时测试验证，另一方面则是指前端开发时测试使用。检测是如安全性测试等方面的检测。而通常运营部署通常不需要调试磨合，直接部署到 CaaS 提供商即可。

8.1.1 原生运行

原生运行首先要构建⁶然后部署，最后运行。如果已获得构建好的二进制文件，请直接跳过下面构建的过程。

Windows 下的构建 首先需要安装 [Haskell Platform 7.10.3 x64](#)，然后克隆 [GitHub/Dingo-Lab/DingoBackend](#) 仓库到本地，然后安装 stack，安装方式可参考 [Stack Install & Upgrade](#)。安装完之后跳转到仓库的目录：

```
$ cd DingoBackend
```

⁵基于的是 Ubuntu (Linux) 原声的 Docker，暂不讨论 Mac OS X 与 Windows 下原生的 Docker。

⁶Dindo 是不直接发行二进制文件的，发行的只有 Docker 镜像。

然后执行构建：

```
$ stack build
```

然后在 `.stack_work` 文件夹中某个文件夹下面的 `bin` 文件夹中可以找到编译好的二进制文件⁷。

Linux 下的构建 首先安装 GHC⁸。安装的方式通常通过

Max OS 下的构建 部署的方式分为两部分：后端组件与数据库。由于处于测试的目的，并不需要使用均衡负载与法务发现的部分。所以直接载入配置文件就可以启动。对于数据库，要求是实用 PostgreSQL 数据库，并使用 `dindo-database` 模块中的 SQL 文件初始化数据库并使用。

后端模块的启动 无论是在那个系统下，当获得某个模块的二进制文件时。运行这个文件再将配置传入即可。通常在 UNIX Shell⁹ 或与之类似的 Shell 环境中¹⁰ 以用户管理模块为例，假设文件 `config.yml` 为 YAML 格式的配置文件，则输入如下：

```
$ cat config.yml | dindo-um --form=yaml
```

就可以启动用户管理部分的模块。其中 `config.yml` 文件的内容如下

```
1  port: 3000
2  database-config:
3    addr: '192.168.1.224'
4    port: '5432'
5    user: postgres
6    name: dingo
7    con-limit: 10
8    password: abcdefg
```

其中 `port` 是指该模块侦听的端口，`database-config` 部分是数据库的配置。由上到下依次是：数据库地址、数据库侦听端口、数据库用户名、数据库名称、数据库连接数限制与用户密码。启动配置还可以是 JSON 格式：

⁷为何不直接搜索。

⁸要求 7.10 以上，之前的版本没有测试过，无法保证可以正常编译运行。

⁹比如 Bash、Zsh 等。

¹⁰例如 Windows 下的 PowerShell。

```
1 { "port":3000
2   , "database-config":
3     { "addr" : "192.168.1.224"
4       , "port" : "5432"
5       , "user" : "postgres"
6       , "name" : "dingo"
7       , "con-limit" : 10
8       , "password" : "johnjing"
9     }
10 }
```

同时启动的命令是：

```
$ cat config.json | dindo-um
```

其中默认的文件格式是 JSON，然而推荐使用 YAML 的格式。同时还可以直接执行可执行文件，然后通过标准输入键入，然后输入文件结束符 EOF ¹¹。

9 Dindo 软件使用与维护说明

10 Dindo 源码及说明

这一部分是关于 Dindo 源代码及其解释说明。

10.1 dindo-database

这一部分的功能是数据库驱动与数据库内容的表示。

10.1.1 src/Dindo/Database.lhs

数据库内容

```
1 module Dindo.Database where
2 import Prelude hiding (String)
```

¹¹Windows 下按 Ctrl + Z, Linux 与 Mac 按 Ctrl + D

```

3  import Import
4  import Data.Text
5  import Data.ByteString
6  import Paths_dindo_database
7  import Data.Version

8  instance FromJSON ByteString where
9      parseJSON (String x) = pure $ encodeUtf8 x
10 instance ToJSON ByteString where
11     toJSON = String . decodeUtf8

12 share [mkPersist sqlSettings] [persistLowerCase|
13 Account json sql=table_account
14     Id sql=
15     uid Text sql=key_uid sqltype=vchar(64)
16     pash Text sql=key_pash sqltype=vchar(64)
17     tel Int sql=key_tel
18     name Text sql=key_name sqltype=vchar(64)
19     Primary uid
20     deriving Show Eq
21 Usr json sql=table_usr
22     Id sql=
23     uid Text sql=key_uid sqltype=vchar(64)
24     email Text sql=key_email
25     rname Text sql=key_rname sqltype=vchar(64)
26     prcid Text sql=key_prcid sqltype=vchar(18)
27     addr Text sql=key_addr
28     status Text sql=key_status sqltype=vchar(1)
29     Primary uid
30     Foreign Account fkuid uid
31     deriving Show Eq
32 Addr json sql=table_addr
33     Id sql=
34     aid Text sql=key_aid sqltype=vchar(64)

```

```

35     uid Text sql=key_uid sqltype=varchar(64)
36     zip Text sql=key_zip sqltype=varchar(64)
37     addr Text sql=key_addr
38     Primary aid
39     Foreign Account fkaddruid uid
40     deriving Show Eq
41 Apic sql=table_apic
42     Id sql=
43     pid Text sql=key_pic_id sqltype=varchar(64)
44     uid Text sql=key_uid sqltype=varchar(64)
45     bpic ByteString sql=binary_pic
46     typ Int Maybe sql=key_status default=0
47     Primary pid
48     Foreign Account fkuidb uid
49     deriving Show Eq
50 Task json sql=table_task
51     Id sql=
52     tid Text sql=key_tid sqltype=varchar(64)
53     ca Text Maybe sql=key_ca sqltype=varchar(64)
54     cb Text Maybe sql=key_cb sqltype=varchar(64)
55     Primary tid
56     Foreign Account fkca ca
57     Foreign Account fkvcb cb
58     deriving Show Eq
59 Taskinfo json sql=table_task_info
60     Id sql=
61     tid Text sql=key_tid sqltype=varchar(64)
62     ew Double sql=key_ew
63     ns Double sql=key_ns
64     r Double sql=key_r
65     wei Double sql=key_wei
66     size [Double] sql=key_size
67     note Text Maybe sql=key_note
68     cost Int sql=key_cost

```

```

69     des Text Maybe sql=key_des
70     Primary tid
71     Foreign Task fktid tid
72     deriving Show Eq
73 Taskcost json sql=table_task_cost
74     Id sql=
75     tid Text sql=key_tid sqltype=varchar(64)
76     ad [Int] sql=key_ad
77     bd [Int] sql=key_bd
78     Primary tid
79     Foreign Task fktidb tid
80     deriving Show Eq
81 Dd json sql=table_dd
82     Id sql=
83     did Text sql=key_did sqltype=varchar(64)
84     uid Text sql=key_tid sqltype=varchar(64)
85     dd Text sql=key_dd
86     ew Double sql=key_ew
87     ns Double sql=key_ns
88     r Double sql=key_r
89     Primary did
90     Foreign Account fkuidc uid
91     deriving Show Eq
92 TmpToken json sql=table_tmptoken
93     Id sql=
94     tt Text sql=key_tmptoken sqltype=varchar(150)
95     time UTCTime sql=key_timeup
96     uid Text sql=key_uid sqltype=varchar(64)
97     Primary tt
98     Foreign Account fkuidd uid
99     deriving Show Eq
100 ]

```

```

101 dindo_database_version = version

```

```
102 | dindo_database_version_quasi = stringE $ showVersion version
```

10.1.2 src/Import.lhs

用于本模块的导入内容，不导出

```
1 | module Import
2 | ( module X
3 | , persistFileWithC
4 | ) where
5 |
6 | import Language.Haskell.TH as X
7 | import Data.Aeson as X
8 | import Database.Persist as X
9 | import Data.Text.Encoding as X
10 | import Database.Persist.TH as X
11 | import Database.Persist.Quasi as X
12 | import Data.Time as X
13 |
14 | persistFileWithC :: PersistSettings
15 |                 -> FilePath
16 |                 -> Q Exp
17 | persistFileWithC s = persistFileWith s (" ../dindo-config/"++)
```

10.2 dindo-common

这一部分是 dindo 各个微组件使用的基础公共设施。

10.2.1 src/Dindo/Import.lhs

这个系列的模块是用来导入模块的，以减少代码重复度

```
1 | module Dindo.Import
2 | ( module X
3 | ) where
```

```
4 import Data.Maybe as X
5 import Data.Time as X
6 import Dindo.MicroFramework.Register as X
7 import Dindo.MicroFramework.Destroy as X
8 import Dindo.MicroFramework.API as X
9 import Data.Conduit as X
```

10.2.2 src/Dindo/Import/Aeson.lhs

导入 Data.Aeson 及相关内容

```
1 module Dindo.Import.Aeson
2   ( module X
3   ) where
4   import Data.Aeson as X
```

10.2.3 src/Dindo/Import/ByteString.lhs

导入 bytestring 包中相关模块

```
1 module Dindo.Import.ByteString
2   ( module X
3   , fromStrictBS
4   ) where
5
6   import Data.ByteString as X
7   import Data.ByteString.Lazy
8   fromStrictBS = fromStrict
```

10.2.4 src/Dindo/Import/Database.lhs

导入与数据库相关的模块

```
1 module Dindo.Import.Database
2   ( module X
```

```

3 |     , tryRunDB
4 |   ) where

5 |   import Database.Persist as X
6 |   import Database.Persist.Postgresql as X
7 |   import Dindo.Database as X
8 |   import Control.Exception
9 |   import Yesod

10 |   tryRunDB :: ( Yesod site
11 |                , YesodPersist site
12 |                , YesodPersistBackend site ~ SqlBackend
13 |                )
14 |           => YesodDB site a -> HandlerT site IO (Either SomeException a)
15 |   tryRunDB f = do
16 |     runInnerHandler <- handlerToIO
17 |     liftIO $ try $ runInnerHandler $ runDB f

```

10.2.5 src/Dindo/Import/Digest.lhs

导入与摘要算法有关的内容模块

```

1 | module Dindo.Import.Digest
2 |   ( module X
3 |   ) where
4 |   import Data.Digest.Pure.SHA as X

```

10.2.6 src/Dindo/Import/Rable.lhs

导入返回值有关的内容模块

```

1 | module Dindo.Import.Rable
2 |   ( module X
3 |   ) where

```



```
4 import Dindo.Common.Rable as X
5 import Text.Hamlet.XML as X
6 import Text.XML as X
```

10.2.7 src/Dindo/Import/Text.lhs

导入 text 包中相关的模块

```
1 module Dindo.Import.Text
2   ( module X
3   , showT
4   ) where
5
6   import Data.Text as X
7   import Data.Text.Encoding as X
```

```
8 showT :: Show a => a -> Text
9 showT = pack.show
```

10.2.8 src/Dindo/Import/TH.lhs

导入与 TemplateHaskell 与 QuasiQuote 有关的模块

```
1 module Dindo.Import.TH
2   ( module X
3   ) where
4
5   import Language.Haskell.TH as X
6   import Language.Haskell.TH.Syntax as X
```

10.2.9 src/Dindo/Import/Yaml.lhs

导入与 Yaml 有关模块

```
1 module Dindo.Import.Yaml
```

```

2  ( module X
3  ) where
4  import Data.Yaml as X

```

10.2.10 src/Dindo/Import/Yesod.lhs

导入与 Yesod 有关的模块

```

1  module Dindo.Import.Yesod
2  ( module X
3  , mkYesodData
4  , mkShomeR
5  ) where

6  import Yesod as X hiding (mkYesodData)
7  import qualified Yesod (mkYesodData)
8  import Dindo.Common.Rable as X
9  import Dindo.Common.Auth as X
10 import Dindo.Common.Yesod.Launch as X
11 import Dindo.Common.Yesod.Config as X
12 import Dindo.Import.TH
13 import Data.Maybe
14 import Data.Time
15 import Data.Text
16 import qualified Data.Text.Encoding as TE
17 import Data.Aeson
18 import Data.ByteString.Lazy as BL hiding(unpack)

19 mkYesodData a b = Yesod.mkYesodData a b'
20   where
21     b' = b ++ [parseRoutes|/ ShomeR GET|]
22     homeR :: Yesod site
23           => Text
24           -> HandlerT site IO Text
25     homeR info = do

```

```

26     addD' <- lookupGetParam "add"
27     let addD = fromRational $ toRational $ fromMaybe 0 $ fmap (read.unpack) addD'
28     now <- fmap (show.addUTCTime addD) $ liftIO getCurrentTime
29     return $ TE.decodeUtf8 $ toStrict $ encode $ object
30         [ "server-time" .= now
31         , "server-info"  .= info
32         ]
33 mkShomeR :: Text -> Q [Dec]
34 mkShomeR info = [d|
35     getShomeR :: Yesod site => HandlerT site IO Text
36     getShomeR = homeR info
37     |]

```

10.2.11 src/Dindo/Common.lhs

提供版本号的部分

```

1 module Dindo.Common
2   ( dindo_common_version
3   , dindo_common_version_quasi
4   ) where
5
6   import Data.Version
7   import Paths_dindo_common
8   import Language.Haskell.TH
9   import Language.Haskell.TH.Syntax
10
11   dindo_common_version = version
12   dindo_common_version_quasi = stringE $ showVersion version

```

10.2.12 src/Dindo/Common/Auth.lhs

提供身份确认的函数的部分

```

1 module Dindo.Common.Auth

```

```

2   ( runPash
3     , tokenAuth
4     , pskAuth
5     , noAuth
6     , fromEntity
7     , pickF
8     , pickU
9     , getUid
10  ) where

```

```

11  import Yesod
12  import Database.Persist
13  import Database.Persist.Sql
14  import Dindo.Database
15  import Data.Time
16  import Data.Text.Encoding
17  import Data.Maybe
18  import qualified Data.ByteString as B
19  import qualified Data.ByteString.Lazy as B hiding (concat, ByteString)
20  import Data.Text (unpack, pack, Text)
21  import Data.Digest.Pure.SHA

```

```

22  pickU [] = []
23  pickU ((y, Just x):oth) = (y ==. x):pickU oth
24  pickU ((_, Nothing):oth) = pickU oth
25  pickF [] = []
26  pickF ((y, Just x):oth) = (y ==. x):pickF oth
27  pickF ((_, Nothing):oth) = pickF oth
28  getUid :: ( Yesod site
29             , YesodPersist site
30             , YesodPersistBackend site ~ SqlBackend
31             )
32      => HandlerT site IO Text
33  getUid = do

```

```

34     tt' <- lookupHeader "TMP-TOKEN"
35     let Just tt = fmap decodeUtf8 tt'
36     rt':_ <- liftHandlerT $ runDB $ selectList [TmpTokenTt ==. tt] []
37     let rt = fromEntity rt'
38     return $ tmpTokenTt rt

```

用于用户验证的 runPash 0 -> uid 1 -> name 2 -> tel

```

39 runPash :: Int -> B.ByteString -> Text -> Text
40 runPash i time pash = pack $ showDigest $ sha512 $ B.fromStrict $ B.concat [pre,
    encodeUtf8 pash,time]
41 where
42     pre = case i of
43         0 -> "uid"
44         1 -> "nnnn"
45         2 -> "+86"
46 runPash _ _ x = id x
47 noAuth :: Yesod site => HandlerT site IO AuthResult
48 noAuth = return Authorized
49
50 tokenAuth :: ( Yesod site
51                , YesodPersist site
52                , YesodPersistBackend site ~ SqlBackend
53                )
54            => HandlerT site IO AuthResult
55 tokenAuth = do
56     token' <- lookupHeader "TMP-TOKEN"
57     case token' of
58         Nothing -> return $ Unauthorized "Who_are_you!"
59         Just token -> do
60             rt' <- liftHandlerT $ runDB $ selectList [TmpTokenTt ==. decodeUtf8 token][
                Desc TmpTokenTime]
61             case rt' of
62                 rt:_ -> do
63                     now <- liftIO getCurrentTime

```

```

64         let time = tmpTokenTime.fromEntity $ rt
65         if diffUTCTime now time >= 0
66             then return $ Unauthorized "Who_are_you!"
67             else return Authorized
68     _ -> return $ Unauthorized "Who_are_you!"
69
70     pskAuth :: ( Yesod site
71                 , YesodPersist site
72                 , YesodPersistBackend site ~ SqlBackend
73                 )
74             => HandlerT site IO AuthResult
75     pskAuth = checkTime $ \time -> do
76         pash <- getPash
77         uid' <- lookupPostParam "uid"
78         name' <- lookupPostParam "name"
79         tel'' <- lookupPostParam "tel"
80         let tel' = fmap (read.unpack) tel'' :: Maybe Int
81         case (uid', name', tel') of
82             (Nothing, Nothing, Nothing) -> return $ Unauthorized "Who_are_you!"
83             (Just uid, name, tel) -> do
84                 rt <- liftHandlerT $ runDB $ selectList (
85                     [AccountUid ==. uid] ++ pickF [(AccountName, name)] ++ pickF [(AccountTel,
86                         tel)]) []
87                 checkPash pash rt (runPash 0 time)
88             (Nothing, Just name, tel) -> do
89                 rt <- liftHandlerT $ runDB $ selectList (
90                     [AccountName ==. name] ++ pickF [(AccountTel, tel)]) []
91                 checkPash pash rt (runPash 1 time)
92             (Nothing, Nothing, Just tel) -> do
93                 rt <- liftHandlerT $ runDB $ selectList
94                     [AccountTel ==. tel] []
95                 checkPash pash rt (runPash 2 time)
96         _ -> return $ Unauthorized "Who_are_you!"
97     where

```

```

97     getPash = do
98         pash' <- lookupPostParam "pash"
99         return $ fromMaybe "" pash'
100     checkPash pash rt f = do
101         case rt of
102             item:_ -> do
103                 let usrPash = f.accountPash.fromEntity $ item
104                 if usrPash == pash
105                     then return Authorized
106                     else return $ Unauthorized "Who_are_you!"
107             _ -> return $ Unauthorized "Who_are_you!"
108     checkTime f = do
109         time' <- liftHandlerT $ lookupHeader "TIME-STAMP"
110         now <- liftIO getCurrentTime
111         case time' of
112             Just time -> do
113                 let t = read.unpack.decodeUtf8 $ time
114                 let diff = diffUTCTime now t
115                 if diff <= 12 && diff >= (-12)
116                     then f time
117                     else return $ Unauthorized "I_bought_a_watch_last_year!"
118             _ -> return $ Unauthorized "I_bought_a_watch_last_year!"
119
120     fromEntity :: Entity a -> a
121     fromEntity (Entity _ x) = x

```

10.2.13 src/Dindo/Common/Rable.lhs

提供数据返回的部分部分
返回的类型的通用类型类

```

1 module Dindo.Common.Rable
2   ( RtType(..)
3   , RtWhere(..)

```

```

4   , Variable (..)
5   , defToContent
6   , defToContentXml
7   , defToContentYaml
8   , defToContentJson
9   , Rable (..)
10  , defReturnR
11  , RtStatus (..)
12  , statusHead
13  , RtCommon(..)
14  ) where

```

```

15  import Data.Aeson as A
16  import Data.Yaml as Y
17  import Text.XML as X
18  import Text.Hamlet.XML
19  import Data.ByteString.Internal as BI
20  import Data.ByteString.Lazy as BL (fromStrict, toStrict)
21  import Data.Text as T
22  import Data.Text.Encoding
23  import GHC.Exts(fromList)
24  import Control.Monad
25  import Yesod.Core hiding(toContent)

```

JSON,Yaml,XML

```

26  data RtType = RtJson | RtYaml | RtXml | RtText
27      deriving (Eq,Show)
28  data RtWhere = RtBody | RtOther Text
29      deriving (Eq,Show)

```

```

30  class Show a => Variable a where
31      toValue :: a -> Value
32      toNodes :: a -> [Node]
33      toContents :: RtType -> a -> BI.ByteString

```



```

34     toContents = defToContent
35     defToContent :: Variable a => RtType -> a -> BI.ByteString
36     defToContent RtJson = defToContentJson
37     defToContent RtYaml = defToContentYaml
38     defToContent RtXml = defToContentXml
39     defToContentJson :: Variable a => a -> BI.ByteString
40     defToContentJson = toStrict . A.encode . toValue
41     defToContentYaml :: Variable a => a -> BI.ByteString
42     defToContentYaml = Y.encode . toValue
43     defToContentXml :: Variable a => a -> BI.ByteString
44     defToContentXml x = toStrict $ renderLBS def $ Document p root []
45     where
46         root = Element "data" (fromList []) $ toNodes x
47         p = Prologue [] Nothing []

```

```

48     class Variable a => Rable a where
49         toWhere :: a -> RtWhere
50         toStatus :: a -> RtStatus
51         returnR :: MonadHandler m => a -> m TypedContent
52         returnR = defReturnR
53     defReturnR :: ( MonadHandler m
54                     , Rable a
55                     )
56                 => a -> m TypedContent
57     defReturnR x = do
58         addHeader "Status" $ status x
59         if toWhere x == RtBody
60         then addHeader "Context-Where" "Body"
61         else addHeader "Context-Where" $ \(RtOther a) -> a $ toWhere x
62     addContent
63     where
64         status = statusHead.toStatus
65         addContent = case toWhere x of
66             RtBody -> selectRep $ do

```

```

67         provideRepType "application/json" $ return $ decodeUtf8 $ toContents RtJson
           x
68         provideRepType "application/yaml" $ return $ decodeUtf8 $ toContents RtYaml
           x
69         provideRepType "application/xml" $ return $ decodeUtf8 $ toContents RtXml
           x
70     RtOther y -> do
71         addHeader y $ pack $ show x
72         selectRep $ provideRep $ return ("" :: Text)

```

```

73     data RtStatus = RtSucc | RtFail
74     statusHead :: RtStatus -> Text
75     statusHead RtSucc = "Success"
76     statusHead RtFail = "Failed"

```

将 Yesod 中的 ErrorResponse 实现 Variable 与 Rable

```

77     instance Variable ErrorResponse where
78         toValue NotFound = A.String "NotFound"
79         toValue (InternalError x) = object ["internal-error" .= x]
80         toValue (PermissionDenied x) = object ["permission-denied" .= x]
81         toValue (InvalidArgs x) = object ["invalid-args" .= x]
82         toValue NotAuthenticated = A.String "NotAuthenticated"
83         toValue (BadMethod x) = object ["bad-method" .= show x]
84         toNodes NotFound = [xml|NotFound|]
85         toNodes (InternalError x) = [xml|<InternalError>#{x}|]
86         toNodes (PermissionDenied x) = [xml|<PermissionDenied>:#{x}|]
87         toNodes (InvalidArgs x) = [xml|<InvalidArgs>#{x'}|]
88         where
89             x' = T.unlines x
90         toNodes NotAuthenticated = [xml|NotAuthenticated|]
91         toNodes (BadMethod x) = [xml|<BadMethod>#{pack $ show x}|]
92
93     instance Rable ErrorResponse where
94         toWhere _ = RtBody

```

```
95 | toStatus _ = RtFail
```

通用成功与失败标志

```
96 | data RtCommon = RtCommonSucc
97 |     | RtCommonSuccT Text
98 |     | RtCommonFail Text
99 |     deriving (Eq, Show)
100 | instance Variable RtCommon where
101 |     toValue RtCommonSucc = Null
102 |     toValue (RtCommonSuccT t) = object ["tmp-token" .= t]
103 |     toValue (RtCommonFail x) = String x
104 |     toNodes RtCommonSucc = [xml|null|]
105 |     toNodes (RtCommonSuccT x) = [xml|<tmp-token>#{x}|]
106 |     toNodes (RtCommonFail x) = [xml|<error>#{x}|]
107 | instance Rable RtCommon where
108 |     toWhere RtCommonSucc = RtBody
109 |     toWhere (RtCommonFail _) = RtBody
110 |     toWhere (RtCommonSuccT _) = RtBody
111 |     toStatus RtCommonSucc = RtSucc
112 |     toStatus (RtCommonSuccT _) = RtSucc
113 |     toStatus (RtCommonFail _) = RtFail
```

10.2.14 src/Dindo/Common/Yesod/Config.lhs

提供模块配置的部分

```
1 | module Dindo.Common.Yesod.Config
2 |     ( SvrConfig(..)
3 |     , DbConfig(..)
4 |     , ScError(..)
5 |     , scError
6 |     , dbConfig2Str
7 |     ) where
```

```
8 | import Data.Yaml
```

```
9      import Data.ByteString as B
10     import Data.ByteString.Lazy
11     import Data.String
12     import Control.Exception
```

模块配置与数据库链接配置。

svrPost 后端侦听端口

svrDb 后端的数据库配置（由下面的项组成）

dbAddr 数据库的地址（ip / 域名，不包含端口）

dbPort 数据库侦听的端口

dbUser 链接数据库的用户名

dbName 链接的数据库

dbPsk 链接的密码

ConThd 连接数限制

```
13     data SvrConfig = SvrConfig
14         { svrPort :: Int
15         , svrDb  :: DbConfig
16         }
17     data DbConfig = DbConfig
18         { dbAddr :: String
19         , dbPort :: String
20         , dbUser :: String
21         , dbName :: String
22         , dbPsk  :: String
23         , dbConThd :: Int
24         }
```

将模块配置与数据库连接设置实现 ToJSON 与 FromJSON 类型类，以供数据转换为 JSON 与 YAML。

```

25  instance ToJSON SvrConfig where
26      toJSON SvrConfig{..} = object
27          [ "port" .= svrPort
28            , "database-config" .= svrDb
29          ]
30  instance ToJSON DbConfig where
31      toJSON DbConfig{..} = object
32          [ "addr" .= dbAddr
33            , "port" .= dbPort
34            , "user" .= dbUser
35            , "name" .= dbName
36            , "con-limit" .= dbConThd
37            , "password" .= dbPsk
38          ]
39  instance FromJSON SvrConfig where
40      parseJSON (Object v) = SvrConfig
41          <$> v .: "port"
42          <*> v .: "database-config"
43      parseJSON _ = throw $ ScError "Invailed"
44  instance FromJSON DbConfig where
45      parseJSON (Object v) = DbConfig
46          <$> v .: "addr"
47          <*> v .: "port"
48          <*> v .: "user"
49          <*> v .: "name"
50          <*> v .: "password"
51          <*> v .: "con-limit"
52      parseJSON _ = throw $ ScError "Invailed"

```

将数据库配置转化成链接字符串。

```

53  dbConfig2Str :: DbConfig -> (B.ByteString,Int)
54  dbConfig2Str DbConfig{..} = (str,dbConThd)
55  where

```

```

56     str = toStrict $
57         fromString $    "host=\'" ++ dbAddr
58                        ++ "\_port=\'" ++ dbPort
59                        ++ "\_user=\'" ++ dbUser
60                        ++ "\_password=\'" ++ dbPsk
61                        ++ "\_dbname=\'" ++ dbName
62                        ++ "\'\'

```

设置读写异常

```

63     data ScError = ScError String
64     deriving (Eq)
65     scError = throw.ScError
66     instance Show ScError where
67         show (ScError e) = "parse_server_config_ file FAILED:\n\t" ++ e
68     instance Exception ScError where
69         displayException e = "parse_server_config_ file FAILED:\n\t"

```

JSON 与 Yaml 例程。

```

1  { "port":3000
2  , "database-config":
3    { "addr": "127.0.0.1"
4      , "port": "5432"
5      , "user": "postgres"
6      , "name": "postgres"
7      , "password": "postgres"
8      , "con-limit": 10
9    }
10 }

```

```

1  port: 3000
2  database-config:
3    addr: '127.0.0.1'
4    port: '5432'
5    user: postgres

```

```

6 | name: postgres
7 | password: postgres

```

这个需要在运行时传入。假设配置文件在 config.yml 中, 启动 UsrManage 模块。

```
# cat config.yml | dindo-um
```

10.2.15 src/Dindo/Common/Yesod/Launch.lhs

提供了启动的相关部分

```

1 | module Dindo.Common.Yesod.Launch
2 |   ( Dindoble(..)
3 |   ) where
4 |
5 |   import Dindo.MicroFramework.Register
6 |   import Yesod
7 |   import Dindo.Common.Yesod.Config
8 |   import Database.Persist.Postgresql
9 |   import Control.Monad.Logger

```

Dingo 后端的服务的“标准”

```

9 |   class Registrable a => Dindoble a where
10 |     fromPool :: ConnectionPool -> SvrConfig -> a
11 |     warpDindo :: SvrConfig -> (Int -> a -> IO()) -> IO ()
12 |     warpDindo x warpF =
13 |       runStdoutLoggingT $ withPostgresqlPool connStr cT $
14 |         \pool -> liftIO $ do
15 |           let site = fromPool pool x
16 |               register site
17 |               warpF port site
18 |     where
19 |       (connStr,cT) = dbConfig2Str.svrDb $ x
20 |       port = svrPort x

```

微服务架构这一部分, 就大部分内容犹豫某些原因为实现, 是有能使之运行的空壳。

10.2.16 src/Dindo/MicroFramework/API.lhs

提供了微服务架构中的 API 注册的部分

```
1 module Dindo.MicroFramework.API
2   ( APIble(..)
3     , regAPI
4   ) where
```

```
5 import Yesod.Core
```

注册的 API 的类型类

apis 所公开注册的 API, (API 名称, 相关 Route 信息)

```
6 class ( RenderRoute a
7         ) => APIble a where
8   apis :: a -> [(String,String)]
```

```
9   regAPI :: APIble a => a -> IO Bool
10  regAPI x = do
11    -- 注册 API
12    -- 实际上应该是 数据生成+http 请求, 此处仅输出内容
13    putStrLn "API_内容"
14    print $ apis x
15    return True
```

10.2.17 src/Dindo/MicroFramework/Destory.lhs

提供了微服务架构中销毁的部分

```
1 module Dindo.MicroFramework.Destory
2   ( Destorable(..)
3     , regDestory
4   ) where
```

```
5 import Yesod.Core
```


服务实例销毁的类型类

destoryAPI 销毁的 API

destoryHead 所需的 Head 中特定“签名的内容”

```

6   class ( Yesod a
7       ) => Destorable a where
8       destoryAPI :: a -> String
9       destoryHead :: a -> String

10  regDestory :: Destorable a => a -> IO Bool
11  regDestory x = do
12      -- 注册 销毁接口
13      -- 实际上应该是 http 请求，此处仅输出内容
14      putStrLn "销毁接口注册"
15      print $ destoryAPI x
16      print $ destoryHead x
17      return True

```

10.2.18 src/Dindo/MicroFramework/Register.lhs

提供了微服务架构中服务实例注册的部分

```

1  module Dindo.MicroFramework.Register
2      ( Registrable (..)
3      , Heartbeatable (..)
4      , register
5      ) where

6      import Yesod.Core
7      import Control.Concurrent

8
9      import Dindo.MicroFramework.API
10     import Dindo.MicroFramework.Destroy

```

可注册的服务的类型类。

regSvrAddr 注册目标的地址 ip 或域名

regSvrPost 访问端口

regAddr 注册的服务的地址

regPort 注册的端口

```

11     class ( Yesod a
12             , APIble a
13             , Destorable a
14             , Heartbeatable a
15             ) => Registrable a where
16     regAddr :: a -> String
17     regAddr = defRegAddr
18     regPort :: a -> Int
19     regPort = defRegPort
20     regSvrAddr :: a -> String
21     regSvrPort :: a -> Int
22     defRegPort _ = 3000
23     defRegAddr _ = "localhost"

```

状态获取的类型类

```

24     class ( Yesod a
25             , RenderRoute a
26             ) => Heartbeatable a where
27     heartbeat :: a -> IO ()

```

注册服务实例的函数

False 注册失败

True 注册成功

```

28     register :: Registrable a => a -> IO Bool
29     register x = do
30         -- 注册 服务

```

```

31      -- 实际上应该是 http 请求，此处仅输出内容
32      putStrLn "注册服务的端口"
33      print $ regSvrPort x
34      putStrLn "注册服务的地址"
35      print $ regSvrAddr x
36      putStrLn "被注册的实例的地址"
37      print $ regPort x
38      putStrLn "被注册的实例的端口"
39      print $ regPort x
40      regAPI' $ regDestory' $ do
41          forkIO $ heartbeat x
42          return True
43      where
44          regAPI' a = do
45              ra <- regAPI x
46              if ra then a else return False
47          regDestory' a = do
48              rd <- regDestory x
49              if rd then a else return True

```

10.3 dindo-launch

这一部分是 dindo 的服务的启动部分。

10.3.1 src/Main.lhs

启动器的主体

```

1  module Main
2      ( main
3      ) where
4
5      import qualified GHC.IO.Encoding as E
6      import System.IO
7      import Dindo.Std

```

```

7  import System.Console.CmdArgs
8  import Dindo.Import.Aeson as A
9  import Dindo.Import.Yaml as Y
10 import Dindo.Import.Yesod
11 import Data.Maybe
12 import qualified Dindo.Import.ByteString as B
13 import qualified Dindo.Import.Text as T
14 import Dindo.Common.Yesod.Launch
15 import Dindo.Common.Yesod.Config
16 import Paths_dindo_launch
17 import Data.Version
18 import Dindo.Common(dindo_common_version_quasi)
19 import Dindo.Import.Database(dindo_database_version_quasi)
20 import Control.Exception(try, SomeException, ErrorCall(..), throw, evaluate)
21 import Data.Char

```

启动方式是通过标准输入流输入，输入的格式是 JSON 或者是 YAML，“-form=” 这个选项是控制输入或输出的是的，是 JSON 或者是 YAML。

```

22 data Launch = Launch {form :: String}
23   deriving (Show, Data, Typeable)
24 launch = Launch{form="auto" &= typ "AUTO|YAML|JSON" &= help "格式"}
25   &= summary ( "dindo-common-"
26               ++ $(dindo_common_version_quasi)
27               ++ ";_dindo-database-"
28               ++ $(dindo_database_version_quasi)
29               ++ ";_" ++ $(dindo_module_name) ++ "-"
30               ++ $(dindo_module_version)
31               ++ ";_dindo-launch-"
32               ++ showVersion version)

```

```

33 main :: IO ()
34 main = do
35 #ifndef WithoutUTF8
36     E.setLocaleEncoding E.utf8

```

```

37     hSetEncoding stdout utf8
38 #endif
39     cfg' <- cmdArgs launch >>= cfg
40     warpDindo cfg' itemWarp
41     where
42         itemWarp :: Int -> $(std) -> IO()
43         itemWarp = warp
44     cfg :: Launch -> IO SvrConfig
45     cfg l = getContents >>= (decode'.T.encodeUtf8.T.pack)
46     where
47         tryList :: [a -> SvrConfig] -> [ScError] -> a -> IO SvrConfig
48         tryList [] es a = scError.concatWith "\n\t".map getError $ es
49         tryList (x:xs) es a = do
50             rt <- try.evaluate $ x a :: IO (Either ScError SvrConfig)
51             case rt of
52                 Left e -> tryList xs (e:es) a
53                 Right sc -> return sc
54         getError (ScError a) = a
55         concatWith a xs = foldr sig "all _ failed " xs
56         where
57             sig x os = x ++ a ++ os
58         decJ = fromMaybe (throw $ ScError "Invailed_JSON").A.decode.B.fromStrictBS
59         decY = fromMaybe (throw $ ScError "Invailed_YAML").Y.decode
60         decA = tryList [decY,decJ] []
61         decode' = let Launch ll = l in
62             case map toLower ll of
63                 "auto" -> decA
64                 "json" -> evaluate.decJ
65                 "yaml" -> evaluate.decY
66                 _ -> error "error_form"

```

10.4 dindo-usrmanage

这一部分是 dindo 的用户管理了部分。

10.4.1 src/Dindo/Std.lhs

与 Dindo 启动器对接的部分

```
1 module Dindo.Std
2   ( module X
3     , std
4     , dindo_module_name
5     , dindo_module_version
6   ) where
7
8   import Dindo.UM as X -- need change
9   import Dindo.Import.TH
10
11   dindo_module_name = stringE "dindo-usrmanage"
12   dindo_module_version = dindo_usrmanage_version_quasi
13   std = [t|UM|]
```

10.4.2 src/Dindo/UM.lhs

用户管理部分的导出的部分

```
1 module Dindo.UM
2   ( module X
3     , dindo_usrmanage_version
4     , dindo_usrmanage_version_quasi
5   ) where
6   import Dindo.UM.Foundation as X
7   import Dindo.UM.Handler as X
8   import Dindo.Import.Yesod
9   import Dindo.Import.TH
10  import Data.Version
```

```

11 import Paths_dindo_usrmanage
12
13 dindo_usrmanage_version = version
14 dindo_usrmanage_version_quasi = stringE $ showVersion version
15 mkYesodDispatch "UM" resourcesUM

```

10.4.3 src/Dindo/UM/Data.lhs

定义返回数据的部分

```

1 module Dindo.UM.Data
2   ( RtRegist(..)
3   , RtIdy(..)
4   , RtIdfed(..)
5   , RtUImg(..)
6   , RtUInfo(..)
7   , RtChPsk(..)
8   , RtEaddr(..)
9   , RtGEadd(..)
10  ) where
11
12 import Dindo.Import.Rable
13 import Dindo.Import.Aeson as A
14 import Dindo.Import.Yaml as Y
15 import Dindo.Import.Text as T
16 import Dindo.Import.ByteString as B
17 import Dindo.Import.Yesod
18 import Dindo.Import.Database

```

用户注册返回数据

```

19 data RtRegist = RtRegist
20   { uid :: Text
21   }
22 | RtRegistFail
23   { regReason :: Text

```

```

24     }
25     deriving (Eq)
26 instance Show RtRegist where
27     show (RtRegist x) = T.unpack x
28     show (RtRegistFail x) = T.unpack x
29 instance Variable RtRegist where
30     toValue (RtRegist x) = object ["uid" .= x]
31     toValue (RtRegistFail x) = object ["error" .= x]
32     toNodes (RtRegist x) = [xml|<uid>#{x}|]
33     toNodes (RtRegistFail x) = [xml|<error>#{x}|]
34 instance Rable RtRegist where
35     toWhere (RtRegist _) = RtBody
36     toWhere (RtRegistFail _) = RtBody
37     toStatus (RtRegist _) = RtSucc
38     toStatus (RtRegistFail _) = RtFail

```

用户认证信息的返回数据

```

39 data Rtldy = Rtldy
40     | RtldyFail
41     { idyReason :: Text
42     }
43     deriving (Eq)
44 instance Show Rtldy where
45     show (RtldyFail x) = T.unpack x
46 instance Variable Rtldy where
47     toValue Rtldy = Null
48     toValue (RtldyFail x) = object ["error" .= x]
49     toNodes Rtldy = [xml|null|]
50     toNodes (RtldyFail x) = [xml|<error>#{x}|]
51 instance Rable Rtldy where
52     toWhere (RtldyFail _) = RtBody
53     toWhere Rtldy = RtBody
54     toStatus Rtldy = RtSucc
55     toStatus (RtldyFail _) = RtFail

```


用户查询认证状态信息

```

56  data RtIdfed = RtIdfedPass | RtIdfedNo
57      deriving (Eq, Show)
58  instance Variable RtIdfed where
59      toValue RtIdfedPass = object ["status" .= ("pass" :: Text)]
60      toValue RtIdfedNo   = object ["status" .= ("no" :: Text)]
61      toNodes RtIdfedPass = [xml|<status>pass|]
62      toNodes RtIdfedNo   = [xml|<status>no|]
63  instance Rable RtIdfed where
64      toWhere RtIdfedPass = RtBody
65      toWhere RtIdfedNo   = RtBody
66      toStatus RtIdfedPass = RtSucc
67      toStatus RtIdfedNo   = RtSucc

```

用户信息查询返回结果

```

68  data RtUInfo = RtUInfo
69      { rtuiUid :: Text
70      , rtuiName :: Text
71      , rtuiTel :: Text
72      , rtuiEmail :: Text
73      }
74      | RtUInfoNSU
75  instance Show RtUInfo where
76      show RtUInfoNSU = "no_such_a_user"
77  instance Variable RtUInfo where
78      toValue RtUInfo{..} = object
79          [ "uid" .= rtuiUid
80          , "name" .= rtuiName
81          , "tel" .= rtuiTel
82          , "email" .= rtuiEmail
83          ]
84      toNodes RtUInfo{..} = [xml|
85          <uid> #{rtuiUid}
86          <name> #{rtuiName}

```

```

87     <tel> #{rtuiTel}
88     <email> #{rtuiEmail}
89   ]
90   instance Rable RtUInfo where
91     toWhere RtUInfo{..} = RtBody
92     toWhere RtUInfoNSU = RtOther "CONTEXT"
93     toStatus RtUInfo{..} = RtSucc
94     toStatus RtUInfoNSU = RtFail

```

获取用户头像返回内容

```

95   data RtUImg = RtUImg ByteString
96               | RtUImgFail
97   deriving (Eq)
98   instance Show RtUImg
99   instance Variable RtUImg
100  instance Rable RtUImg where
101    returnR (RtUImg img) =
102      selectRep $ provideRepType "image/png" $ return img
103    returnR RtUImgFail = do
104      addHeader "CONTEXT-WHERE" "CONTEXT"
105      addHeader "CONTEXT" "Failed_on_get_image"
106      selectRep $ provideRep $ return ("": Text)

```

更改密码的返回值

```

108  data RtChPsk = RtChPsk
109              | RtChPskFail Text
110  deriving (Eq)
111  instance Show RtChPsk where
112    show (RtChPskFail x) = T.unpack x
113  instance Variable RtChPsk where
114    toValue RtChPsk = Null
115    toValue (RtChPskFail x) = object ["error" .= x]
116    toNodes RtChPsk = [xml|null|]
117    toNodes (RtChPskFail x) = [xml|<error>#{x}|]

```

```

118 instance Rable RtChPsk where
119     toWhere RtChPsk = RtBody
120     toWhere (RtChPskFail _) = RtBody
121     toStatus RtChPsk = RtSucc
122     toStatus (RtChPskFail _) = RtFail

```

收货地址的增删的返回值

```

123 data RtEaddr = RtEaddrAdd Text
124             | RtEaddrChn
125             | RtEaddrDel
126             | RtEaddrFail Text
127 deriving (Eq, Show)
128 instance Variable RtEaddr where
129     toValue (RtEaddrAdd x) = object ["aid" .= x]
130     toValue RtEaddrChn = Null
131     toValue RtEaddrDel = Null
132     toValue (RtEaddrFail x) = object ["error" .= x]
133     toNodes (RtEaddrAdd x) = [xml|<aid>#{x}|]
134     toNodes RtEaddrChn = [xml|null|]
135     toNodes RtEaddrDel = [xml|null|]
136     toNodes (RtEaddrFail x) = [xml|<error>#{x}|]
137 instance Rable RtEaddr where
138     toWhere (RtEaddrAdd _) = RtBody
139     toWhere RtEaddrChn = RtBody
140     toWhere RtEaddrDel = RtBody
141     toWhere (RtEaddrFail _) = RtBody
142     toStatus (RtEaddrAdd _) = RtSucc
143     toStatus RtEaddrChn = RtSucc
144     toStatus RtEaddrDel = RtSucc
145     toStatus (RtEaddrFail _) = RtFail

```

获取地址

```

146 data RtGEadd = RtGEadd [Addr]
147             | RtGEaddFail Text

```

```

148     deriving (Eq, Show)
149 instance Variable RtGEadd where
150     toValue (RtGEadd x) = toJSON x
151     toValue (RtGEaddFail x) = object ["error" .= x]
152     toNodes (RtGEadd xs) = [xml|
153         $forall x <- xs
154             <aid>#{addrAid x}
155             <addr>#{addrAddr x}
156             <zip>#{addrZip x}
157         |]
158     toNodes (RtGEaddFail x) = [xml|<error>#{x}|]
159 instance Rable RtGEadd where
160     toWhere (RtGEadd _) = RtBody
161     toWhere (RtGEaddFail _) = RtBody
162     toStatus (RtGEadd _) = RtSucc
163     toStatus (RtGEaddFail _) = RtFail

```

10.4.4 src/Dindo/UM/Foundation.lhs

基础的部分

```

1 module Dindo.UM.Foundation where
2
3     import Dindo.Common
4     import Dindo.Import
5     import Dindo.Import.Yesod
6     import Dindo.Import.Database
7     import Paths_dindo_usrmanage
8     import Dindo.Import.Text as T
9     import Data.Version

```

定义基本类型路由表

```

10 data UM = UM
11     { connPool :: ConnectionPool
12     , config    :: SvrConfig

```

```

13     }
14     mkYesodData "UM" [parseRoutes|
15         /regist  RegistR POST
16         /identify IdentifyR POST
17         /identified Identified POST
18         /login  LoginR POST
19         /logout LogoutR POST
20         /usrinfo  UsrinfoR POST
21         /usrhimg  UshrimgR POST
22         /usrinfochange  UsrinfochangeR POST
23         /changpash ChangpashR POST
24         /upeaddr  UpeaddrR POST
25         /geteaddr  GeteaddrR POST
26     |]

```

实现 Yesod 类型类

```

27     instance Yesod UM where
28         errorHandler = returnR
29         isAuthorized ShomeR _ = return Authorized
30         isAuthorized RegistR _ = noAuth
31         isAuthorized LoginR _ = pskAuth
32         isAuthorized _ _ = tokenAuth
33     instance YesodPersist UM where
34         type YesodPersistBackend UM = SqlBackend
35         runDB a = getYesod >>= (runSqlPool a.connPool)
36         mkShomeR $ pack $ "dindo-um-" ++ showVersion version ++ ";_dindo-common-"
            ++ $(dindo_common_version_quasi)

```

微服务架构

```

37     instance APIble UM where
38         apis _ = []
39     instance Destorable UM where
40         destoryHead _ = ""
41         destoryAPI _ = ""

```

```
42 instance Heartbeatable UM where
43     heartbeat _ = return ()
44 instance Registrable UM where
45     regAddr _ = ""
46     regPort = svrPort . config
47     regSvrPort _ = 80
48     regSvrAddr _ = ""
49 instance Dindoble UM where
50     fromPool = UM
```

10.4.5 src/Dindo/UM/Handler.lhs

处理函数的部分

```
1 module Dindo.UM.Handler
2   ( postRegistR
3   , postUsrinfoR
4   , postLogoutR
5   , postLoginR
6   , postIdentified
7   , postIdentifyR
8   , postUsrinfochangeR
9   , postChangpashR
10  , postUsrhimgR
11  , postUpeaddrR
12  , postGeteaddrR
13  ) where
```

```
14 import Dindo.Import
15 import Dindo.Import.Rable
16 import Dindo.Import.Yesod
17 import Dindo.Import.Database
18 import Dindo.UM.Foundation
19 import Dindo.UM.Data
20 import Dindo.Import.Digest
```

```

21 import Dindo.Import.ByteString as B hiding(unpack,pack,splitAt, take,map,null)
22 import Dindo.Import.Text as T hiding(splitAt, take,map,null)
23 import Dindo.Common.Auth(fromEntity,pickU,pickF)
24 import Control.Exception(try, SomeException)
25 import Control.Monad

```

注册的 API

```

26 postRegistR :: Handler TypedContent
27 postRegistR =
28   getParam insertAltem
29   where
30     getParam f = do
31       name' <- lookupPostParam "name"
32       pash' <- lookupPostParam "pash"
33       tel' <- lookupPostParam "tel"
34       case (name',pash', tel') of
35         (Just name,Just pash,Just tel) -> do
36           x <- liftIO getCurrentTime
37           let (time,p) = splitAt 10 $ show x
38           let to = showDigest $ sha1 $ fromStrictBS $ encodeUtf8 $ T.concat [pash,
39             name]
40           let uid = 'U':time ++ to
41           f (pack uid,name,pash,read (unpack tel))
42           _ -> returnR $ RtRegistFail "param:␣less␣and␣less"
43       insertAltem (uid,name,pash,tel) = do
44         rt <- liftHandlerT $ tryRunDB $
45           insert $ Account uid pash tel name
46         returnR $ case rt of
47           Left e -> RtRegistFail $ pack $ show e
48           Right _ -> RtRegist uid

```

用户认证的 API

```

48 postIdentifyR :: Handler TypedContent
49 postIdentifyR =

```

```

50     checkParam $ addItem $ checkPic addPic
51     where
52         checkParam f = do
53             email' <- lookupPostParam "email"
54             rname' <- lookupPostParam "rname"
55             prcid' <- lookupPostParam "prcid"
56             addr' <- lookupPostParam "addr"
57             case (email', rname', prcid', addr') of
58                 (Just email, Just rname, Just prcid, Just addr) ->
59                     f (email, rname, prcid, addr)
60                 _ -> returnR $ RtldyFail "param:␣less␣and␣less"
61     checkPic f ins = do
62         pic' <- lookupFile "pic"
63         case pic' of
64             Just pic -> do
65                 rt <- sourceToList $ fileSource pic
66                 let bpic = B.concat rt
67                 f (bpic, ins)
68                 _ -> returnR $ RtldyFail "param:␣picture␣needed"
69     addItem f (email, rname, prcid, addr) =
70         f $ \uid -> Usr uid email rname prcid addr "N"
71     addPic (pic, usr) = do
72         uid <- getUId
73         now <- liftIO getCurrentTime
74         let str = show now
75         let (time, p) = splitAt 10 $ str
76         let to = showDigest $ sha1 $ fromStrictBS $ encodeUtf8 $ T.concat [uid, pack
77             str]
78         let pid = pack $ 'A':time ++ to
79         rt <- liftHandlerT $ tryRunDB $ do
80             insert $ usr uid
81             insert $ Apic pid uid pic $ Just 0
82         returnR $ case rt of
83             Left e -> RtldyFail $ pack $ show e

```



```
83 | Right _ -> Rtldy
```

认证状态查询

```
84 | postIdentified :: Handler TypedContent
85 | postIdentified = do
86 |   uid <- getUid
87 |   rt <- liftHandlerT $ runDB $ selectList [UsrUid ==. uid] []
88 |   returnR $ case rt of
89 |     (Entity _ item):_ -> if usrStatus item == "P"
90 |       then RtldfedPass
91 |       else RtldfedNo
92 |   _ -> RtldfedNo
```

用户登录

```
93 | postLoginR :: Handler TypedContent
94 | postLoginR = do
95 |   uid' <- lookupPostParam "uid"
96 |   name' <- lookupPostParam "name"
97 |   tel' <- lookupPostParam "tel"
98 |   case (uid', name', tel') of
99 |     (uid, name, tel) -> do
100 |       pash <- getPash
101 |       rt' <- liftHandlerT $ runDB $ selectList (pickF
102 |         [ (AccountUid, uid)
103 |           , (AccountName, name)
104 |         ] ++ pickF
105 |         [ (AccountTel, fmap (read.unpack) tel)
106 |         ]) []
107 |       case rt' of
108 |         (Entity _ item):_ -> do
109 |           let uid = accountUid item
110 |           now <- liftIO getCurrentTime
111 |           let lim = addUTCTime 3600 now
112 |           let time = show lim
```

```

113         let to = showDigest $ sha512 $ fromStrictBS $ encodeUtf8 $ T.concat [uid,
           pash,pack time]
114         let tt = pack $ take 22 time ++ to
115         liftHandlerT $ runDB $ insert $ TmpToken tt lim uid
116         returnR $ RtCommonSuccT tt
117     where
118         getPash = do
119             pash' <- lookupPostParam "pash"
120             return $ fromMaybe "" pash'

```

用户登出

```

121     postLogoutR :: Handler TypedContent
122     postLogoutR = do
123         Just token <- lookupHeader "TMP-TOKEN"
124         Just uid <- lookupHeader "USR-ID"
125         rt <- liftHandlerT $ tryRunDB $ deleteWhere [TmpTokenTt ==. decodeUtf8 token,
           TmpTokenUid ==. decodeUtf8 uid]
126         returnR $ case rt of
127             Left e -> RtCommonFail $ pack $ show e
128             Right _ -> RtCommonSucc

```

查询用户信息

```

129     postUsrinfoR :: Handler TypedContent
130     postUsrinfoR = do
131         tuid <- getUid
132         uid' <- lookupPostParam "uid"
133         let uid = fromMaybe tuid uid'
134         rt' <- liftHandlerT $ runDB $ selectList [UsrUid ==. uid] []
135         case rt' of
136             Entity _ rt:_ -> do
137                 let email = usrEmail rt
138                 Entity _ item:_ <- liftHandlerT $ runDB $ selectList [AccountUid ==. uid] []
139                 returnR $ RtUInfo uid (accountName item) (pack $ show $ accountTel item)
           email

```

```
140 | _ -> returnR RtUInfoNSU
```

获得用户头像

```
141 | postUshrimgR :: Handler TypedContent
142 | postUshrimgR = do
143 |   tuid <- getUId
144 |   uid' <- lookupPostParam "uid"
145 |   let uid = fromMaybe tuid uid'
146 |   rt' <- liftHandlerT $ runDB $ selectList [ApicUId ==. uid] []
147 |   case rt' of
148 |     Entity _ rt:_ -> returnR $ RtUImg $ apicBpic rt
149 |     _ -> returnR $ RtUImgFail
```

用户信息变更

```
150 | postUshrinfochangeR :: Handler TypedContent
151 | postUshrinfochangeR = check update
152 | where
153 |   updatePic uid pic' = case pic' of
154 |     Nothing -> return ()
155 |     Just pic -> do
156 |       rt <- sourceToList $ fileSource pic
157 |       let bpic = B.concat rt
158 |       updateWhere [ApicUId ==. uid, ApicTyp ==. Just 0] [ApicBpic =. bpic]
159 |   update (a,b,pic) = do
160 |     uid <- getUId
161 |     rt <- liftHandlerT $ tryRunDB $ do
162 |       when (not $ null a) $
163 |         updateWhere [AccountUId ==. uid] a
164 |       when (not $ null b) $
165 |         updateWhere [UshrUId ==. uid] b
166 |     updatePic uid pic
167 |     returnR $ case rt of
168 |       Left e -> RtCommonFail $ pack $ show e
169 |       Right _ -> RtCommonSucc
```

```

170     check f = do
171         name <- liftHandlerT $ lookupPostParam "name"
172         tel  <- liftHandlerT $ lookupPostParam "tel"
173         email <- liftHandlerT $ lookupPostParam "email"
174         rname <- liftHandlerT $ lookupPostParam "rname"
175         prcid <- liftHandlerT $ lookupPostParam "prcid"
176         addr <- liftHandlerT $ lookupPostParam "addr"
177         pic <- liftHandlerT $ lookupFile "pic"
178         let a = pickU [(AccountName,name)]
179         let a' = pickU [(AccountTel,fmap (read.T.unpack) tel)]
180         let b = pickU [(UsrEmail,email),(UsrRname,rname),(UsrPrcid,prcid),(UsrAddr,
181             addr)]
182         f (a++a',b,pic)

```

修改密码

```

182     postChangpashR :: Handler TypedContent
183     postChangpashR = check changePash
184     where
185         changePash pash = do
186             uid <- getUid
187             rt <- liftHandlerT $ tryRunDB $ updateWhere [AccountUid ==. uid] [
188                 AccountPash ==. pash]
189             returnR $ case rt of
190                 Left e -> RtChPskFail $ pack $ show e
191                 Right _ -> RtChPsk
192         check f = do
193             pash' <- lookupPostParam "pash"
194             case pash' of
195                 Nothing -> do
196                     returnR $ RtChPskFail "param:␣less␣and␣less"
197                 Just x -> f x

```

收获地址

```

197     postUpeaddrR :: Handler TypedContent

```

```

198 postUpeaddrR = spl
199   where
200     changeltem aid a = do
201       rt <- liftHandlerT $ tryRunDB $ updateWhere [AddrAid ==. aid] a
202       returnR $ case rt of
203         Left e -> RtEaddrFail $ pack $ show e
204         Right _ -> RtEaddrChn
205     checkChn f = do
206       addr <- liftHandlerT $ lookupPostParam "addr"
207       zipcode <- liftHandlerT $ lookupPostParam "zip"
208       aid' <- liftHandlerT $ lookupPostParam "aid"
209       case aid' of
210         Just aid -> f aid $ pickU [(AddrAddr,addr),(AddrZip,zipcode)]
211         Nothing -> returnR $ RtEaddrFail "param:change:␣less␣and␣less"
212     delltem aid = do
213       rt <- liftHandlerT $ tryRunDB $ deleteWhere [AddrAid ==. aid]
214       returnR $ case rt of
215         Left e -> RtEaddrFail $ pack $ show e
216         Right _ -> RtEaddrDel
217     checkDel f = do
218       aid' <- liftHandlerT $ lookupPostParam "aid"
219       case aid' of
220         Just aid -> f aid
221         Nothing -> returnR $ RtEaddrFail "param:del:␣less␣and␣less"
222     addltem (addr,zipcode) = do
223       uid <- getUId
224       now <- liftIO getCurrentTime
225       let aid' = showDigest $ sha256 $ fromStrictBS $ encodeUtf8 addr
226       let aid = pack $ "A" ++ show now ++ aid'
227       rt <- liftHandlerT $ tryRunDB $ insert $ Addr aid uid zipcode addr
228       returnR $ case rt of
229         Left e -> RtEaddrFail $ pack $ show e
230         Right _ -> RtEaddrAdd aid
231     checkAdd f = do

```

```

232     addr' <- liftHandlerT $ lookupPostParam "addr"
233     zip' <- liftHandlerT $ lookupPostParam "zip"
234     case (addr', zip') of
235       (Just addr, Just zipcode) -> f (addr, zipcode)
236       _ -> returnR $ RtEaddrFail "param:addr:_less_and_less"
237     spl = do
238       opt <- liftHandlerT $ lookupHeader "OPT"
239       case opt of
240         Just "ADD" -> checkAdd addItem
241         Just "DEL" -> checkChn changeltem
242         Just "CHANGE" -> checkDel delltem
243         _ -> returnR $ RtEaddrFail "header:opt:_less_and_less"

```

获取收货地址

```

244     postGeteaddrR :: Handler TypedContent
245     postGeteaddrR = spl
246     where
247       getByUid uid = do
248         rt <- liftHandlerT $ runDB $ selectList [AddrUid ==. uid] []
249         returnR $ RtGEadd $ map fromEntity rt
250       getByAid aid = do
251         uid <- getUid
252         rt <- liftHandlerT $ runDB $ selectList [AddrAid ==. aid, AddrUid ==. uid] []
253         returnR $ RtGEadd $ map fromEntity rt
254     spl = do
255       uid' <- liftHandlerT $ lookupPostParam "uid"
256       aid' <- liftHandlerT $ lookupPostParam "aid"
257       case (uid', aid') of
258         (Just uid, _) -> getByUid uid
259         (Nothing, Just aid) -> getByAid aid
260         _ -> returnR $ RtGEaddFail "param:_less_and_less"

```

10.5 dindo-tools

dindo 的辅助工具

dindo-pash 测试用的辅助工具

10.5.1 src/pash/Main.lhs

主函数部分

产生密钥的工具

```
1 module Main
2   ( main
3   ) where

4   import qualified GHC.IO.Encoding as E
5   import System.IO
6   import System.Environment
7   import Dindo.Import
8   import Dindo.Common.Auth
9   import Dindo.Import.Digest
10  import qualified Dindo.Import.Text as T
11  import qualified Dindo.Import.ByteString as B
12  import Dindo.Common(dindo_common_version_quasi)
13  import Data.Version
14  import System.Console.CmdArgs
15  import Paths_dindo_tools

16  main :: IO ()
17  main = do
18    #ifndef WithoutUTF8
19      E.setLocaleEncoding E.utf8
20      hSetEncoding stdout utf8
21    #endif
22    Pash key t at <- cmdArgs pash
23    now' <- getCurrentTime
24    let now = addUTCTime (fromIntegral at) now'
```

```

25     pash <- getPash t key now
26     a' <- getContents
27     let a = concat.lines $ a'
28     case t of
29       100 -> putStr $ a ++ "\u-d\u\"pash="++pash++"\n"
30       _ -> putStr $ a ++ "\u-d\u\"pash="++pash++"\u-H\u\"TIME-STAMP:"++
           show now++"\n"
31     return ()
32     where
33       getPash typ key now = case typ of
34         100 -> return $ showDigest $ sha256 $ B.fromStrictBS $ T.encodeUtf8 $ T.pack
           key
35         x -> do
36           let k = T.pack $ showDigest $ sha256 $ B.fromStrictBS $ T.encodeUtf8 $ T.
           pack key
37           let time = T.encodeUtf8.T.pack.show $ now
38           return $ T.unpack $ runPash x time k

```

dindo-pash 使用说明 一共有两个参数：一个是密码，另一个是散列方式，也就是认证方式。

100 注册时

0 使用 uid 登录时

1 使用 name 登录时

2 使用 tel 登录时

有一个 flag 开关是关于时间矫正的，矫正单位是秒。

```

39     data Pash = Pash {pKey :: String,pType :: Int,aTime :: Int}
40     deriving (Show,Data,Typeable)
41     pash = Pash
42       { pKey = def &= argPos 1 &= typ "PASSWORD"
43       , pType = def &= argPos 2 &= typ "IDENTIFY-TYPE"
44       , aTime = 0 &= typ "UTCDiffTime" &= help "时间矫正"
45       } &= summary ( "dindo-common:-"

```



```
46         ++ $(dindo_common_version_quasi)
47         ++ ";_dindo-tools-"
48         ++ showVersion version
49     )
```

11 Dindo 公共组件

这部分是关于 Dindo 的公共组件的。由于 Dingo 后端采用的微服务架构¹²，不同的微服务之间，会有包括服务发现¹³、数据库¹⁴、授权认证等是共用的。所以为了减少代码的重复使用，则独立出这一部分。

12 Dindo 数据库

13 Dindo Launcher

14 Dindo 微服务组件——用户管理

15 DIndo 测试说明

15.1 如何测试

¹²后面随时可能会称之为微架构。

¹³目前的版本并没有开发实际的服务发现的内容，直接使用 Nginx 进行做均衡负载等。

¹⁴这一部分单独出来的。

A 术语解释

CaaS Container as a Server，是指将容器（Docker）提供作为一种服务。是云计算中的概念，与 PaaS、SaaS 等概念对等。

B Docker 中 Weave 的配置

Weave 是能将 Docker 中每个物理主机中的连接起来一个工具，也就是能使用的 Docker 容器跨主机互联。下面是配置（安装）Weave 的 Shell 脚本：

Listing 1: Weave 安装

```
1 #!/bin/sh
2 wget -O /usr/local/bin/weave \
3 https://github.com/zettio/weave/releases/download/latest_release/weave
4 chmod a+x /usr/local/bin/weave
5 dao pull weaveworks/weave:1.5.1
6 dao pull weaveworks/plugin:1.5.1
7 dao pull weaveworks/weaveexec:1.5.1
8 apt-get update
9 apt-get install bridge-utils
10 dao pull weaveworks/weavedb:latest
11 weave launch 192.168.1.181
```

运行容器需要使用

```
# weave run <ip> <repo>
```

C 后端附带工具使用说明

C.1 dindo-pash

dindo-pash 是用于测试期间生成密码的工具，具体使用请参照 ?? 部分。dindo-pash 直接输出的是对应着 cURL 的参数名称。同时输入的内容应该是 cURL 对应的其他内容。

```
$ echo 'curl --some-flags url://host' | dindo-pash password
```

D 发行（发布）的二进制文件镜像与包的命名规则

这一部分的内容是关于发布或发行的二进制文件包或者 Docker 镜像的命名规则。(构建类型 __ 构建编号)-([commit hash] | [tag name])-(操作系统体系 __ 发行版本)-(编译系统体系 __ 版本)-(cpu 架构体系)-[llvm__ 版本]-[threaded]-[其他特性]-(模块) 例如某二进制包的文件名：
single-7a8c900-win32_windows_10_rs1_14342-x86_64-GHC_8.0.1-llvm_3.8-threaded-all_in_one.tar.xz

参考文献

- [1] 灵雀云收费标准 2016 年 5 月, [Alauda-Price](#)