



Dingo

Dindo Document

DingoLab
June, 2016

Dingo Dindo Document

李约瀚

qinka@live.com

14130140331

June 6th 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	12
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	14
10.2.13	src/Dindo/Common/Rable.lhs	18
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	33
10.4.3	src/Dindo/UM/Data.lhs	33
10.4.4	src/Dindo/UM/Foundation.lhs	39
10.4.5	src/Dindo/UM/Handler.lhs	41
10.5	dindo-tools	50
10.5.1	src/pash/Main.lhs	50
11	Dindo 公共组件	52
12	Dindo 数据库	52
13	Dindo Launcher	52
14	Dindo 微服务组件——用户管理	52
15	DIndo 测试说明	52
15.1	如何测试	52
A	术语解释	53
B	Docker 中 Weave 的配置	53
C	后端附带工具使用说明	53
C.1	dindo-pash	53

目录	iii
D 发行（发布）的二进制文件镜像与包的命名规则	54

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=varcher(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.Destory 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   , readT
5   ) where
6
7   import Data.Text as X
8   import Data.Text.Encoding as X
```

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

```
11  readT :: Read a => Text -> a
12  readT = read.unpack
```

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
7   import Yesod as X hiding (mkYesodData)
8   import qualified Yesod (mkYesodData)
9   import Dindo.Common.Rable as X
10  import Dindo.Common.Auth as X
11  import Dindo.Common.Yesod.Launch as X
12  import Dindo.Common.Yesod.Config as X
13  import Dindo.Import.TH
14  import Data.Maybe
15  import Data.Time
16  import Data.Text
17  import qualified Data.Text.Encoding as TE
18  import Data.Aeson
19  import Data.ByteString.Lazy as BL hiding(unpack)
20
21  mkYesodData a b = Yesod.mkYesodData a b'
22    where
23      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   getUserId = do
34     tt' <- lookupHeader "TMP-TOKEN"
35     uid' <- lookupHeader "UID"
36     let Just tt = fmap decodeUtf8 tt'
37     let Just uid = fmap decodeUtf8 uid'
38     rt' :: _ <- liftHandlerT $ runDB $ selectList [TmpTokenTt ==. tt, TmpTokenUid ==.
        uid] []
39     let rt = fromEntity rt'
40     return $ tmpTokenUid rt

```

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

```

41   runPash :: Int -> B.ByteString -> Text -> Text
42   runPash i time pash = pack $ showDigest $ sha512 $ B.fromStrict $ B.concat [pre,
        encodeUtf8 pash,time]
43   where
44     pre = case i of
45       0 -> "uid"
46       1 -> "nnnn"
47       2 -> "+86"
48   runPash _ _ x = id x
49   noAuth :: Yesod site => HandlerT site IO AuthResult
50   noAuth = return Authorized
51
52   tokenAuth :: ( Yesod site
53     , YesodPersist site
54     , YesodPersistBackend site ~ SqlBackend
55     )
56     => HandlerT site IO AuthResult
57   tokenAuth = do
58     token' <- lookupHeader "TMP-TOKEN"
59     case token' of

```

```

60     Nothing -> return $ Unauthorized "Who_are_you!"
61     Just token -> do
62         rt' <- liftHandlerT $ runDB $ selectList [TmpTokenTt ==. decodeUtf8 token][
            Desc TmpTokenTime]
63         case rt' of
64             rt:_ -> do
65                 now <- liftIO getCurrentTime
66                 let time = tmpTokenTime.fromEntity $ rt
67                 if diffUTCTime now time >= 0
68                     then return $ Unauthorized "Who_are_you!"
69                     else return Authorized
70             _ -> return $ Unauthorized "Who_are_you!"
71
72     pskAuth :: ( Yesod site
73                 , YesodPersist site
74                 , YesodPersistBackend site ~ SqlBackend
75                 )
76     => HandlerT site IO AuthResult
77     pskAuth = checkTime $ \time -> do
78         pash <- getPash
79         uid' <- lookupPostParam "uid"
80         name' <- lookupPostParam "name"
81         tel'' <- lookupPostParam "tel"
82         let tel' = fmap (read.unpack) tel'' :: Maybe Int
83         case (uid', name', tel') of
84             (Nothing,Nothing,Nothing) -> return $ Unauthorized "Who_are_you!"
85             (Just uid, name, tel) -> do
86                 rt <- liftHandlerT $ runDB $ selectList (
87                     [AccountUid ==. uid] ++ pickF [(AccountName, name)] ++ pickF [(AccountTel,
88                         tel)]) []
89                 checkPash pash rt (runPash 0 time)
90             (Nothing,Just name, tel) -> do
91                 rt <- liftHandlerT $ runDB $ selectList (
                    [AccountName ==. name] ++ pickF [(AccountTel, tel)]) []

```

```

92     checkPash pash rt (runPash 1 time)
93     (Nothing,Nothing,Just tel) -> do
94         rt <- liftHandlerT $ runDB $ selectList
95             [AccountTel ==. tel] []
96         checkPash pash rt (runPash 2 time)
97     _ -> return $ Unauthorized "Who_are_you!"
98 where
99     getPash = do
100         pash' <- lookupPostParam "pash"
101         return $ fromMaybe "" pash'
102     checkPash pash rt f = do
103         case rt of
104             item:_ -> do
105                 let usrPash = f.accountPash.fromEntity $ item
106                 if usrPash == pash
107                     then return Authorized
108                     else return $ Unauthorized "Who_are_you!"
109             _ -> return $ Unauthorized "Who_are_you!"
110     checkTime f = do
111         time' <- liftHandlerT $ lookupHeader "TIME-STAMP"
112         now <- liftIO getCurrentTime
113         case time' of
114             Just time -> do
115                 let t = read.unpack.decodeUtf8 $ time
116                 let diff = diffUTCTime now t
117                 if diff <= 12 && diff >= (-12)
118                     then f time
119                     else return $ Unauthorized "I_bought_a_watch_last_year!"
120             _ -> return $ Unauthorized "I_bought_a_watch_last_year!"
121
122     fromEntity :: Entity a -> a
123     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 -> Bl.ByteString
34     toContents = defToContent
35   defToContent :: Variable a => RtType -> a -> Bl.ByteString
36   defToContent RtJson = defToContentJson
37   defToContent RtYaml = defToContentYaml
38   defToContent RtXml = defToContentXml
39   defToContentJson :: Variable a => a -> Bl.ByteString
40   defToContentJson = toStrict . A.encode . toValue
41   defToContentYaml :: Variable a => a -> Bl.ByteString
42   defToContentYaml = Y.encode . toValue
43   defToContentXml :: Variable a => a -> Bl.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
68             x
69             provideRepType "application/yaml" $ return $ decodeUtf8 $ toContents RtYaml
70             x
71             provideRepType "application/xml" $ return $ decodeUtf8 $ toContents RtXml
72             x
73         RtOther y -> do
74             addHeader y $ pack $ show x
75             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) = String t
103        toValue (RtCommonFail x) = String x
104        toNodes RtCommonSucc = [xml|null|]
105        toNodes (RtCommonSuccT x) = [xml|#{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 $ SError "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 $ SError "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.Destory
```

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      import qualified GHC.IO.Encoding as E
5      import System.IO
6      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
22     import System.Exit
23     import Control.Concurrent
24     import System.Signal

```

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。

```

25     data Launch = Launch {form ::String}
26     deriving (Show,Data,Typeable)
27     launch = Launch{ form="auto" &= typ "AUTO|YAML|JSON" &= help "格式"
28                 }
29     &= summary ( "dindo-common-"
30                 ++ $(dindo_common_version_quasi)

```

```

31         ++ ";_dindo-database-"
32         ++ $(dindo_database_version_quasi)
33         ++ ";" ++ $(dindo_module_name) ++ "-"
34         ++ $(dindo_module_version)
35         ++ ";_dindo-launch-"
36         ++ showVersion version)

```

```

37     main :: IO ()
38     main = do
39 #ifndef WithoutUTF8
40         E.setLocaleEncoding E.utf8
41         hSetEncoding stdout utf8
42 #endif
43     tid <- myThreadId
44     installHandler sigINT $ \ sig -> do
45         if sig == sigINT
46             then do
47                 putStrLn "going to turn down"
48                 killThread tid
49                 exitSuccess
50             else putStrLn $ "catch" ++ show sig
51     cfg' <- cmdArgs launch >>= cfg
52     warpDindo cfg' itemWarp
53     where
54         itemWarp :: Int -> $(std) -> IO()
55         itemWarp = warp
56     cfg :: Launch -> IO SvrConfig
57     cfg l = getContents >>=
58         (decode'.T.encodeUtf8.T.pack)
59     where
60         tryList :: [a -> SvrConfig] -> [ScError] -> a -> IO SvrConfig
61         tryList [] es a = scError.concatWith "\n\t".map getError $ es
62         tryList (x:xs) es a = do
63             rt <- try.evaluate $ x a :: IO (Either ScError SvrConfig)

```

```

64     case rt of
65         Left e -> tryList xs (e:es) a
66         Right sc -> return sc
67     getError (ScError a) = a
68     concatWith a xs = foldr sig "all_ failed " xs
69     where
70         sig x os = x ++ a ++ os
71     decJ = fromMaybe (throw $ ScError "Invailed_JSON").A.decode.B.fromStrictBS
72     decY = fromMaybe (throw $ ScError "Invailed_YAML").Y.decode
73     decA = tryList [decY,decJ] []
74     decode' = let ll = form l in
75         case map toLower ll of
76             "auto" -> decA
77             "json" -> evaluate.decJ
78             "yaml" -> evaluate.decY
79             _ -> 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 dindo_module_name = stringE "dindo-usrmanage"
11 dindo_module_version = dindo_usrmanage_version_quasi
12 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     , Rtldy (..)
4     , Rtldfed (..)
5     , RtUImg(..)
6     , RtUInfo(..)
7     , RtChPsk(..)
```

```

8   , RtEaddr(..)
9   , RtGEaddr(..)
10  , RtLogin(..)
11  ) where
12
13  import Dindo.Import.Rable
14  import Dindo.Import.Aeson as A
15  import Dindo.Import.Yaml as Y
16  import Dindo.Import.Text as T
17  import Dindo.Import.ByteString as B
18  import Dindo.Import.Yesod
19  import Dindo.Import.Database

```

用户注册返回数据

```

20  data RtRegist = RtRegist
21      { uid :: Text
22      }
23  | RtRegistFail
24      { regReason :: Text
25      }
26  deriving (Eq)
27  instance Show RtRegist where
28      show (RtRegist x) = T.unpack x
29      show (RtRegistFail x) = T.unpack x
30  instance Variable RtRegist where
31      toValue (RtRegist x) = object ["uid" .= x]
32      toValue (RtRegistFail x) = object ["error" .= x]
33      toNodes (RtRegist x) = [xml|<uid>#{x}|]
34      toNodes (RtRegistFail x) = [xml|<error>#{x}|]
35  instance Rable RtRegist where
36      toWhere (RtRegist _) = RtBody
37      toWhere (RtRegistFail _) = RtBody
38      toStatus (RtRegist _) = RtSucc
39      toStatus (RtRegistFail _) = RtFail

```


用户认证信息的返回数据

```

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

```

用户查询认证状态信息

```

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

```

用户信息查询返回结果

```

69  data RtUInfo = RtUInfo
70      { rtuiUid :: Text
71        , rtuiName :: Text
72        , rtuiTel :: Text
73        , rtuiEmail :: Text
74        }
75      | RtUInfoNSU
76  instance Show RtUInfo where
77      show RtUInfoNSU = "no_such_a_user"
78  instance Variable RtUInfo where
79      toValue RtUInfo{..} = object
80          [ "uid" .= rtuiUid
81            , "name" .= rtuiName
82            , "tel" .= rtuiTel
83            , "email" .= rtuiEmail
84          ]
85      toNodes RtUInfo{..} = [xml|
86          <uid> #{rtuiUid}
87          <name> #{rtuiName}
88          <tel> #{rtuiTel}
89          <email> #{rtuiEmail}
90          |]
91  instance Rable RtUInfo where
92      toWhere RtUInfo{..} = RtBody
93      toWhere RtUInfoNSU = RtOther "CONTEXT"
94      toStatus RtUInfo{..} = RtSucc
95      toStatus RtUInfoNSU = RtFail

```

登录

```

96  data RtLogin = RtLoginSucc Text Text
97              | RtLoginFail Text
98  deriving (Show,Eq)
99  instance Variable RtLogin where

```

```

100     toValue (RtLoginSucc u t) = object ["uid".=u,"tmp-token".=t]
101     toValue (RtLoginFail e) = object ["error" .= e]
102     toNodes (RtLoginFail e) = [xml|<error>#{e}|]
103     toNodes (RtLoginSucc u t) =[xml|
104         <uid>#{u}
105         <tmp-token>#{t}
106         |]
107     instance Rable RtLogin where
108         toWhere _ = RtBody
109         toStatus (RtLoginSucc _ _) = RtSucc
110         toStatus (RtLoginFail _) = RtFail

```

获取用户头像返回内容

```

111     data RtUImg = RtUImg ByteString
112                 | RtUImgFail
113     deriving (Eq)
114     instance Show RtUImg
115     instance Variable RtUImg
116     instance Rable RtUImg where
117         returnR (RtUImg img) =
118             selectRep $ provideRepType "image/png" $ return img
119         returnR RtUImgFail = do
120             addHeader "CONTEXT-WHERE" "CONTEXT"
121             addHeader "CONTEXT" "Failed_on_get_image"
122             selectRep $ provideRep $ return (" " :: Text)

```

更改密码的返回值

```

123     data RtChPsk = RtChPsk
124                 | RtChPskFail Text
125     deriving (Eq)
126     instance Show RtChPsk where
127         show (RtChPskFail x) = T.unpack x
128     instance Variable RtChPsk where
129         toValue RtChPsk = Null

```

```

130     toValue (RtChPskFail x) = object ["error" .= x]
131     toNodes RtChPsk = [xml|null|]
132     toNodes (RtChPskFail x) = [xml|<error>#{x}|]
133     instance Rable RtChPsk where
134         toWhere RtChPsk = RtBody
135         toWhere (RtChPskFail _) = RtBody
136         toStatus RtChPsk = RtSucc
137         toStatus (RtChPskFail _) = RtFail

```

收货地址的增删的返回值

```

138     data RtEaddr = RtEaddrAdd Text
139                 | RtEaddrChn
140                 | RtEaddrDel
141                 | RtEaddrFail Text
142     deriving (Eq,Show)
143     instance Variable RtEaddr where
144         toValue (RtEaddrAdd x) = object ["aid" .= x]
145         toValue RtEaddrChn = Null
146         toValue RtEaddrDel = Null
147         toValue (RtEaddrFail x) = object ["error" .= x]
148         toNodes (RtEaddrAdd x) = [xml|<aid>#{x}|]
149         toNodes RtEaddrChn = [xml|null|]
150         toNodes RtEaddrDel = [xml|null|]
151         toNodes (RtEaddrFail x) = [xml|<error>#{x}|]
152     instance Rable RtEaddr where
153         toWhere (RtEaddrAdd _) = RtBody
154         toWhere RtEaddrChn = RtBody
155         toWhere RtEaddrDel = RtBody
156         toWhere (RtEaddrFail _) = RtBody
157         toStatus (RtEaddrAdd _) = RtSucc
158         toStatus RtEaddrChn = RtSucc
159         toStatus RtEaddrDel = RtSucc
160         toStatus (RtEaddrFail _) = RtFail

```

获取地址

```

161   data RtGEadd = RtGEadd [Addr]
162           | RtGEaddFail Text
163   deriving (Eq, Show)
164   instance Variable RtGEadd where
165     toValue (RtGEadd x) = toJSON x
166     toValue (RtGEaddFail x) = object ["error" .= x]
167     toNodes (RtGEadd xs) = [xml|
168       $forall x <- xs
169         <aid>#{addrAid x}
170         <addr>#{addrAddr x}
171         <zip>#{addrZip x}
172       |]
173     toNodes (RtGEaddFail x) = [xml|<error>#{x}|]
174   instance Rable RtGEadd where
175     toWhere (RtGEadd _) = RtBody
176     toWhere (RtGEaddFail _) = RtBody
177     toStatus (RtGEadd _) = RtSucc
178     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  , postGeteadrR
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

```

```

80         insert $ Apic pid uid pic $ Just 0
81     returnR $ case rt of
82         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

这里所有的登录失败都是返回的服务器内部错误

```

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,
114             pash,pack time]
115         let tt = pack $ take 22 time ++ to
116         rt <- liftHandlerT $ tryRunDB $ insert $ TmpToken tt lim uid
117         case rt of
118             Left e -> returnR $ RtLoginFail $ showT e
119             Right _ -> returnR $ RtLoginSucc uid tt
120     where
121         getPash = do
122             pash' <- lookupPostParam "pash"
123             return $ fromMaybe "" pash'

```

用户登出

```

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

```

查询用户信息

```

131 postUsrinfoR :: Handler TypedContent
132 postUsrinfoR = do
133     tuid <- getUid
134     uid' <- lookupPostParam "uid"
135     let uid = fromMaybe tuid uid'
136     rt' <- liftHandlerT $ runDB $ selectList [UsrUid ==. uid] []

```

```

137     case rt' of
138       Entity _ rt:_ -> do
139         let email = usrEmail rt
140         Entity _ item:_ <- liftHandlerT $ runDB $ selectList [AccountUid ==. uid] []
141         returnR $ RtUInfo uid (accountName item) (pack $ show $ accountTel item)
142           email
143       _ -> returnR RtUInfoNSU

```

获得用户头像

```

143 postUshrimgR :: Handler TypedContent
144 postUshrimgR = do
145   tuid <- getUid
146   uid' <- lookupPostParam "uid"
147   let uid = fromMaybe tuid uid'
148   rt' <- liftHandlerT $ runDB $ selectList [ApicUid ==. uid] []
149   case rt' of
150     Entity _ rt:_ -> returnR $ RtUImg $ apicBpic rt
151     _ -> returnR $ RtUImgFail

```

用户信息变更

```

152 postUsrinfochangeR :: Handler TypedContent
153 postUsrinfochangeR = check update
154 where
155   updatePic uid pic' = case pic' of
156     Nothing -> return ()
157     Just pic -> do
158       rt <- sourceToList $ fileSource pic
159       let bpic = B.concat rt
160       updateWhere [ApicUid ==. uid, ApicTyp ==. Just 0] [ApicBpic =. bpic]
161   update (a,b,pic) = do
162     uid <- getUid
163     rt <- liftHandlerT $ tryRunDB $ do
164       when (not $ null a) $
165         updateWhere [AccountUid ==. uid] a

```

```

166         when (not $ null b) $
167             updateWhere [UsrUid ==. uid] b
168             updatePic uid pic
169         returnR $ case rt of
170             Left e -> RtCommonFail $ pack $ show e
171             Right _ -> RtCommonSucc
172         check f = do
173             name <- liftHandlerT $ lookupPostParam "name"
174             tel  <- liftHandlerT $ lookupPostParam "tel"
175             email <- liftHandlerT $ lookupPostParam "email"
176             rname <- liftHandlerT $ lookupPostParam "rname"
177             prcid <- liftHandlerT $ lookupPostParam "prcid"
178             addr <- liftHandlerT $ lookupPostParam "addr"
179             pic  <- liftHandlerT $ lookupFile "pic"
180             let a = pickU [(AccountName,name)]
181             let a' = pickU [(AccountTel,fmap (read.T.unpack) tel)]
182             let b = pickU [(UsrEmail,email),(UsrRname,rname),(UsrPrcid,prcid),(UsrAddr,
183                 addr)]
184             f (a++a',b,pic)

```

修改密码

```

184     postChangpashR :: Handler TypedContent
185     postChangpashR = check changePash
186     where
187         changePash pash = do
188             uid <- getUid
189             rt <- liftHandlerT $ tryRunDB $ updateWhere [AccountUid ==. uid] [
190                 AccountPash ==. pash]
191             returnR $ case rt of
192                 Left e -> RtChPskFail $ pack $ show e
193                 Right _ -> RtChPsk
194             check f = do
195                 pash' <- lookupPostParam "pash"
196                 case pash' of

```

```

196     Nothing -> do
197         returnR $ RtChPskFail "param:␣less␣and␣less"
198     Just x -> f x

```

收获地址

```

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

```

```

228     let aid = pack $ "A" ++ show now ++ aid'
229     rt <- liftHandlerT $ tryRunDB $ insert $ Addr aid uid zipcode addr
230     returnR $ case rt of
231         Left e -> RtEaddrFail $ pack $ show e
232         Right _ -> RtEaddrAdd aid
233     checkAdd f = do
234         addr' <- liftHandlerT $ lookupPostParam "addr"
235         zip' <- liftHandlerT $ lookupPostParam "zip"
236         case (addr', zip') of
237             (Just addr, Just zipcode) -> f (addr, zipcode)
238             _ -> returnR $ RtEaddrFail "param:addr:␣less␣and␣less"
239     spl = do
240         opt <- liftHandlerT $ lookupHeader "OPT"
241         case opt of
242             Just "ADD" -> checkAdd addItem
243             Just "DEL" -> checkChn changeltem
244             Just "CHANGE" -> checkDel delltem
245             _ -> returnR $ RtEaddrFail "header:opt:␣less␣and␣less"

```

获取收货地址

```

246     postGeteaddrR :: Handler TypedContent
247     postGeteaddrR = spl
248     where
249         getByUid uid = do
250             rt <- liftHandlerT $ runDB $ selectList [AddrUid ==. uid] []
251             returnR $ RtGEadd $ map fromEntity rt
252         getByAid aid = do
253             uid <- getUid
254             rt <- liftHandlerT $ runDB $ selectList [AddrAid ==. aid, AddrUid ==. uid] []
255             returnR $ RtGEadd $ map fromEntity rt
256     spl = do
257         uid' <- liftHandlerT $ lookupPostParam "uid"
258         aid' <- liftHandlerT $ lookupPostParam "aid"
259         case (uid', aid') of

```

```

260         (Just uid, _) -> getByUid uid
261         (Nothing, Just aid) -> getByAid aid
262         _ -> 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
5      import qualified GHC.IO.Encoding as E
6      import System.IO
7      import System.Environment
8      import Dindo.Import
9      import Dindo.Common.Auth
10     import Dindo.Import.Digest
11     import qualified Dindo.Import.Text as T
12     import qualified Dindo.Import.ByteString as B
13     import Dindo.Common(dindo_common_version_quasi)
14     import Data.Version
15     import System.Console.CmdArgs
16     import Paths_dindo_tools
17
18     main :: IO ()
19     main = do
20         #ifndef WithoutUTF8
21             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++"\\"
30         _ -> putStr $ a ++ "\u-d\u\"pash="++pash++"\\" \u-H\u\"TIME-STAMP:"++
                show now++"\\"
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](#)