一、就诊人管理

1、需求

预约下单需要选择就诊人，因此我们要实现就诊人管理，就诊人管理其实就是要实现一个完整的增删改查

2、就诊人管理接口

（1）引入依赖

```
<dependencies>
    <dependency>
        <groupId>com.atguigu</groupId>
        <artifactId>service_cmn_client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
</dependencies>
```

（2）添加PatientMapper

```
public interface PatientMapper extends BaseMapper<Patient> {

}
```

（3）添加PatientService

```
public interface PatientService extends IService<Patient> {
    //获取就诊人列表
    List<Patient> findAllUserId(Long userId);
    //根据id获取就诊人信息
    Patient getPatientId(Long id);
}
```

(4) 添加PatientServiceImpl

```
@Service
public class PatientServiceImpl extends
        ServiceImpl<PatientMapper, Patient> implements PatientService {

    @Autowired
    private DictFeignClient dictFeignClient;

    //获取就诊人列表
    @Override
    public List<Patient> findAllUserId(Long userId) {
        //根据userid查询所有就诊人信息列表
        QueryWrapper<Patient> wrapper = new QueryWrapper<>();
        wrapper.eq("user_id",userId);
        List<Patient> patientList = baseMapper.selectList(wrapper);
        //通过远程调用，得到编码对应具体内容，查询数据字典表内容
        patientList.stream().forEach(item -> {
            //其他参数封装
            this.packPatient(item);
        });
        return patientList;
    }

    @Override
    public Patient getPatientId(Long id) {
        return this.packPatient(baseMapper.selectById(id));
    }

    //Patient对象里面其他参数封装
    private Patient packPatient(Patient patient) {
        //根据证件类型编码，获取证件类型具体指
        String certificatesTypeString =
                dictFeignClient.getName(DictEnum.CERTIFICATES_TYPE.getDictCode(), pat
        //联系人证件类型
       // String contactsCertificatesTypeString =dictFeignClient.getName(DictEnum.CER
        //省
        String provinceString = dictFeignClient.getName(patient.getProvinceCode());
        //市
        String cityString = dictFeignClient.getName(patient.getCityCode());
```

```
        //区
        String districtString = dictFeignClient.getName(patient.getDistrictCode());
        patient.getParam().put("certificatesTypeString", certificatesTypeString);
       // patient.getParam().put("contactsCertificatesTypeString", contactsCertificat
        patient.getParam().put("provinceString", provinceString);
        patient.getParam().put("cityString", cityString);
        patient.getParam().put("districtString", districtString);
        patient.getParam().put("fullAddress", provinceString + cityString + districtS
        return patient;
    }
}
```

（5）添加Patientcontroller

```
//就诊人管理接口
@RestController
@RequestMapping("/api/user/patient")
public class PatientApiController {
    @Autowired
    private PatientService patientService;
    //获取就诊人列表
    @GetMapping("auth/findAll")
    public R findAll(HttpServletRequest request) {
        //获取当前登录用户id
        Long userId = AuthContextHolder.getUserId(request);
        List<Patient> list = patientService.findAllUserId(userId);
        return R.ok().data("list",list);
    }
    //添加就诊人
    @PostMapping("auth/save")
    public R savePatient(@RequestBody Patient patient, HttpServletRequest request) {
        //获取当前登录用户id
        Long userId = AuthContextHolder.getUserId(request);
        patient.setUserId(userId);
        patientService.save(patient);
        return R.ok();
    }
    //根据id获取就诊人信息
    @GetMapping("auth/get/{id}")
    public R getPatient(@PathVariable Long id) {
```

```
        Patient patient = patientService.getPatientId(id);
        return R.ok().data("patient",patient);
    }
    //修改就诊人
    @PostMapping("auth/update")
    public R updatePatient(@RequestBody Patient patient) {
        patientService.updateById(patient);
        return R.ok();
    }
    //删除就诊人
    @DeleteMapping("auth/remove/{id}")
    public R removePatient(@PathVariable Long id) {
        patientService.removeById(id);
        return R.ok();
    }
}
```