中山大學
SUN YAT-SEN UNIVERSITY

# Ehelp
# 软件设计文档

林国丹　李敏惠　李为　胡南　童云钊 符永宁

# 目录

一、项目概述与背景.....3
1、项目背景介绍.....3
2、项目概述及目标.....3
3、项目团队成员.....3
二、产品框架.....4
三、详细技术.....5
3.1、前端技术.....5
3.1.1 前端概述.....5
3.1.2 安卓前端技术策略.....7
3.1.3 后期技术研究.....9
3.2、后端技术.....11
3.2.1 后端概述.....11
3.2.1 主要技术的原理及优势.....12
四、总结.....20
文档版本控制.....21

# 一、项目概述与背景

## 1、项目背景介绍

目前社会上的老年人生活问题、儿童安全问题、社区服务管理问题和社会关系一定程度上趋于冷漠等情况都日益突出。比方说针对老年人的生活问题来说，目前中国 65 岁以上的老年人口已经占全国总人口数的 10.1%， 这一比例仍在不断增长。老年人属于疾病高发群体，且由于身体各项机能的衰退，在日常生活中也比较容易遇到一些问题。但由于某些法制政策的不完善，目前仍然存在“空 巢老人”现象不断加剧，医疗保险覆盖率低、养老体系不健全等社会问题，导致一些老年人在遇到危险困难时很难得到有效的救助。 而儿童是祖国的未来，家庭的希望。但小孩子由于身心发育尚未成熟，缺乏辨别善恶是非的能力，易轻信他人，且自我保护能力差，因此很容易受到伤害。据统计，我国每年约有 5.5 万名未成年人意外死亡，每年失踪儿童数字 更是高达 20 万以上。如何能切实有效地降低孩子遇难的风险，保障孩子的安全是现今很多家庭所关注的问题。现今中国正处于转型时期，道德建设滞后，人们之间缺乏信任。加上近年来，由于一些讹诈案件层出不穷，施助者基本权益得不到保证，好心反遭坏报，导致现在许多人看到需要帮助的人不敢伸出援助之手，担心祸及自身， 这直接阻断了人们之间的帮助的桥梁。

## 2、项目概述及目标

“易助”项目本着“让帮助变得简便有效，让人人都乐于帮助”的理念，针对弱势群体（初期为老人、小孩和残疾人），把高新技术运用到他们的日常生活中，提供求助、求救和提问三大功能，简化各种帮助的形式，设计了不同的模块来鼓励大家互帮互助并建立信誉和权益保障体系，致力于搭建一个更加简便、有效、安全的求助帮助 020 平台，高效连接求助者和爱心志愿者。

## 3、项目团队成员

李敏惠，邮箱：1341059201@qq.com

林国丹，邮箱：824928207@qq.com

童云钊，邮箱：314342613@qq.com

胡　南，邮箱：1067861587@qq.com

李　为，邮箱：123653079@qq.com

符永宁，邮箱：535802703@qq.com

# 二、产品框架

EHelp 项目开发过程总体采用前后端分离的方法，后端实现功能接口，前端通过访问接口获取 JSON 数据用于展示。后端采用 J2EE（Spring+Springmvc+Hibernate）的架构，前端采用的是 Android 原生开发。前端发送请求至 Nginx 服务器，Nginx 通过实现配置好的方式将请求转发至不同的 web 服务器。

整体项目部署图如下：

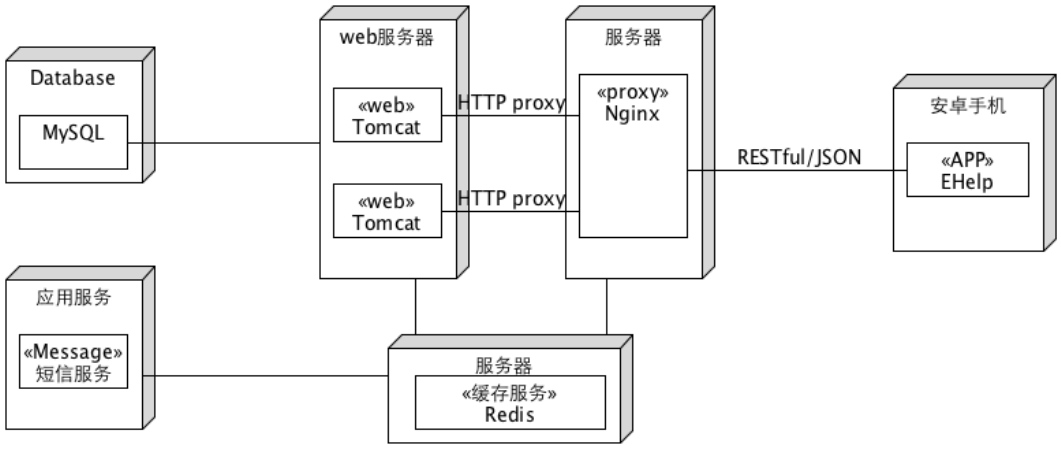


图 2-1 项目部署图

# 三、详细技术

## 3.1、前端技术

### 3.1.1 前端概述

目前前端主要开发安卓版本，Android APP 的开发采用 Android 原生开发技术开发，语言为 Java，安卓版本我们目前也做了很多优化工作，比如核心功能的完善，app cookie 缓存，等等。为了不增加项目的复杂程度， 我们在开发的过程中适当的引入了一些安全可靠适用的第三分类库。

整体的安卓构架：

为了保持项目代码的整洁和利于组内成员的协作开发，减少 git 在合并代码时出现冲突的情况， 针对每个人负责的模块不同，也对项目进行了不同的模块的拆分， 对于一些公用的代码统一放入 utility， activity 统一放入包的第一层目录下面， 方便大家查看代码进度等等。下面是我们小组的整体代码划分截图。

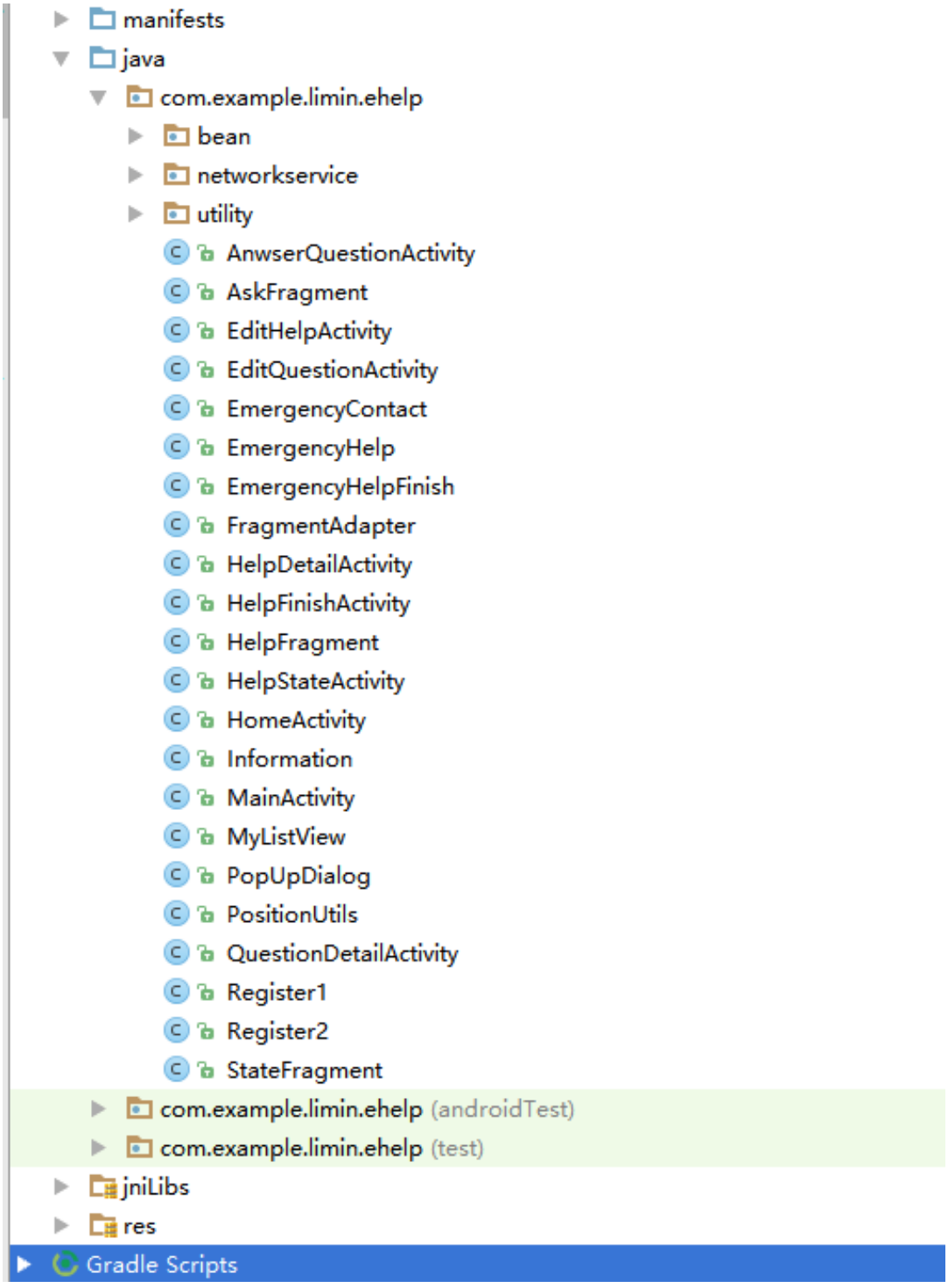
manifests
java
com.example.limin.ehelp
bean
networkservice
utility
AnwserQuestionActivity
AskFragment
EditHelpActivity
EditQuestionActivity
EmergencyContact
EmergencyHelp
EmergencyHelpFinish
FragmentAdapter
HelpDetailActivity
HelpFinishActivity
HelpFragment
HelpStateActivity
HomeActivity
Information
MainActivity
MyListView
PopUpDialog
PositionUtils
QuestionDetailActivity
Register1
Register2
StateFragment
com.example.limin.ehelp (androidTest)
com.example.limin.ehelp (test)
jniLibs
res
Gradle Scripts

图 3-1 安卓代码结构图

对应模块的解释:

| 模块 | 内容 |
| --- | --- |
| bean | 从网络中获取的数据的最小单位。每个 bean 既是网络访问中用都到的实体，也是持久化用到的实体 |
| manifests | 存放一些权限管理 |
| res | 存放一些静态的页面和静态文件如图片等等 |
| networkservice | 封装了网络访问操作 |
| utility | 是一些我们需要用到的工具，如校验，封装的语音助手等。 |
| . | 主要是 activity 逻辑的编写,集中处理业务逻辑。前缀为 fragment 的为动态加载的界面 |

为了产品更加稳定同时减少开发量，前端在实现时我们也使用了很多的第三方库，具体如下：

| 名称 | 功能 |
| --- | --- |
| retrofit | 用于网络访问 |
| gson | 解析 json |
| AMap3DMap_5 | 高德地图的相关依赖包， 用于内嵌地图界面和获取用户的当前定位，距离计算等等 |
| smarttablayout | 用于管理界面核心首页的 tab 界面， 支持滑动， 点击等不同场景切换 |
| roundedimageview | 用于完成应用里面的圆角图片的转换 |
| SwipeRefreshLayout | 实现网络访问延迟时的等待提示和手动刷新 |
| constraint-layout | 引如最新的约束性布局 |

### 3.1.2 安卓前端技术策略

- **网络访问：**

网络访问首先需要解决两个问题， 一个是如何进行进行一个 api 编写定义， 另外一个是如何进行为何 cookie

之前有尝试自己利用安卓本身提供的网络访问， 去写对外的访问接口， 后面代码实现的并不太优雅，项目其他成员也难于调用这块的 api 访问代码，于是就用了一个页面比较受欢迎的网络访问库。

**使用 retrofit 进行网络访问，**

注解的语法使得编码规范可维护，逻辑清晰。如：

```
@GET("/api/users/{user_id}")
Call<UserResult> requestUser(@Path("user_id") int user_id);
```

这就完成了一个接口定。

编程思想上，retrofit 减少解耦，降低耦合，让接口开发灵活，不同 api 之间互相不干扰。代码风格上，retrofit 使用注解方式，代码简洁，易懂，易上手。设计思想上，retrofit 采用建造者模式，开发构建简便。

**如何对 cookie 进行维护呢？**

基本思想是， 首先根据 http 协议，将 cookie 从请求头中解析出来， 为了在整个 app 运行的过程中都使用到同一个 cookie， 我们把 cookie 存入在一个单例模式的缓冲对象中，这样的话，整个 app 的运行时期， 就能获取同一个 cookie, 为了持久化，这里还实现了将 cookie 存入到 app 的私有区。

```
// 维护cookie 和 记录id
String cookit = "";
String headersString = response.headers().toString();
String pattern = "(JSESSIONID[^;]*;)[\\s\\S]*(user[^;]*;)";
Pattern r = Pattern.compile(pattern);
Matcher m = r.matcher(headersString);
if (m.find()) {
    if (m.group(1) != null) {
        cookit += m.group(1);
    }
    if (m.group(2) != null) {
        cookit += m.group(2);
    }
}

ToastUtils.show(APITestActivity.this, cookit);
CurrentUser.cookie = cookit;
CurrentUser.id = response.body().data.id;
```

**还有更好的么？**

对网络处理错误的抽象， 我们知道网络处理更加一般的说只有两种状态，一种为成功，一种为失败， 为了避免频繁的编写一下网络逻辑代码，我们把网络处理基于 api 维度进行封装起来，对外只是提供 OnSuccess OnFailure 两个接口， 那么调用的时候，用户只需要实现这个结果接口即可。

抽象的网络结果处理接口

```
public interface ApiServiceRequestResultHandler {
    public void onSuccess(Object dataBean);
    public void onError(Object errorMessage);
}
```

**• 自动登录：**

不同于 web，Android 手机属于用户个人设备，往往不需要在每次打开时都需要输入用户名密码进行登录。一般来说，一个 APP 需要让用户感觉到只输入过一次用户名密码。这就需要所谓的“自动登录”。

自动登录在用户以往登录过，但 session 超时等原因造成未登录的时候，APP 携带用户的验证信息，自动登录，与后台建立会话。

为了更好的用户体验，消除掉那仅剩一次的输入用户名密码过程，实现自动登录， 我们将第一次登录后返回的 token 进行保存下来， 下次用于进入 app 的时候，利用这个 token 直接和服务器进行通信，登录成功后，重新更新 app 的数据， 安全性方面，token 存放在程序私有数据空间，一般情况下（手机没有被 root）不可被访问，保证了用户登录的安全性。

### 3.1.3 后期技术研究

图片处理一直是 app 性能最大的瓶颈， 如何处理不好，那么在用户打开 app 的时候，由于 app 大量的进行图片的加载，会导致内存溢出，为了项目后期对图片的优化，我们项目同时也对图片这一块进行了技术的预研，包括从网络获取图片，图片的缓存策略等等。

**• 图片加载：**

采用 universal-image-loader（UIL），主要有以下优点：

1. 多线程下载图片，图片可以来源于网络，文件系统，项目文件夹 assets 中以及 drawable 中等；

2. 支持随意的配置 ImageLoader，例如线程池，内存缓存策略，硬盘缓存策略，图片显示选项以及其他的一些配置；

3. 支持图片的内存缓存，文件系统缓存或者 SD 卡缓存；

4. 支持图片下载过程的监听；

5. 根据控件(ImageView)的大小对 Bitmap 进行裁剪，减少 Bitmap 占用过多的内存；

6. 较好的控制图片的加载过程，例如暂停图片加载，重新开始加载图片，一般使用在 ListView,GridView 中，滑动过程中暂停加载图片，停止滑动的时候去加载图片；

7. 提供在较慢的网络下对图片进行加载。

**• 缓存：**

流量、内存的精打细算是安卓开发常需面对的问题提，UIL 提供了很好的缓存策略：

1. UI 请求数据，使用唯一的 Key 值索引 Memory Cache 中的 Bitmap；

2. 内存缓存：缓存搜索，如果能找到 Key 值对应的 Bitmap，则返回数据。否则执行第三步；

3. 硬盘存储：使用唯一 Key 值对应的文件名，检索 SDCard 上的文件；

4. 如果有对应文件，使用 BitmapFactory.decode*方法，解码 Bitmap 并返回数据，同时将数据写入缓存。如果没有对应文件，执行第五步；

5. 下载图片：启动异步线程，从数据源下载数据(Web)；

6. 若下载成功，将数据同时写入硬盘和缓存，并将 Bitmap 显示在 UI 中。

原理如图：

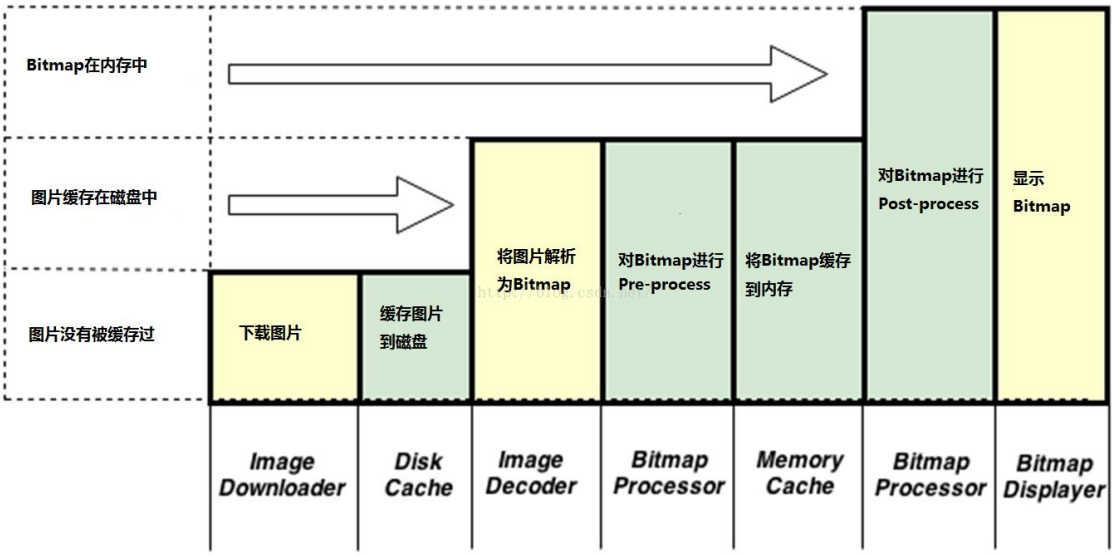


图 3-2 缓存原理

# 3.2、后端技术

## 3.2.1 后端概述

易助项目后端整体采用Spring+Springmvc+Hibernate 的开发框架，结合Nginx、Junit、Redis 等技术完成项目开发。

总体后端技术选型表如下：

| 项目 | EHelp |
|---|---|
| 1.语言 | Java |
| 2.web 框架 | Spring+Springmvc |
| 3.ORM 框架 | Hibernate |
| 4.关系数据库 | MySQL |
| 5.数据缓存(非关系) | Redis |
| 6.负载均衡机制 | Nginx |
| 7.数据交换格式 | JSON |
| 8.单元测试框架 | Junit |

### 3.2.1 主要技术的原理及优势

- **RESTful 及 JSON**

RESTful 是一种软件架构设计风格， 其原则是客户端和服务器之间的交互在请求之间是**无状态**的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。在有 RESTful 风格的 Web 服务中，每个资源都有一个地址。资源本身都是方法调用的目标，方法列表对所有资源都是一样的。这些方法都是标准方法，包括 HTTP GET、POST、PUT、DELETE，还可能包括 HEADER 和 OPTIONS。

JSON 是一种轻量级的**数据交换格式**，通过**键值对**保存各种对象。JSON 采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。简单的 JSON 对象如下：

```
{"firstName": "John"}
```

EHelp 项目采用的是**前后端分离**的方式，前端通过访问 RESTful API，后端返回 JSON 格式的数据用于前端展示。在项目初期我们小组完成接口文档的编写，所有接口均符合 RESTful 风格，如下：

| | |
|---|---|
| 接口功能 | 获取问题列表 |
| 接口地址 | /api/questions |
| 接口参数 | |
| 接口返回值 | {<br>“status”：200/500,<br>“data”：[{<br>“id”：问题 id,<br>“title”：“问题标题”，<br>“description”:“问题描述”,<br>“date”：“提问时间”,<br>“asker_username”：“提问者用户名”，<br>“asker_avatar”：“提问者头像路径”, |

| | |
|---|---|
| | “answer_num”：回答数<br>},<br>{},{}],<br>“errmsg”:“status=500 时返回错误信息”<br>}<br>注：<br>当 status=200，没有 errmsg 信息<br>当 status=500，没有 data 信息 |
| 请求方法 | GET |

后端根据接口文档实现各个接口，完成各个业务需求。

- **Spring**

Spring 是一个分层的 JavaSE/EE **full-stack(一站式)**轻量级开源框架。Spring 是一个容器，通过反转控制(IoC)和依赖注入(DI)来实现高内聚、低耦合的应用。除此之外它可以整合很多第三方框架，它还提供面向切面编程（AOP）的能力，对数据库事务的管理尤其方便。

Spring 的核心容器包括 Core、Beans、Context、EL 模块，同时也包含 AOP、Aspects 模块，整体架构图如下：

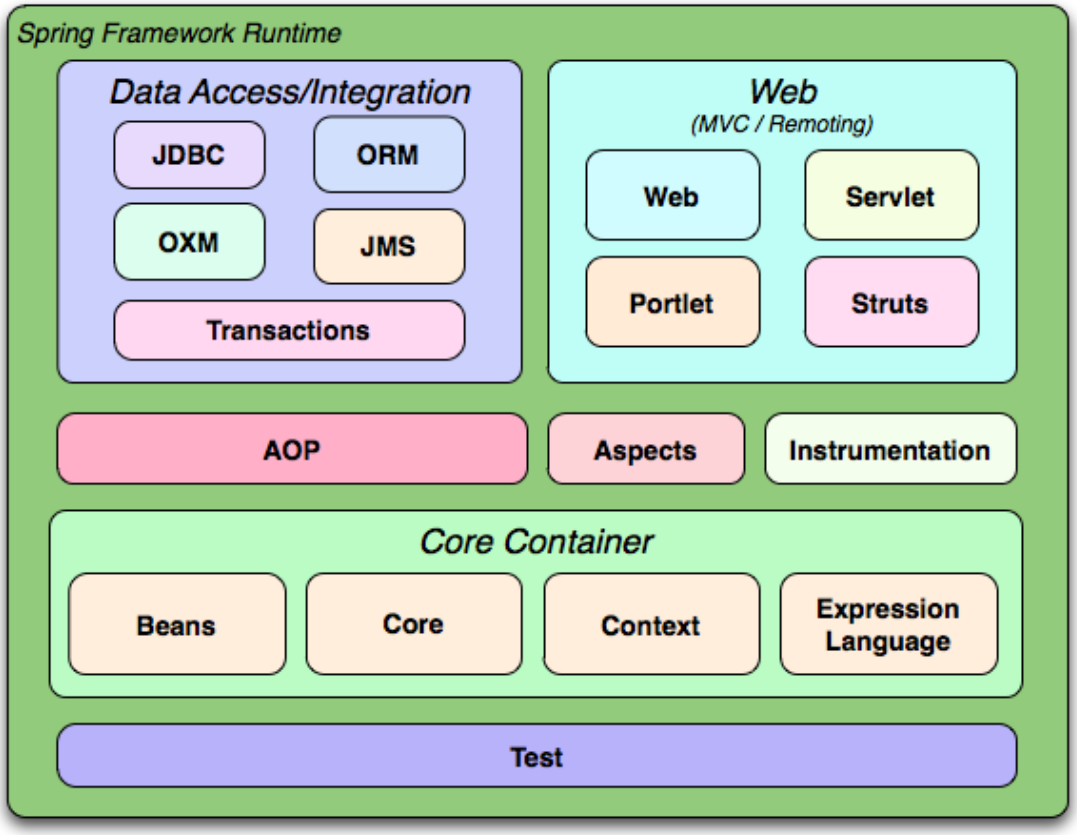


图 3-3 Spring 整体框架图

Spring 是一个分层架构，由 7 个定义良好的模块组成。Spring 模块构建在核心容器之上，核心容器定义了创建、配置和管理 bean 的方式，组成 Spring 框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。每个模块的功能如下：

核心容器:核心容器提供Spring 框架的基本功能。核心容器的主要组件是BeanFactory，它是工厂模式的实现。BeanFactory 使用控制反转（IOC）模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。

Spring 上下文:Spring 上下文是一个配置文件，向Spring 框架提供上下文信息。Spring 上下文包括企业服务，例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。

Spring AOP：通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。所以，可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

Spring DAO：JDBC DAO 抽象层提供了有意义的异常层次结构，可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理，并且极大地降低

了需要编写的异常代码数量（例如打开和关闭连接）。Spring DAO 的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。

Spring ORM：Spring 框架插入了若干个 ORM 框架，从而提供了 ORM 的对象关系工具，其中包括 JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

Spring Web 模块：Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文。所以，Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

Spring MVC 框架：MVC 框架是一个全功能的构建 Web 应用程序的 MVC 实现。通过策略接口，MVC 框架变成为高度可配置的，MVC 容纳了大量视图技术，其中包括 JSP、Velocity、Tiles、iText 和 POI。

- **Springmvc**

Springmvc 是一种基于 Java 的实现了 Web MVC 设计模式的请求驱动类型的轻量级 Web 框架，即使用了 MVC 架构模式的思想，将 web 层进行职责解耦，基于请求驱动指的就是使用请求-响应模型，框架的目的就是帮助我们简化开发。

Springmvc 处理请求的流程如下：

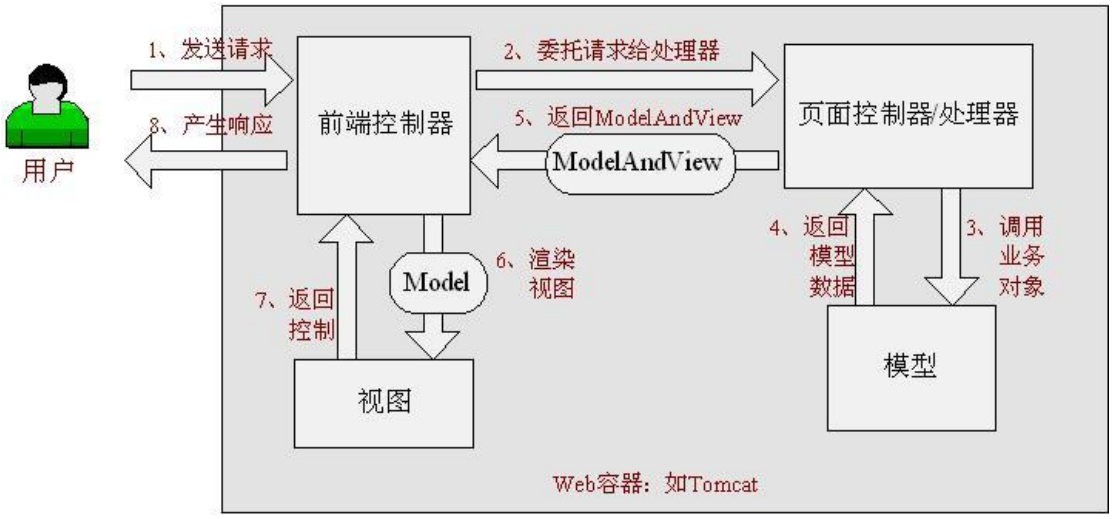


图 3-4 Springmvc 处理请求流程图

具体执行步骤如下：

1．首先用户发送请求到前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分；对应图中的 1、2 步骤；

2．页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，这个对象在 Springmvc 中叫命令对象，并进行验证，然后将命令对象委托给业务对象进行处理；处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名）；对应图中的 3、4、5 步骤；

3．前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染；对应图中的步骤 6、7；

4．前端控制器再次收回控制权，将响应返回给用户，对应图中的步骤 8；

至此整个结束。

- **Hibernate**

Hibernate 是一个对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架。Hibernate 可以自动生成 SQL 语句，自动执行，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用。特别的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

Hibernate 的核心如下：

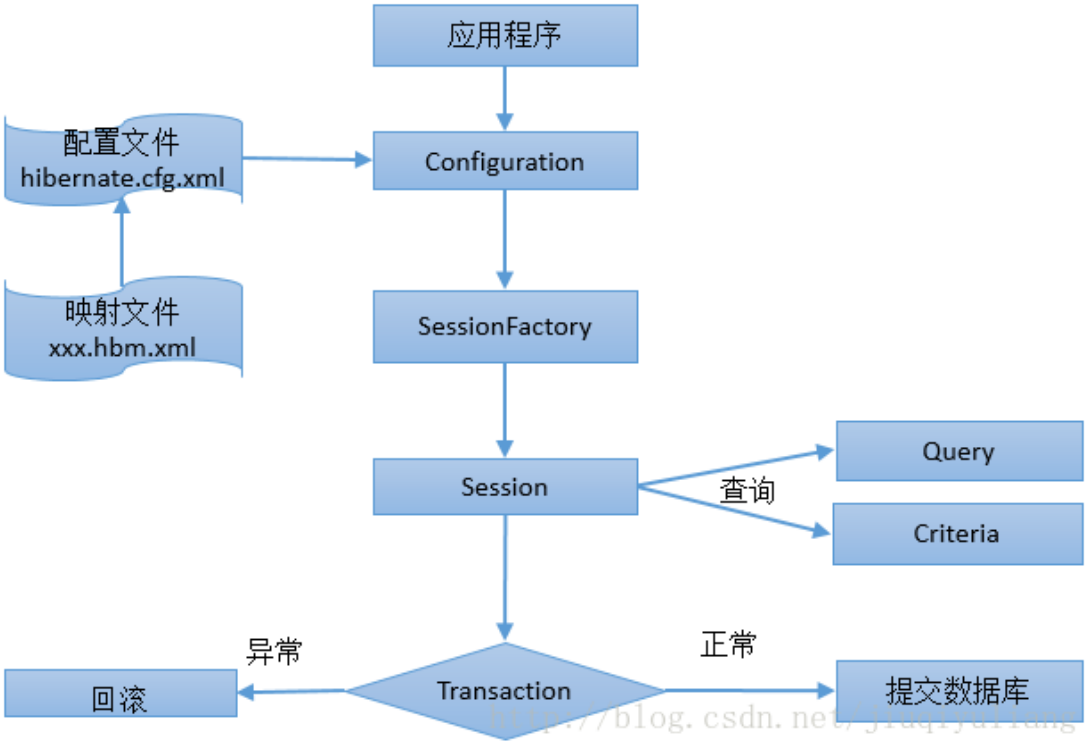


图 3-5　Hibernate 核心图

使用 Hibernate 首先要配置 hibernate.cfg.xml 文件，然后配置映射文件或者直接使用注解。通过 SessionFactory 创建 Session 对象对数据库进行 CURD 操作。

- **Redis**

Redis 是完全开源免费的，遵守 BSD 协议，是一个高性能的 key-value 数据库。Redis 是一个开源的使用 ANSI C 语言编写、遵守 BSD 协议、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。它通常被称为数据结构服务器，因为值(value)可以是字符串(String)，哈希(Map)，列表(list)，集合(sets)和有序集合(sorted sets)等类型。

Redis 具有几个明显的优势：①性能极高。Redis 能读的速度是 110000 次/s,写的速度是 81000 次/s。②丰富的数据类型。Redis 支持二进制案例的 Strings，Lists，Hashes，Sets 及 Ordered Sets 数据类型操作。③原子。Redis 的所有操作都是原子性的，同时 Redis 还支持对几个操作全并后的原子性执行。④丰富的特性。Redis 还支持 publish/subscribe，通知 key 过期等等特性。

Redis 的内部内存管理如下:

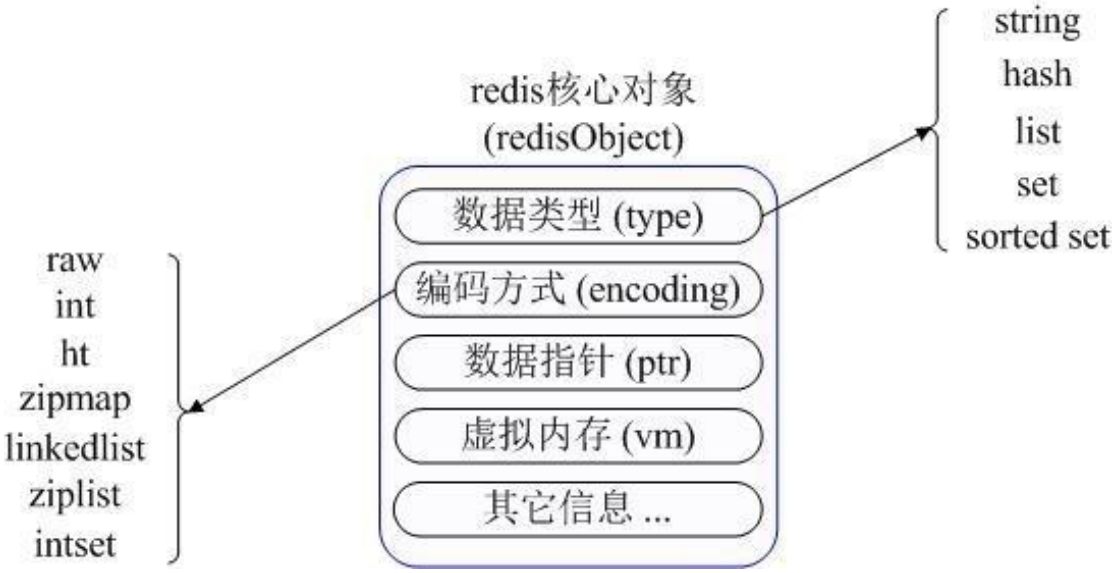

图 3-6 Redis 的内部内存管理

首先 Redis 内部使用一个 redisObject 对象来表示所有的 key 和 value。redisObject 最主要的信息如上图所示：type 代表一个 value 对象具体是何种数据类型，encoding 是不同数据类型在 redis 内部的存储方式，比如：type=string 代表 value 存储的是一个普通字符串，那么对应的 encoding 可以是 raw 或者是 int，如果是 int 则代表实际 redis 内部是按数值型类存储和表示这个字符串的。只有打开了 Redis 的虚拟内存功能，此字段才会真正的分配内存，该功能默认是关闭状态的。

- **Nginx**

Nginx 是一个高性能的 HTTP 和反向代理服务器，也是一个 IMAP/POP3/SMTP 服务器。

Nginx 的模块从结构上分为核心模块、基础模块和第三方模块：①核心模块：HTTP 模块、EVENT 模块和 MAIL 模块；②基础模块：HTTP Access 模块、HTTP FastCGI 模块、HTTP Proxy 模块和 HTTP Rewrite 模块；③第三方模块：HTTP Upstream Request Hash 模块、Notice 模块和 HTTP Access Key 模块。

Nginx 模块常规的 HTTP 请求和响应的过程如下：

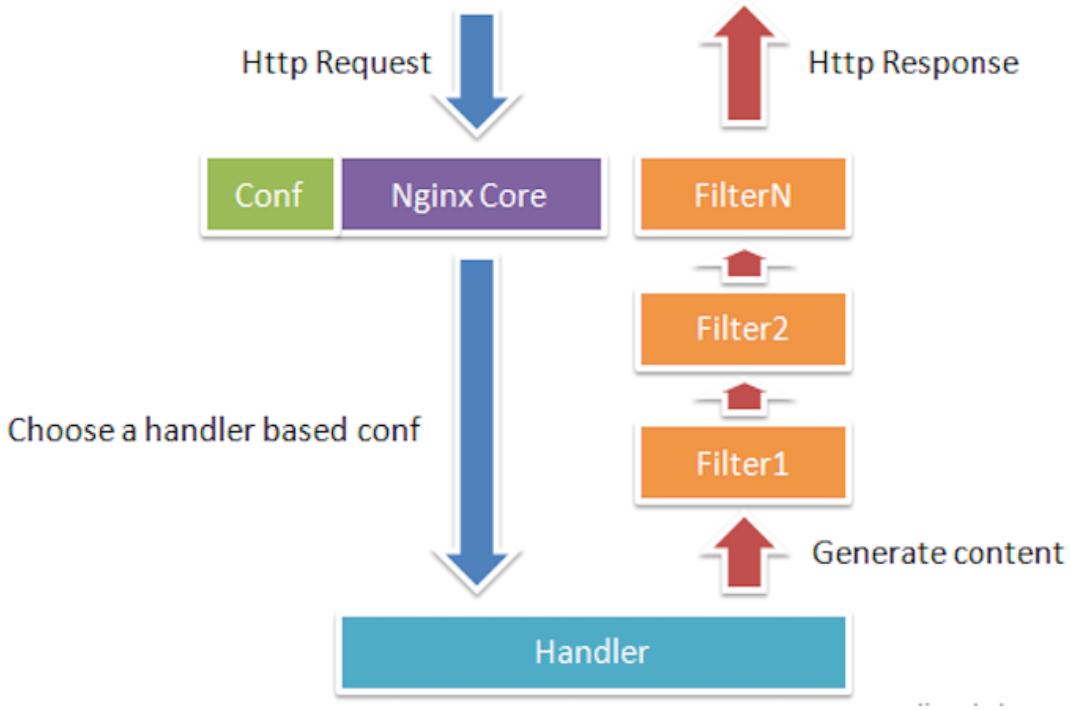


图 3-7 Nginx 模块常规的 HTTP 请求和响应的过程

当 Nginx 接到一个 HTTP 请求时，它仅仅是通过查找配置文件将此次请求映射到一个 location block，而此 location 中所配置的各个指令则会启动不同的模块去完成工作，因此模块可以看做 Nginx 真正的劳动工作者。通常一个 location 中的指令会涉及一个 handler 模块和多个 filter 模块（当然，多个 location 可以复用同一个模块）。handler 模块负责处理请求，完成响应内容的生成，而 filter 模块对响应内容进行处理。

# 四、总结

秉着简便高效的原则，EHelp 项目的技术选型都是我们小组成员经过仔细研究讨论过选择的。从前端都后端，从包图到类图，我们都按照合理的方式进行开发。由于资源有限，我们只是选择了 1 核 1GB 内存 1Mbps 带宽的云服务器作为项目服务器，因此在开发过程中有时会出现服务器宕机的情况，所以我们也编写了脚本实现了服务器宕机后自动重启。同时也因为如此，我们在开发过程中采用了很多提高性能的方法，包括前后端都使用缓存，使用 Nginx 做反向代理，实现 Redis 共享内存缓存等等，通过最终获得的效果也可以让人满意。当然，现在实现的还有很多可以改进的地方，如果有机会，我们会再继续完善架构，实现一个既稳定而又能处理高并发请求的系统。

# 文档版本控制

| 修订日期 | 版本号 | 修订人 | 描述 |
| --- | --- | --- | --- |
| 2017-6-20 | 0.1 | 林国丹 | 建立文档 |
| 2017-6-22 | 0.2 | 李敏惠 | 添加项目概述与背景 |
| 2017-6-24 | 0.3 | 林国丹 | 添加产品框架概述 |
| 2017-6-25 | 0.4 | 符永宁 | 添加前端技术介绍 |
| 2017-6-28 | 0.5 | 林国丹 | 添加后端技术介绍 |
| 2017-7-2 | 0.6 | 符永宁 | 追加前端后期技术研究部分 |
| 2017-7-4 | 0.7 | 林国丹 | 追加总结部分 |
| 2017-7-7 | 1.0 | 李敏惠 | 排版，校对 |