大厂面试课之Go语言

第34课：用户头像

同学们好！截至目前，我们已经将用户为自己选择的头像数据传送到后端微服务，也已经搭建好了基于FastDFS和Nginx的文件服务器，而且就在上节课，我们也学会了在Go语言程序中将文件上传至文件服务器的方法。对于《我家租房网》项目而言，是时候实现完整的用户头像上传功能了。

我们将首先实现支持用户头像上传的后端微服务，再将前端、后端和文件服务器连在一起做完整的功能测试。

后端

首先我们来完善先前编写的UploadAvatar后端微服务。

持久化

我们先为UploadAvatar微服务编写与数据持久化有关的代码。

项目实战

GOPATH/src/iHome/back-end/UploadAvatar/model/mysql.go

```
...
// 更新MySQL数据库中的头像
func UpdateAvatar(username, fileid string) error {
    return db.Model(new(User)).Where("name=?", username).
        Update("avatar_url", fileid).Error
}
...
```

打开UploadAvatar微服务目录下，model子目录中的mysql.go文件，在其中添加有关更新MySQL数据库中的头像的代码：

```
...
// 更新MySQL数据库中的头像
func UpdateAvatar(username, fileid string) error {
    return db.Model(new(User)).Where("name=?", username).
        Update("avatar_url", fileid).Error
}
...
```

定义名为UpdateAvatar的函数，用于更新MySQL数据库中的头像。该函数接收用户名和头像文件凭证两个参数，以用户名为条件，更新用户表中特定记录的头像URL字段为头像文件凭证，并将错误对象返回给函数的调用者。

业务逻辑

在完成UploadAvatar微服务模型层的所有开发工作之后，我们将继续完善用于处理上传头像业务的代码。

项目实战

配置fdfs_client

- 将GOPATH\src\iHome\front-end\conf目录下的fdfs.conf文件复制到GOPATH\src\iHome\back-end\UploadAvatar\conf目录下

首先是复制FastDFS客户机配置文件。这里将front-end工程目录下conf子目录中的fdfs.conf文件，原封不动地复制到UploadAvatar微服务目录下的conf子目录中。

GOPATH/src/iHome/back-end/UploadAvatar/handler/UploadAvatar.go

项目实战

```
...
// 上传头像
func (e *UploadAvatar) Call(ctx context.Context, req *pb.CallRequest, rsp *pb.CallResponse) error {
    logger.Infof("Received UploadAvatar.Call request: %v", req)

    // 创建FastDFS客户机
    client, err := fdfs_client.NewClientWithConfig("./conf/fdfs.conf")
    defer client.Destory()
    if err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_IO
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }
```


GOPATH/src/iHome/back-end/UploadAvatar/handler/UploadAvatar.go

项目实战

```
    // 上传数据到FastDFS
    fileid, err := client.UploadByBuffer(req.Avatar, req.Suffix[1:])
    if err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_IO
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }
```


GOPATH/src/iHome/back-end/UploadAvatar/handler/UploadAvatar.go

项目实战

```
    // 更新MySQL数据库中的头像
    if err := model.UpdateAvatar(req.Username, fileid); err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_DATABASE
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }

    rsp.Data = &pb.Avatar{AvatarUrl: "http://192.168.0.111:8888/" + fileid}

    rsp.Errno = utils.ERROR_OK
    rsp.Errmsg = utils.StrError(rsp.Errno)
    return nil
}
```

打开UploadAvatar微服务目录下，handler子目录中的UploadAvatar.go文件，在其中的Call方法里，添加与向文件服务器上传头像有关的操作：

```
...
// 上传头像
func (e *UploadAvatar) Call(ctx context.Context, req *pb.CallRequest, rsp
*pb.CallResponse) error {
    logger.Infof("Received UploadAvatar.Call request: %v", req)

    // 创建FastDFS客户机
    client, err := fdfs_client.NewClientWithConfig("./conf/fdfs.conf")
    defer client.Destory()
    if err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_IO
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }

    // 上传数据到FastDFS
    fileid, err := client.UploadByBuffer(req.Avatar, req.Suffix[1:])
    if err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_IO
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }

    // 更新MySQL数据库中的头像
    if err := model.UpdateAvatar(req.Username, fileid); err != nil {
        logger.Error(err)
        rsp.Errno = utils.ERROR_DATABASE
        rsp.Errmsg = utils.StrError(rsp.Errno)
        return nil
    }

    rsp.Data = &pb.Avatar{AvatarUrl: "http://192.168.0.111:8888/" + fileid}

    rsp.Errno = utils.ERROR_OK
    rsp.Errmsg = utils.StrError(rsp.Errno)
    return nil
}
...
```

这里首先以FastDFS客户机配置文件的路径为参数，调用FastDFS客户端SDK的NewClientWithConfig函数，创建一个FastDFS客户机对象。接着将请求对象中的头像数据和头像文件名后缀，传给FastDFS客户机对象的UploadByBuffer方法。注意调用该方法的第二个参数不包括文件名后缀中的“.”字符。如果不发生错误的话，用户的头像图片已经上传到FastDFS服务器中了。FastDFS客户机对象的UploadByBuffer方法成功返回字符串形式的头像文件凭证。我们以请求对象中的用户名和上一步得到的头像文件凭证为参数，调用模型层的UpdateAvatar函数，更新MySQL数据库中的头像。若成功，则将头像文件凭证拼接在Nginx服务器的IP地址和侦听端口之后，形成供前端下载头像图片的URL，填入响应对象的特定字段中。

修改用户信息微服务

我们之前编写的用于获取用户信息的微服务，并没有考虑有关用户头像的处理细节，这里需要修改一下。

GOPATH/src/iHome/back-end/GetUser/handler/GetUser.go

项目实战

```
...
// 获取用户信息
func (e *GetUser) Call(ctx context.Context, req *pb.CallRequest, rsp *pb.CallResponse) error {
    ...
    rsp.Data = &pb.User{
        ...
        AvatarUrl: "http://192.168.0.111:8888/" + user.Avatar_url,
        ...
    }
    ...
}
...
```


打开GetUser微服务目录下，handler子目录中的GetUser.go文件，在其中的Call方法里，修改与用户头像有关的代码：

```
...
// 获取用户信息
func (e *GetUser) Call(ctx context.Context, req *pb.CallRequest, rsp
*pb.CallResponse) error {
    ...
    rsp.Data = &pb.User{
        ...
        AvatarUrl: "http://192.168.0.111:8888/" + user.Avatar_url,
        ...
    }
    ...
}
...
```

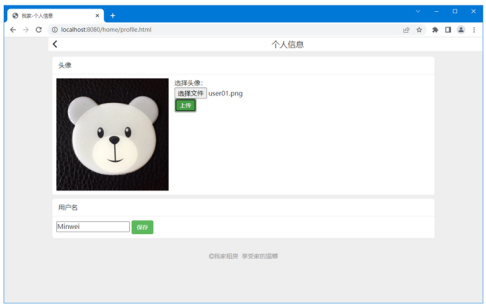
我们从MySQL数据库中读到的头像URL，仅仅是头像文件凭证，并不包括Nginx服务器的IP地址和侦听端口，这里需要进行拼接，再填入响应对象的特定字段，以便于前端浏览器下载和显示。

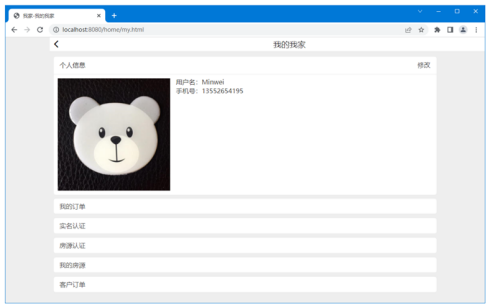
测试

至此，我们已经完成上传头像的全部开发工作。下面我们将对这部分功能进行测试。

执行如下操作：

项目实战





1. 在虚拟机中启动FastDFS和Nginx
2. 启动Consul、后端（UserLogin、GetUser、UploadAvatar）和前端
3. 在用户页面中点击“修改”，进入修改个人信息页面
4. 点击“选择文件”，选择图片文件并点击“上传”
5. 返回用户页面，显示更新后的用户头像

在虚拟机中启动FastDFS和Nginx。启动Consul服务器、UserLogin、GetUser、UploadAvatar后端微服务和前端服务器。通过登录页面登录系统，点击位于搜索页面右上角的用户名，进入用户页面，点击个人信息栏中的“修改”，进入修改个人信息的页面，点击“选择文件”，选择图片文件，点击“上传”。这时用户头像应显示在头像栏左侧。返回用户页面，用户头像应显示在个人信息栏左侧。

更多精彩，敬请期待

谢谢大家，我们下节课再见！