1. 关注用户接口

```
        user_id = g.user_id
        author_id = request.json['author_id']
        # 2. 查询用户关系
        query = db.session.query(Relation)
        query = query.filter(Relation.user_id == user_id,
                             Relation.author_id == author_id)
        relation = query.first()
        # 3. 如果用户关系存在
        if relation:
            #     更新用户关系为 关注
            relation.relation = Relation.RELATION.FOLLOW
            relation.update_time = datetime.now()
        # 4. 如果用户关系不存在
        else:
            #     新建用户关注关系
            new_relation = Relation(user_id=user_id,
                                    author_id=author_id,
                                    relation=Relation.RELATION.FOLLOW)
            db.session.add(new_relation)
        # 5. 更新作者的粉丝数量
        db.session.query(User).filter(User.id == author_id).update({
            "fans_count": User.fans_count + 1
        })
        # 6. 更新当前用户的关注数量
        db.session.query(User).filter(User.id == user_id).update({
            "following_count": User.following_count + 1
        })
```

FollowUserResource › post()

2. 取消关注

```python
        # 1. 查询用户关系
        query = db.session.query(Relation)
        query = query.filter(Relation.user_id == g.user_id,
                             Relation.author_id == author_id)
        relation = query.first()
        # 2. 关系存在
        if relation:
            #    1. 更新为逻辑删除
            relation.relation = Relation.RELATION.DELETE
            relation.update_time = datetime.now()
            #    2. 更新当前用户的关注数量
            #    3. 更新作者的粉丝数量
            # 5. 更新作者的粉丝数量
            db.session.query(User).filter(User.id == author_id).update({
                "fans_count": User.fans_count - 1
            })
            # 6. 更新当前用户的关注数量
            db.session.query(User).filter(User.id == g.user_id).update({
                "following_co
            })
            #    4. 保存数据
            db.session.commit()
```

common.models.user
class User(db.Model)

用户基本信息

UnFollowUserResource › delete()

3. 关注列表

user:1　　关注: 2,3,4,5,6,7,8,9，11

user:1　　粉丝: 3，7，9，10，11，15

user:1　　相关关注: 3，7，9，11

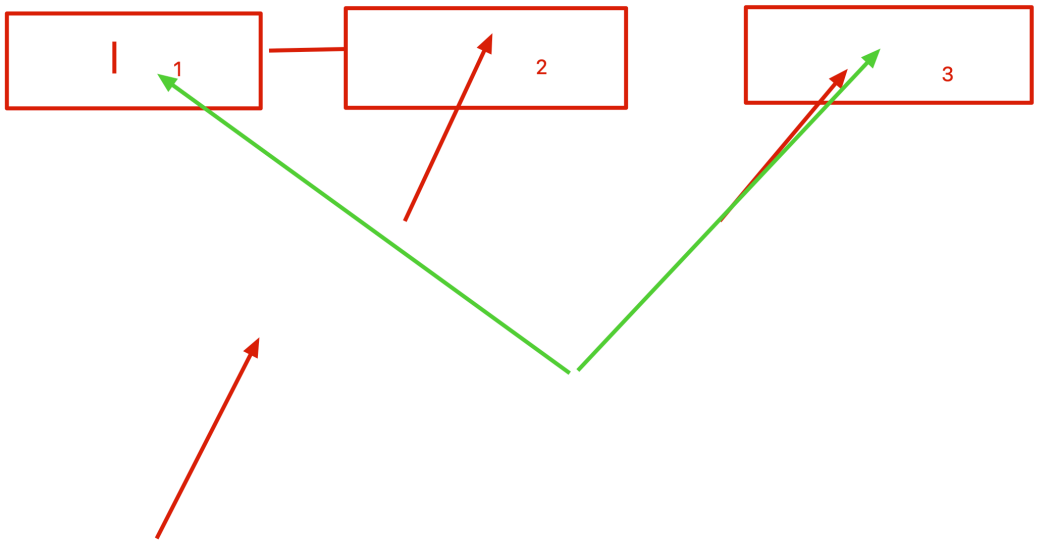
```python
    per_page = args.per_page
    # 4. 查询当前页关注用户数据
    query = db.session.query(User)
    query = query.join(Relation, User.id == Relation.author_id)
    query = query.filter(Relation.user_id == g.user_id,
                         Relation.relation == Relation.RELATION.FOLLOW)
    follow_users = query.paginate(page, per_page)
    # 5. 读取关注用户 id 列表 follow_ids   当前页数据，可迭代follow_users.items
    follow_ids = set([user.id for user in follow_users.items])
    # 6. 查询当前用户的粉丝 id 列表 fans_ids
    query = db.session.query(Relation.user_id)
    query = query.filter(Relation.author_id == g.user_id,
                         Relation.relation == Relation.RELATION.FOLLOW)
    fans_users = query.all()
    fans_ids = set([relation.user_id for relation in fans_users])
    # 7. 取交集 mutual_ids= set(follow_ids).intersection(set(fans_ids))
    mutual_ids = follow_ids.intersection(fans_ids)

    # 8. 构造用户数据返回，在 mutual_ids 的用户就是相互关注
    users = [
        {
            "id": user.id,
            "name": user.name,
            "fans_count": user.fans_count,
            "profile_photo": user.profile_photo,
            "mutual_follow": user.id in mutual_ids
        }

        for user in follow_users.items
```

FollowUserResource › get()

4. 发布评论

```python
    args = parser.parse_args()
    content = args.content
    article_id = args.article_id
    # 2. 新建评论对象
    comment = Comment(user_id=g.user_id,
                      content=content,
                      article_id=article_id)
    db.session.add(comment)
    # 3. 更新文章评论数量
    db.session.query(Article).filter(Article.id == article_id).update({
        "comment_count": Article.comment_count + 1
    })
    # 4. 保存数据
    db.session.commit()
    # 5. 返回
    return {'message': "OK"}
```

5. 评论列表




```
    # 4. 设置要读取的字段提高性能
    fields = [
        Comment.id,
        Comment.content,
        Comment.ctime,
        Comment.user_id,
        Comment.reply_count,
        Comment.like_count,
        User.name,
        User.profile_photo,
    ]
    query = db.session.query(*fields)
    # 5. join User,Comment
    query = query.join(User, Comment.user_id == User.id)
    # 6. filter (article_id == article_id , id > last_comment_id)
    query = query.filter(Comment.article_id == article_id,
                         Comment.id > last_comment_id)
    # 7. limit 数据
    comments = query.limit(limit).all()

    # 8. 构造数据返回
    results = [
```

6. 回复评论

```
26
27      if parent_id:
28          # 子评论
29          # 创建子评论
30          comment = Comment(user_id=g.user_id,
31                            content=content,
32                            parent_id=parent_id)
33          # 更新父评论回复数量
34          db.session.query(Comment).filter(Comment.id == parent_id).update({
35              "reply_count": Comment.reply_count + 1
36          })
37
38      else:
39          # 2. 新建评论对象
40          comment = Comment(user_id=g.user_id,
41                            content=content,
42                            article_id=article_id)
43          # 3. 更新文章评论数量
44          db.session.query(Article).filter(Article.id == article_id).update({
45              "comment_count": Article.comment_count + 1
46          })
47      db.session.add(comment)
48      # 4. 保存数据
49      db.session.commit()
50      # 5. 返回
51      return {'message': "OK"}
```

CommentsResource › post()

7. 回复列表

```
56          # 2. 添加要解析的参数 article_id, last_comment_id, limit
57          # 3. 解析参数
58          parser = RequestParser()
59          parser.add_argument('article_id', type=int)  非必传
60          parser.add_argument('last_comment_id', default=0, type=int)
61          parser.add_argument('limit', default=10, type=int)
62
63          parser.add_argument('parent_id', type=int)
64
65          args = parser.parse_args()
66          article_id = args.article_id
67          last_comment_id = args.last_comment_id
```

```
    # parent_id 存在就过滤回复评论
    if parent_id:
        query = query.filter(Comment.parent_id == parent_id,
                             Comment.id > last_comment_id
                             )
    # 否则过滤文章评论
    else:
        # 6. filter (article_id == article_id , id > last_comment_id)
        query = query.filter(Comment.article_id == article_id,
                             Comment.id > last_comment_id)
    # 7. limit 数据
```