**电子病历**

**技术框架及开发环境应用介绍**

**作成者：王艳丽**

**作成日：2022/3/4**

**目录**

[概述 1](#_Toc97319587)

[1． 后台程序架构 1](#_Toc97319588)

[1.1 开发所用框架 1](#_Toc97319589)

[1.2 业务工程组成，工程一览图 1](#_Toc97319590)

[1.3代码引入eclipse 3](#_Toc97319591)

[1.4单个工程结构 3](#_Toc97319592)

[1.5后台代码开发涉及到几个部分 4](#_Toc97319593)

[1.6后台代码的配置文件 5](#_Toc97319594)

[2． 前端程序架构 7](#_Toc97319595)

[2.1前端Ajax框架 7](#_Toc97319596)

[2.2前端业务代码结构 8](#_Toc97319597)

[3． 代码流程 10](#_Toc97319598)

[3.1前端代码-布局 10](#_Toc97319599)

[3.2后台代码 19](#_Toc97319600)

[4． 版本发布过程 23](#_Toc97319601)

# 概述

本次介绍主要包含后台程序架构、前端程序架构、病历开发配置流程，及具体的开发流程示例，及版本发布说明

# 后台程序架构

## 1.1 开发所用框架

后台程序开发，基于公司开发的xap框架（郭云嵩早期开发，与iih现有xap框架不同），该框架目的是实现与业务隔离，框架集成了spring框架、doma框架（ORM框架，实现了代码sql分离）、shiro框架

## 1.2 业务工程组成，工程一览图

如图1所示

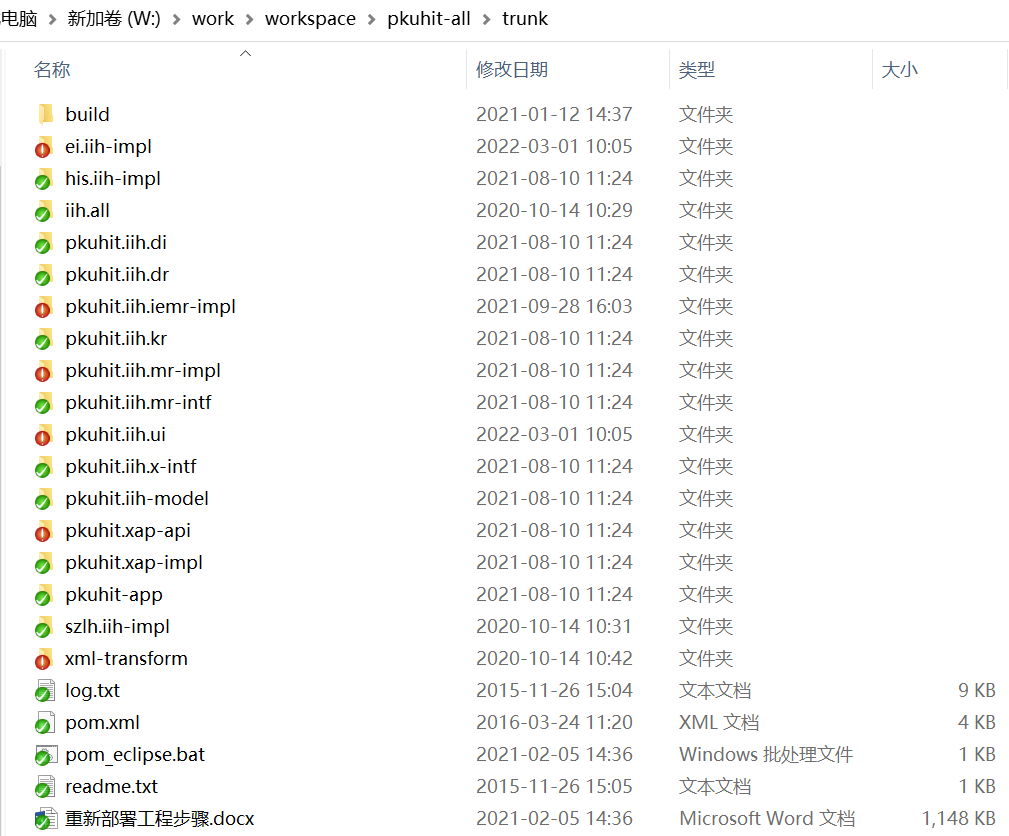


图1

ei.iih-impl：对外接口，多数据源配置

his.iih-impl ：对外接口，主要是同步his、cis数据库数据，28版本以后，已经都迁移到ei工程

pkuhit-app：程序入口工程

pkuhit.iih-model：业务对象，封装数据前后端交互

pkuhit.iih.di：诊断录入

pkuhit.iih.dr：病案检索

pkuhit.iih.iemr-impl：该工程是历史代码，目前不用

pkuhit.iih.kr：该工程是历史代码，目前不用

pkuhit.iih.mr-impl 病历书写、质控，大部分病历业务都在这块实现

pkuhit.iih.mr-intf pkuhit.iih.mr-impl，对应的业务类

pkuhit.iih.ui 前端所有业务代码，不含封装的extjs框架及组件，

pkuhit.iih.x-intf his.iih-impl对应的接口

pkuhit.xap-api pkuhit.xap-impl对应的接口

pkuhit.xap-impl 用户、角色、菜单等后台管理

xml-transform：互联互通转换工程

## 1.3代码引入eclipse

请参考《重新部署工程步骤.docx》，文档路径于代码路径同级



## 1.4单个工程结构

每个工程结构都大体一样，以pkuhit.iih.mr-impl为例，图2所示

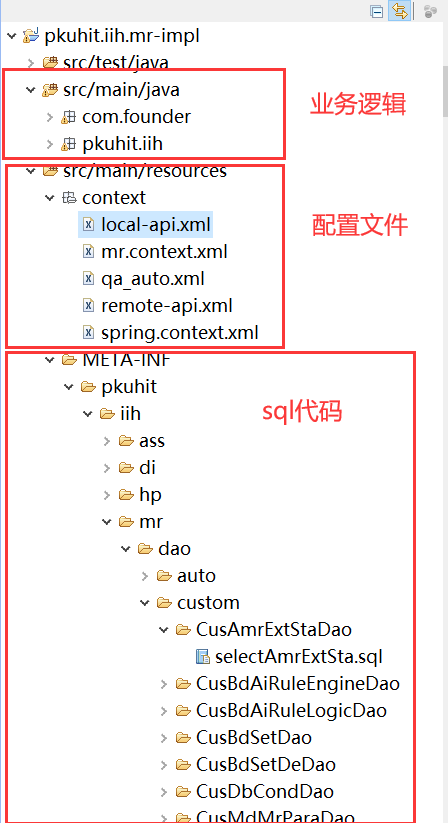


图2

## 1.5后台代码开发涉及到几个部分

主要涉及到：service、serviceImpl、dao、sql 、model、entity

Service定义前端请求所调用的接口

serviceImpl具体业务接口方法的实现类

dao定义sql方法的接口

sql具体查询数据库的sql文件

model业务对象，封装数据前后端交互

entity对应数据库二维表

目前框架通过工具可以自动生成通用的entity、sql（增删改查）、dao（增删改查）等文件，不需要手动生成，业务只需编写多表查询或者多条件查询等相关sql，工具截图，如图3所示



图3

## 1.6后台代码的配置文件

pkuhit.iih.mr.context.xml配置文件，如图4所示

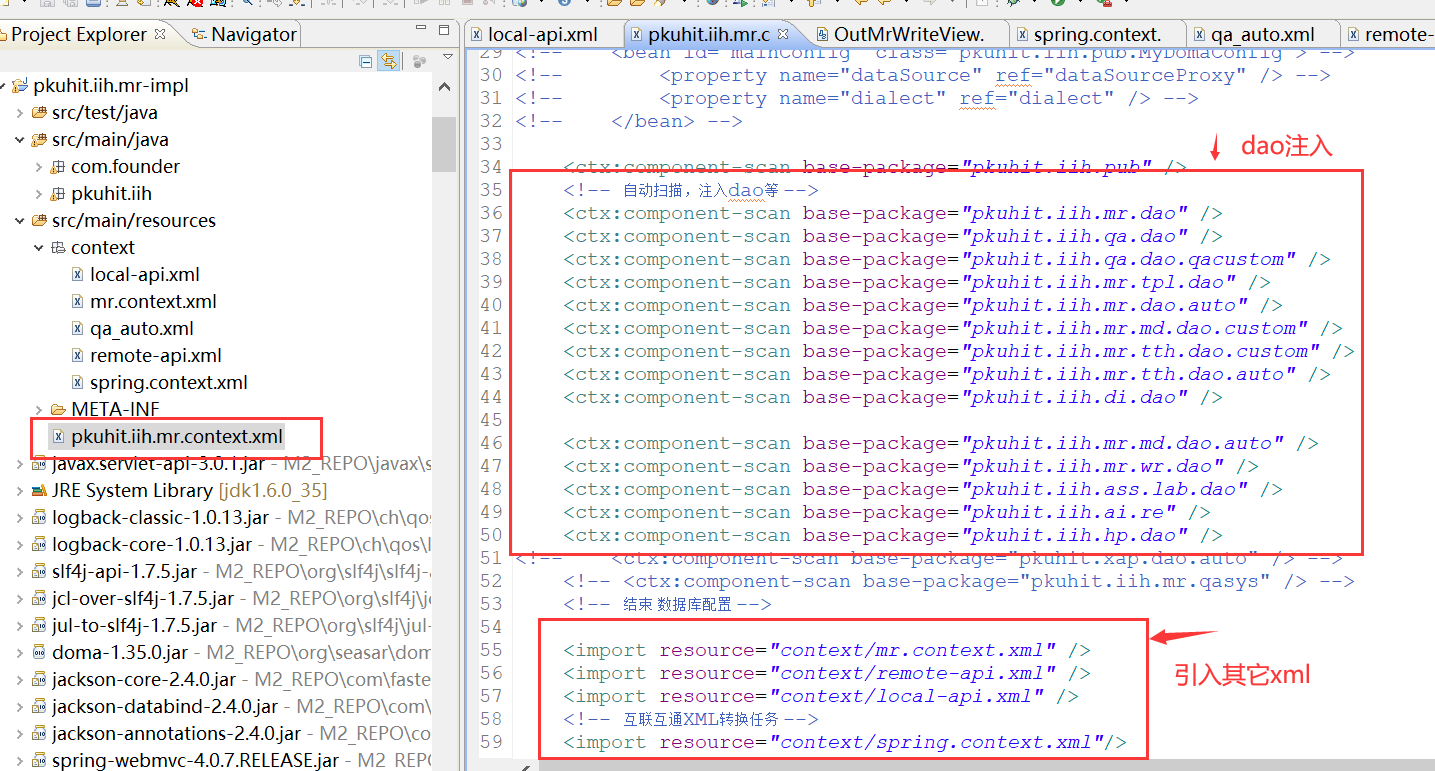


图4

local-api.xml配置文件主要配置本工程的service服务，如图5所示

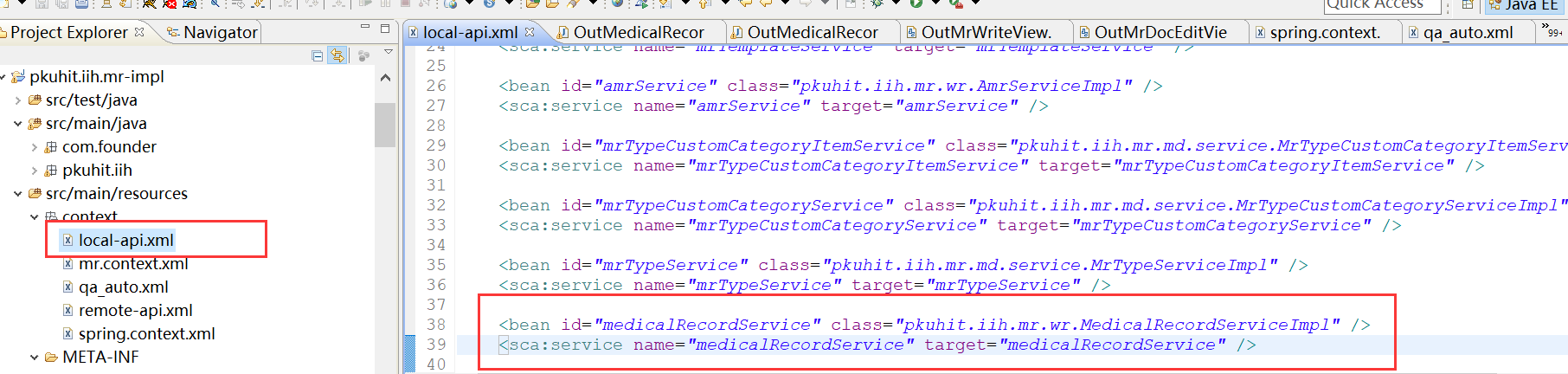


图5

remote-api.xml，如图6所示

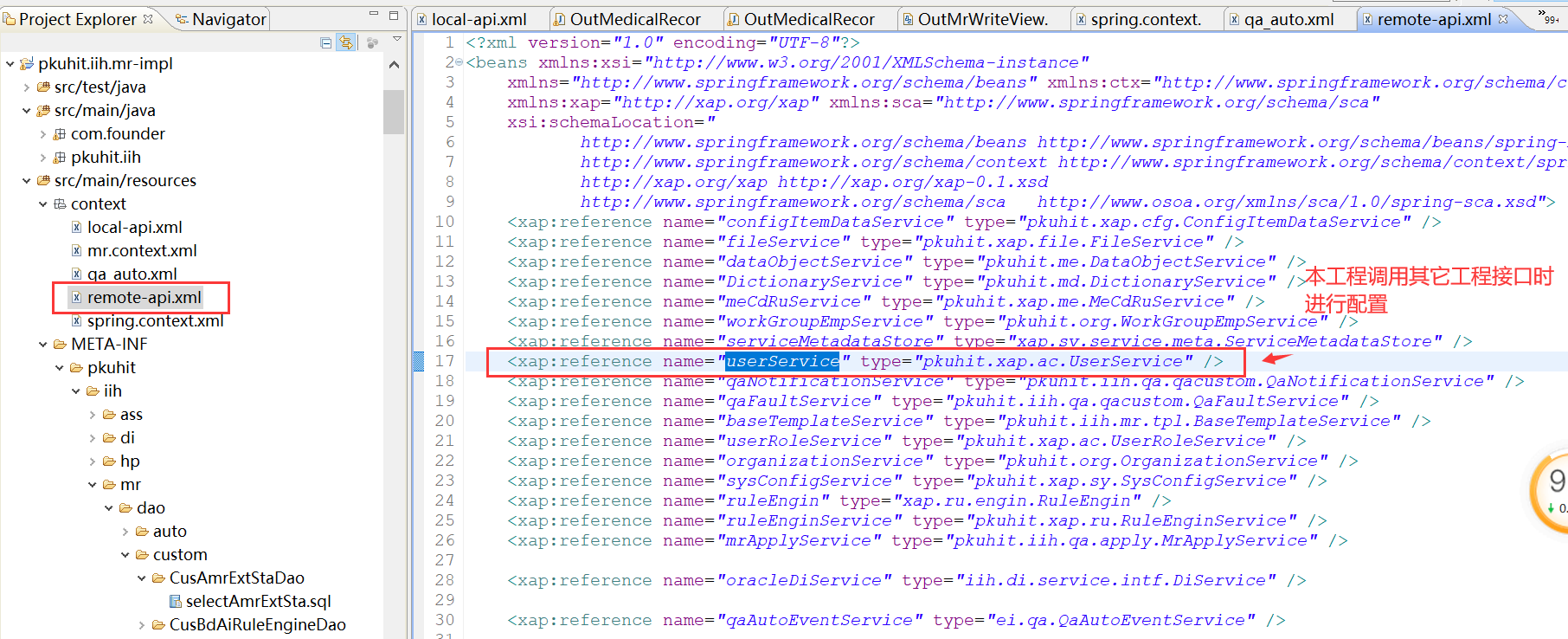


图6

mr.context.xml配置aop信息；spring.context.xml配置定时任务信息，如图7所示



图7

病历数据库配置信息

pkuhit.xap-impl/src/main/resources/context/pkuhit.database.context.xml ，如图8所示



图8

多数据源配置信息，可配置多个数据库的访问地址

ei.iih-impl/src/main/resources/cfg.properties，如图9所示：

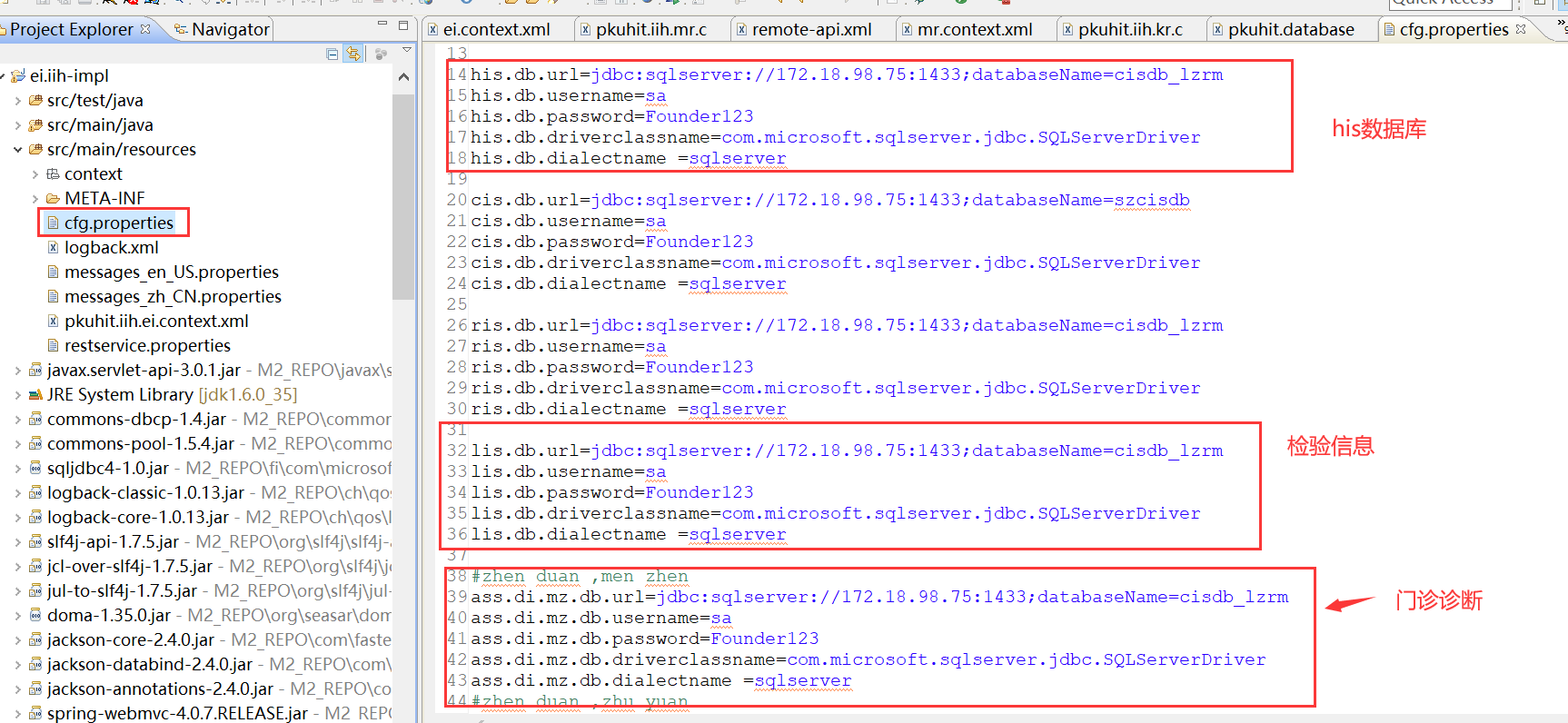


图9

# 前端程序架构

## 2.1前端Ajax框架

前端框架由extjs4.2及轻量级的封装组成，如图10所示

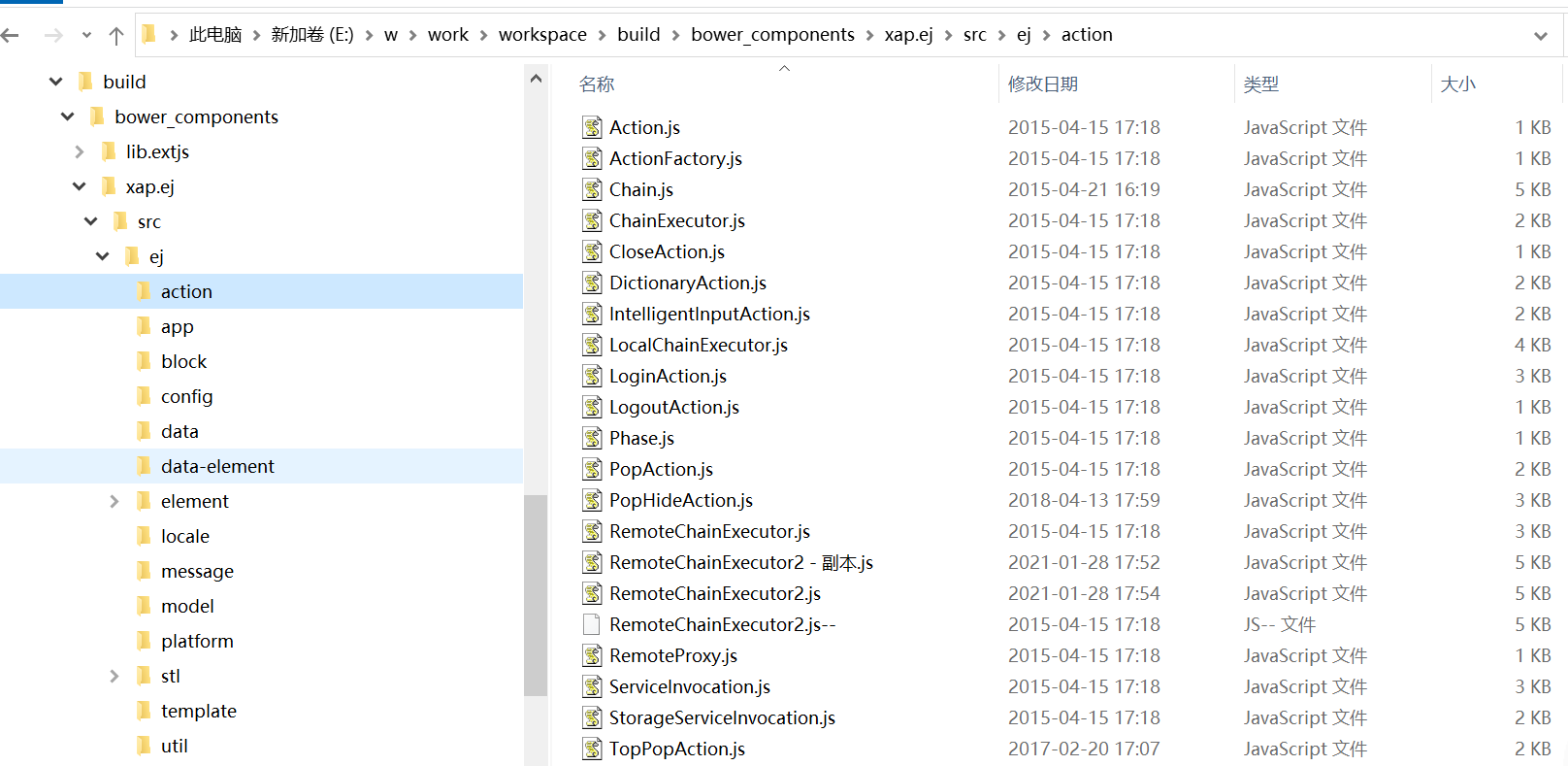


图10

Lib.extjs是extjs原生代码

Xap.ej是病历系统轻量级的封装，包括按钮、布局、ajax请求等封装

## 2.2前端业务代码结构

如图11所示

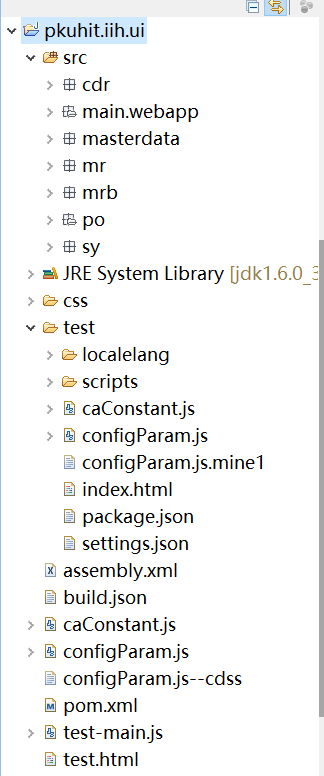


图11

Src：功能模块目录，cdr互联互通相关功能；mr病历书写、病历模板、病案检索、病历质控等相关功能；mrb病案管理相关功能；po病历系统及公共模块相关功能；sy系统管理相关功能

Css：css样式

Test：发布版本时使用，index.html程序入口；

configParam.js，前端参数配置文件，

package.json,版本信息；

settings.json访问后台ip地址配置信息

Test.html:开发环境程序入口

前端所有菜单配置文件，如图12所示：

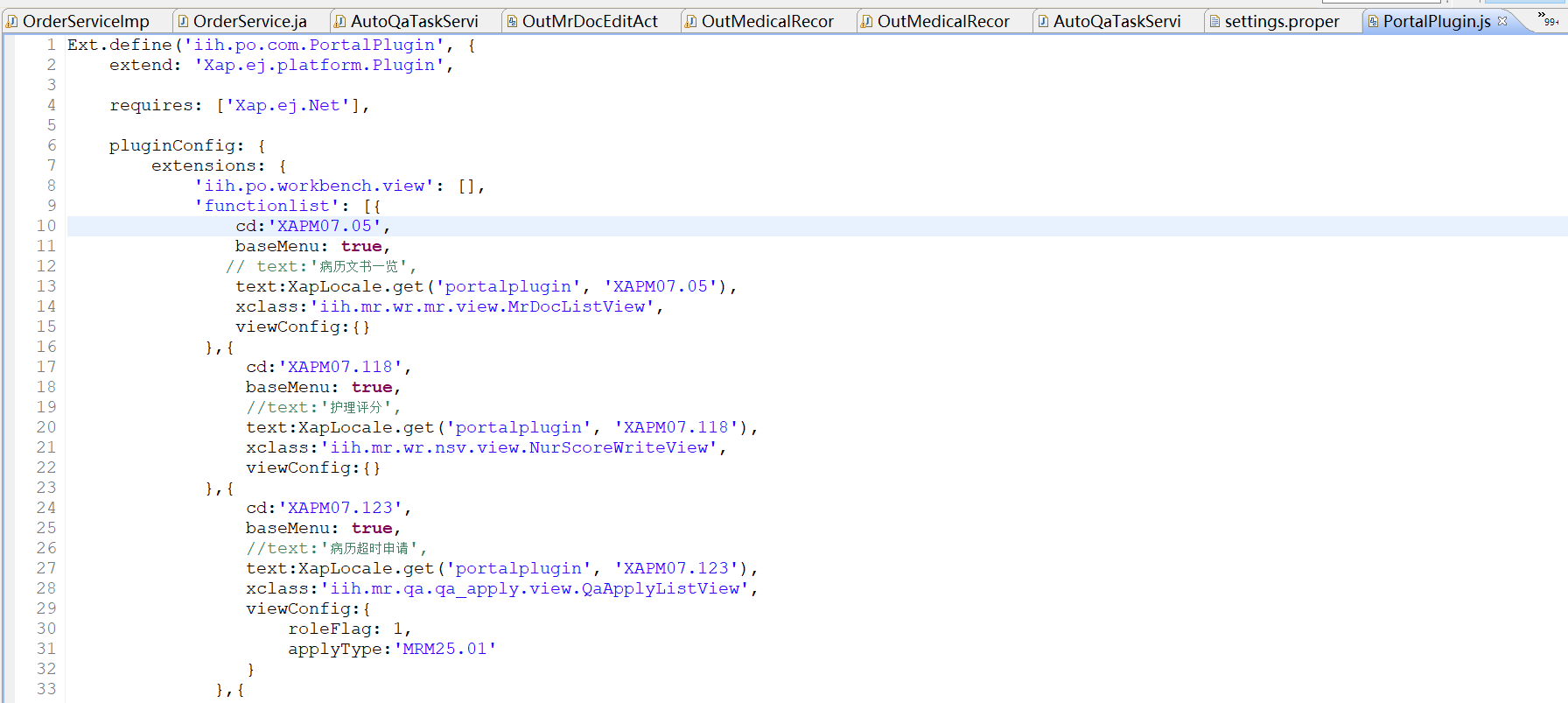


图12

# 代码流程

接下来，以一个页面功能为例，介绍下开发流程，病历书写界面如图13所示



图13

## 3.1前端代码-布局

一个功能涉及到的js一般包含三块view.js、block.js、action.js:

View.js 负责布局及页面事件与方法的绑定，图13中布局对应图14中代码：



图14

图15展示了，该页面用到的处理业务的方法（action）



图15

图16展示了，chain与方法的关联，一个chain可以关联多个action，一个事件只与一个chain绑定



图16

图17展示了，chain与事件的绑定，以一个文书树的文书【单击打开】为例：布局左侧left（iih.mr.wr.omr.block.OutMrDocTreeBlock）中的xtype:'xaptree'（文书树）监听了itemclick事件，当单击一篇文书的节点时，会调用chain：treeNodeChange，根据上边图16、图15的配置，找到对应的action去处理业务，请求后台，返回文书结果展现到右侧right区域



图17

Block.js 负责业务界面布局及展示，下图18，展示了门诊病历文书树、病历模板树以及按钮



图18

Action.js负责处理页面逻辑，比如查询文书数据、保存数据、封装数据向后台发送请求，下图19，展示了打开文书的action处理，



图19

由于该action逻辑中首先要判断右侧是否有打开的文书，文书是否未保存，或者用户取消打开新的文书等业务逻辑，所以真正打开文书的操作通过调用

var initChain = right.getActionChain('init'); initChain.execute();

图20为该chain对应的action方法



图20

代码实现，图21、22、23、24展示了，打开文书请求后台并返回数据：

图21获取请求后台所需的参数等信息

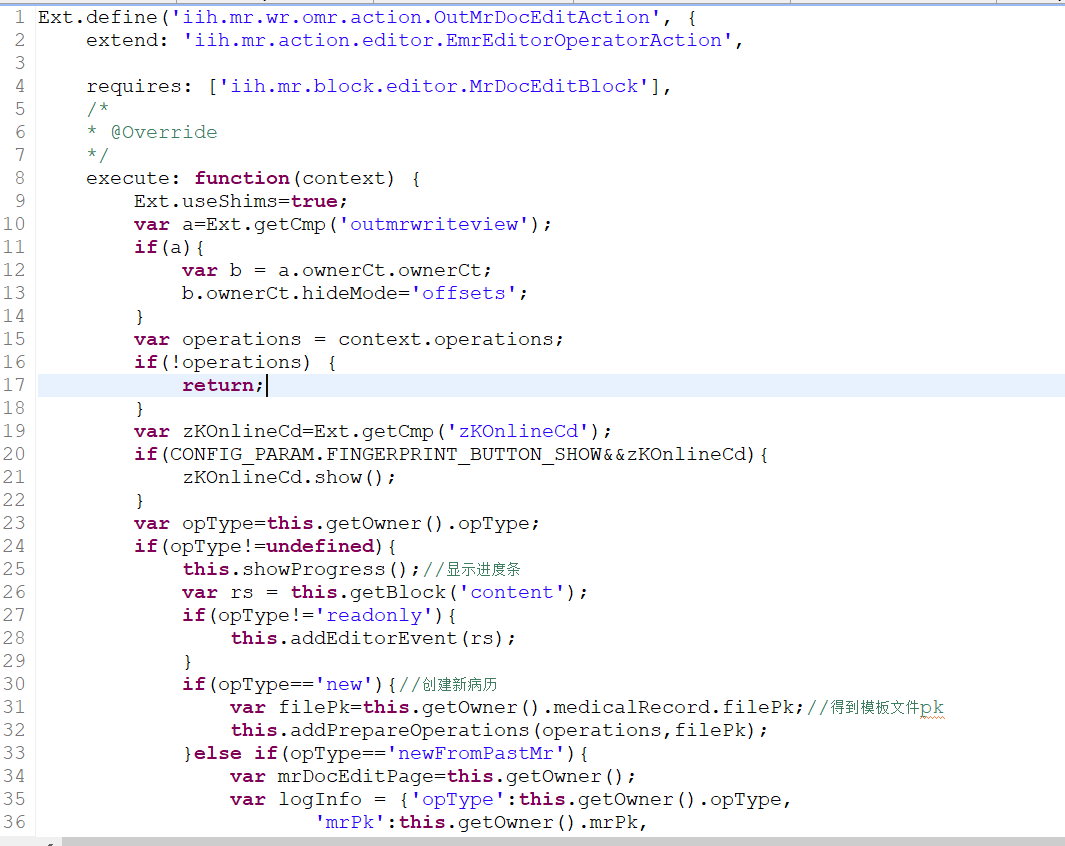


图21

图22，调用请求后台方法



图22

由于我们代码是在封装后的框架上实现的，所以调用后台的ajax请求也被封装了，具体调用的框架代码主要片段如图23

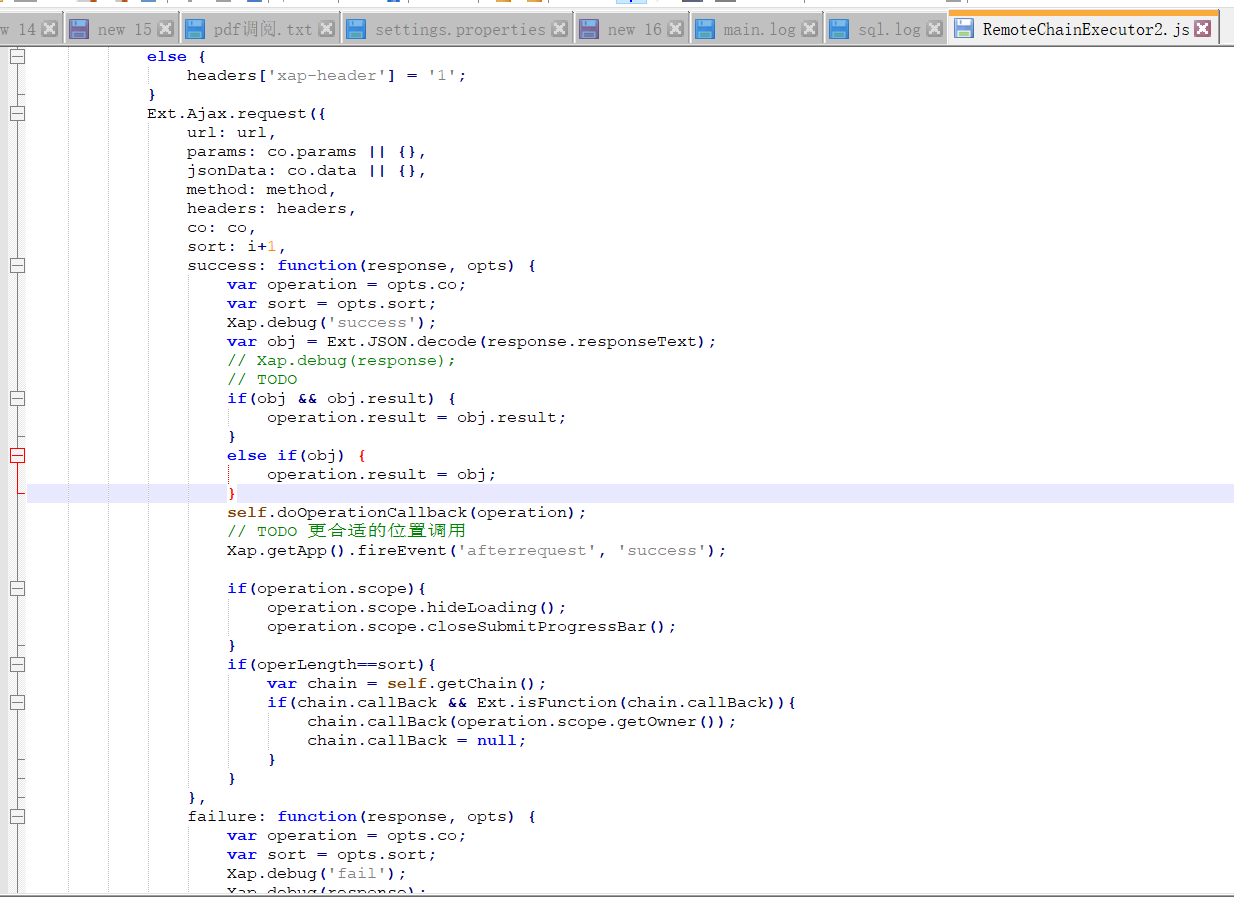


图23

图24请求结果返回值处理



图24

## 3.2后台代码

接着前端打开文书的请求，查看后台调用流程，通过前端调试窗口，可以查看到调用后台请求url，如图25所示

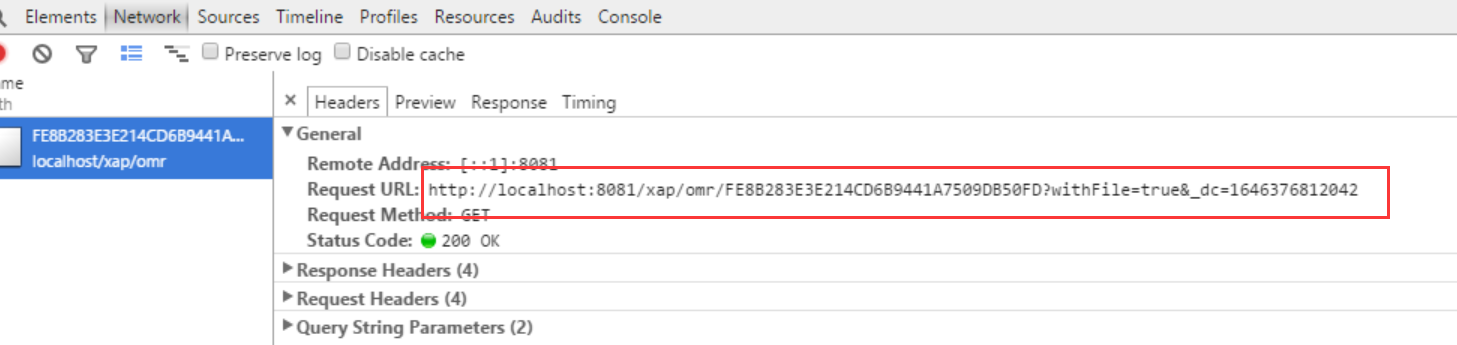


图25

找到后台service接口 ，如图26所示

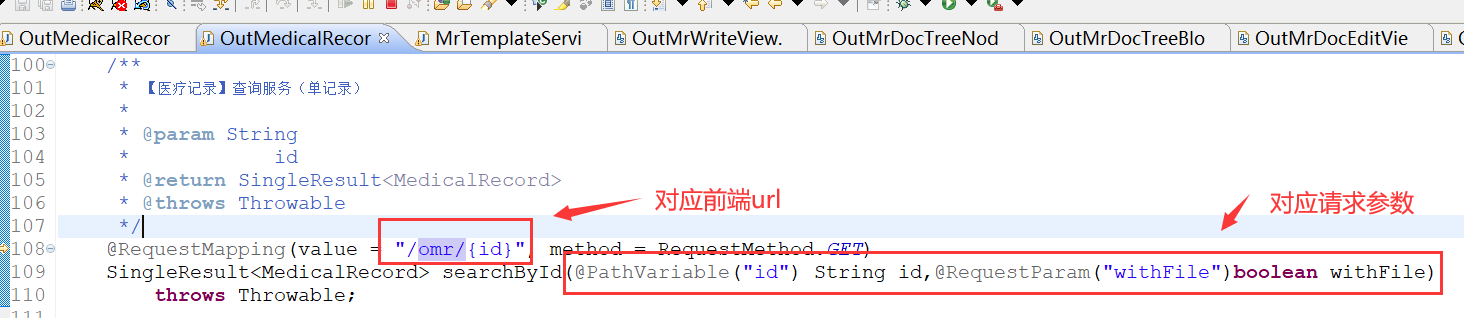


图26

对应的后台接口实现类，如图27所示

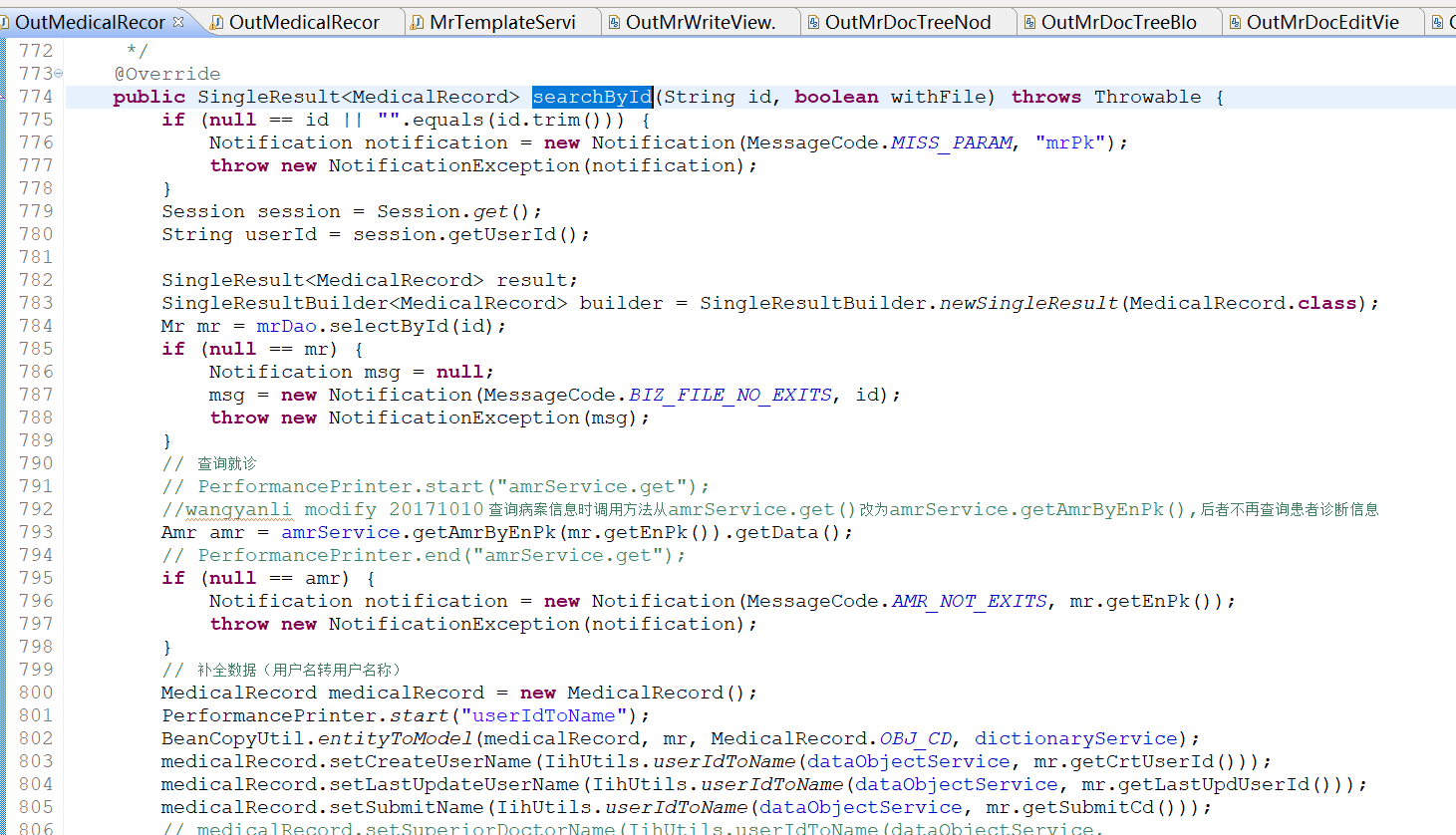


图27

该方法中通过前端获取的文书主键查询文书信息Mr mr = mrDao.selectById(id)；

SelectById.sql是用上边介绍的工具自动生成的，用mr表的主键进行查询，返回信息到MR实体中（entity），sql如图28所示



图28

然后通过Amr amr = amrService.getAmrByEnPk(mr.getEnPk()).getData();根据文书中的患者enpk查询到患者信息，供后续封装前端返回的model（MedicalRecord）使用

Amrservice通过注解方式引入，如图29所示

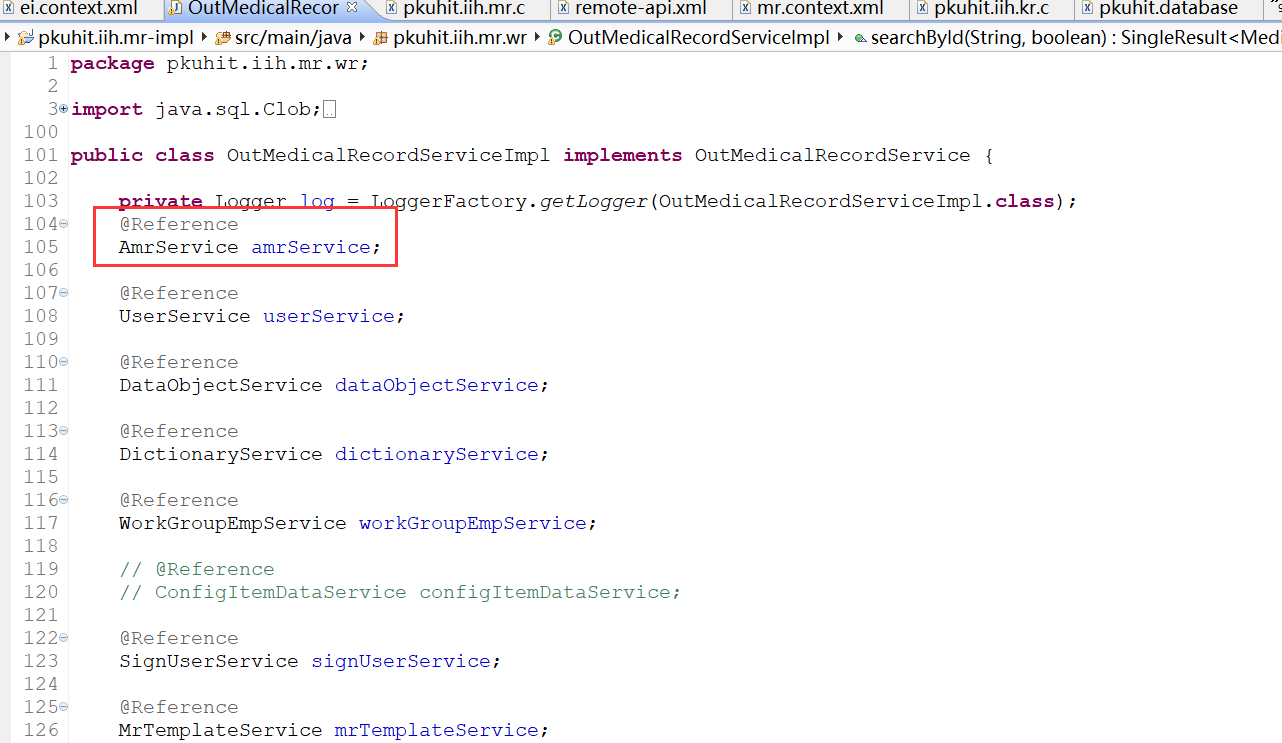


图29

以下是model：MedicalRecord.java代码截图，如图30所示

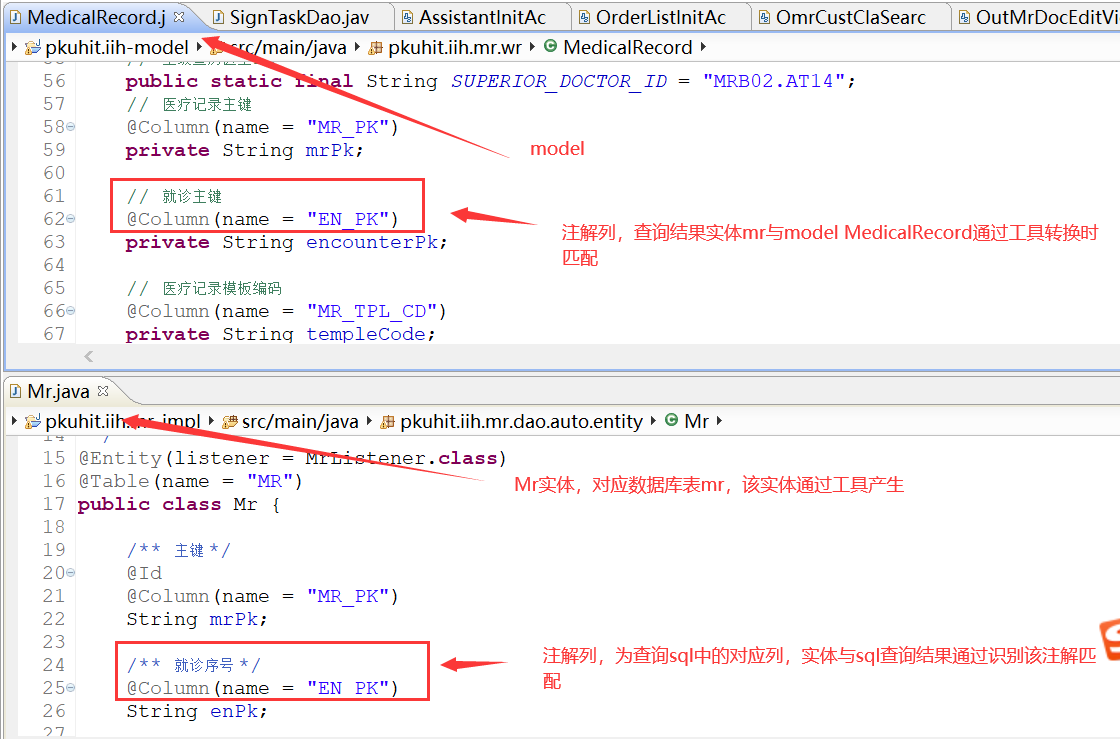


图30

工具转换类，将实体mr转换为model medicalRecord，供返回结果使用，转换过程中，如果有字典数据，例如科室编码，可以自动匹配到科室名称，赋值给model对应字段

BeanCopyUtil.entityToModel(medicalRecord, mr, MedicalRecord.OBJ\_CD, dictionaryService);

该请求文书的方法包含了，查询文书信息、患者信息、模板信息、可编辑、可审签等信息存放到model Medicalrecord中返回给前端，如图31所示：

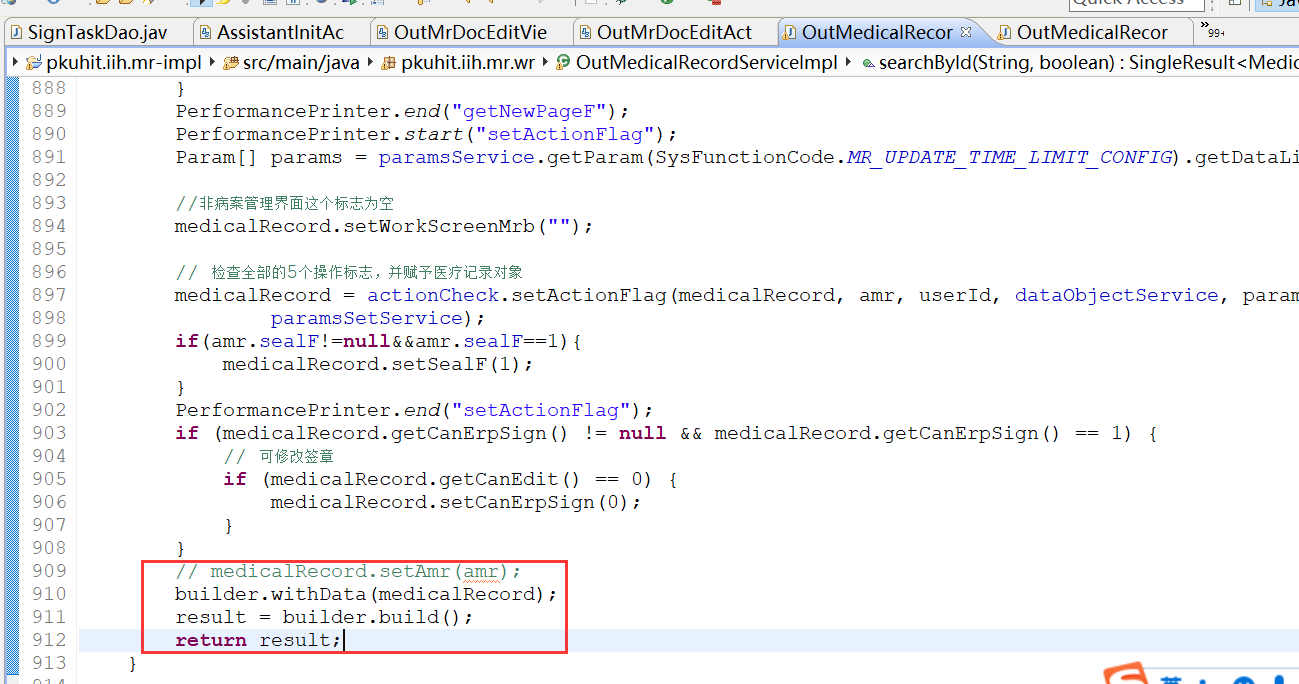


图31

# 版本发布过程

IEMR2.0系统发布版由服务端程序、客户端程序和客户端升级patch包三部分组成，发布发布过程为三个步骤，分别是服务端版本发布、客户端版本发布、客户端升级patch包生成。