云归档动工

**数据采集**

1.

注册患者信息

四种入参：患者信息，就诊信息，检查信息，检查结果信息（对应四张表）

因为Visit（病人就诊记录）表中有PatientIndex中的PatientMasterID（病人主索引ID），而ObservationRequest表中有Visit表中的ObservationUID（观察唯一序号），而ObservationResult（观察结果表）中又有ObservationRequest表中的ObservationUID（关联观察请求序列号），所以这四张表的存在关系，就是1234如下排列。

（1）PatientIndex：病人主索引信息

（2）Visit：病人就诊记录

（3）ObservationRequest：观察请求记录

（4）ObservationResult：观察结果表

第五种为最后插入数据的表。

（5）LogClinicDataOperation临床数据操作日志

namespace eWorld.Core.DTO

{

/// <summary>

/// 注册检查信息请求参数

/// </summary>

public class DataArchivingQuery

{

/// <summary>

/// 患者信息

/// </summary>

public PatientRegisterQuery Patient { get; set; }

/// <summary>

/// 就诊信息

/// </summary>

public PatientVisitQuery Visit { get; set; }

/// <summary>

/// 检查信息

/// </summary>

public ObservationRequestAddQuery ObservationRequest { get; set; }

/// <summary>

/// 检查结果信息

/// </summary>

public List<ObservationResultInfo> ObservationResultList { get; set; }

}

}

最终注册的时候，只调用了 PatientIndex：病人主索引信息 和 Visit：病人就诊记录

/// <summary>

/// 患者注册

/// </summary>

/// <param name="query"></param>

/// <returns></returns>

private ModelResult<string> PatientRegister(PatientRegisterQuery query, PatientVisitQuery visitquery)

{

这一串的注册，感觉应该是同生共死，用事务。

一连串，还是对四个表的插入工作。但是第一个，PatientIndex：病人主索引信息表，涉及了insert或者update的操作。

而且，这里涉及后一个方法的参数用到前一个刚生成的参数。

var patientregister = PatientRegister(query.Patient, query.Visit);

if (patientregister.code == ResultCodeEnum.Success)//患者信息注册成功方可写入检查信息

{

\_log.Info("患者信息注册成功");

string patientmasterid = patientregister.Result;

query.ObservationRequest.PatientMasterID = patientmasterid;

var uploadobservationrequest = UploadObservationRequest(query.ObservationRequest);

if (uploadobservationrequest.code == ResultCodeEnum.Success)

{

\_log.Info("患者检查信息注册成功");

query.Visit.PatientMasterID = patientmasterid;

query.Visit.VisitUID = uploadobservationrequest.Result.VisitUID.Value;

var uploadpatientvisit = UploadPatientVisit(query.Visit);

if (uploadpatientvisit.code == ResultCodeEnum.Success)

{

\_log.Info("患者就诊信息注册成功" );

}

else

{

result.msg = "检查信息上传失败，患者就诊信息注册失败";

\_log.Info("检查信息上传失败,患者就诊信息注册失败，：" + uploadpatientvisit.msg);

}

observationuid = uploadobservationrequest.Result.ObservationUID.Value;

foreach (var observationresult in query.ObservationResultList)

{

observationresult.ObservationUID = observationuid;

}

var uploadobservationresult = UploadObservationResult(query.ObservationResultList);

if (uploadobservationresult.code == ResultCodeEnum.Success)

{

\_log.Info("患者纠缠结果信息注册成功");

try

{

var log = new LogClinicDataOperation();

log.BusinessType = "Exam";

log.BusinessUID = observationuid;

log.ClassCode = "Exam";

log.ClientApplication = "";

log.ClientHostName = "";

log.ClientKind = 1;

log.CreateDate = DateTime.Now;

log.OperationDesc = "采集检查信息入库";

log.OperationType = 0;//新增

log.TypeCode = "Exam";

log.UserKind = 2;

\_logClinicDataOperationRepository.Insert(log);

}

catch (Exception e)

{

\_log.Error("采集检查信息入库，保存采集日志日志出现异常", e);

}

result.code = ResultCodeEnum.Success;

result.msg = "OK";

result.Result = observationuid;

}

else

{

//检查结果信息入库失败，删除检查信息、就诊信息

\_log.Info("采集检查信息入库,检查结果信息保存失败，：" + uploadobservationresult.msg);

result.msg = "检查信息上传失败，检查结果信息保存失败";

}

}

else

{

\_log.Info("采集检查信息入库,检查信息保存失败，：" + uploadobservationrequest.msg);

result.msg = "检查信息上传失败，检查信息保存失败";

}

}

else

{

result.msg = "检查信息上传失败，患者信息注册失败";

\_log.Info("检查信息上传失败,患者信息注册失败，：" + patientregister.msg);

}

最后还有第五个方法，采集检查信息入库。

涉及LogClinicDataOperation临床数据操作日志 表

\_log.Info("患者检查结果信息注册成功");

try

{

var log = new LogClinicDataOperation();

log.BusinessType = "Exam";

log.BusinessUID = observationuid;

log.ClassCode = "Exam";

log.ClientApplication = "";

log.ClientHostName = "";

log.ClientKind = 1;

log.CreateDate = DateTime.Now;

log.OperationDesc = "采集检查信息入库";

log.OperationType = 0;//新增

log.TypeCode = "Exam";

log.UserKind = 2;

\_logClinicDataOperationRepository.Insert(log);

}

数据采集接口：

var returnModel = \_imageArchiveClient.Invoke(i => i.SaveExamDataInfo(query));

第三方用的：

var saveRes = \_imageArchiveClient.Invoke(t => t.SavePatientInfo(param));

用数据采集的，现在是把就诊信息也放在了里面，我要把它隔离出来。

现在是五张表，

拆分之后：

（1）操作主索引表

（2）操作就诊信息，检查信息，检查结果信息，临床数据操作日志暂时没提。

对应的，曹兵那边也会拆成两张表。

改造后，需要第一步完成，才能去操作第二步。

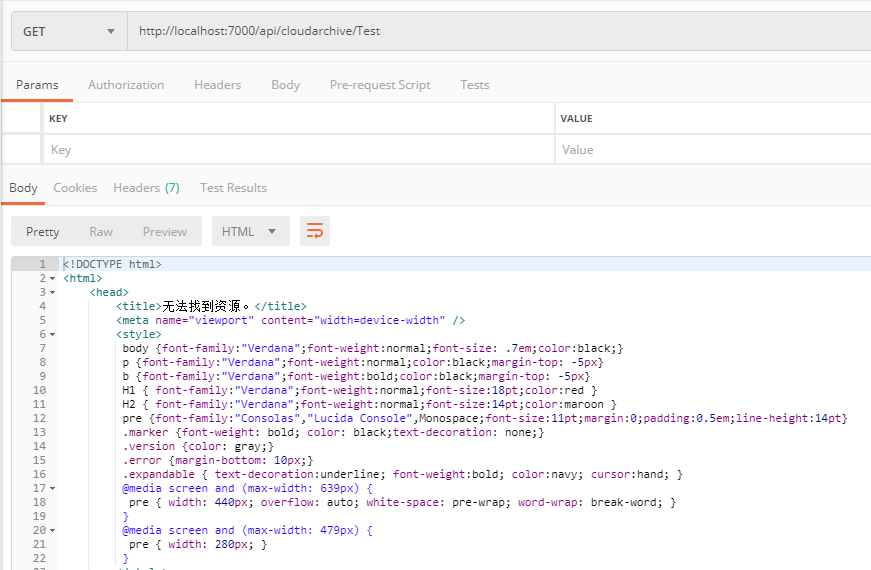
这两个接口是给采集服务用的，我要做的，患者信息注册和数据采集的这个接口，是有两个接口，一个是检查数据的，一个是患者信息的。只要走这两个接口，就可以把检查信息给传上来了。

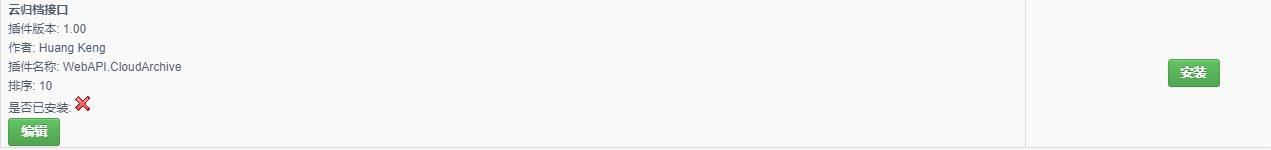
以前的采集服务的接口，也得重新整一遍，包括心电的，很老的，没优化过的，给喻永红用的，都要重新整。

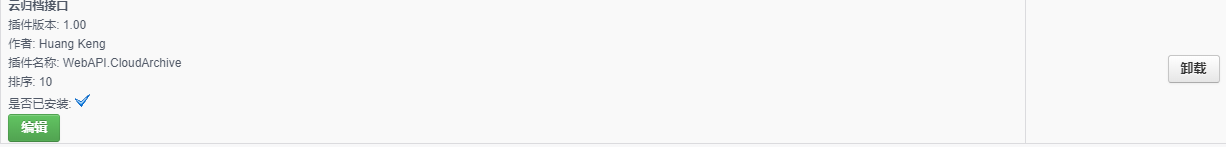
我需要先搞清楚流程。

2.

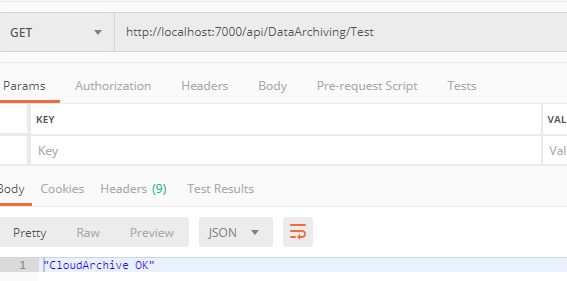
未安装插件之前







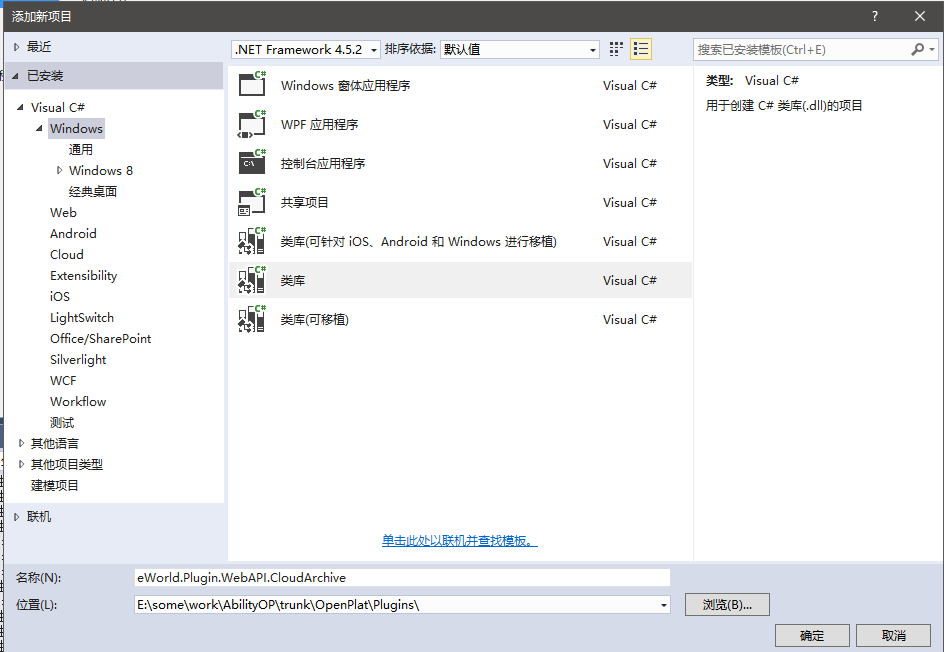
安装之后ok了。



3.

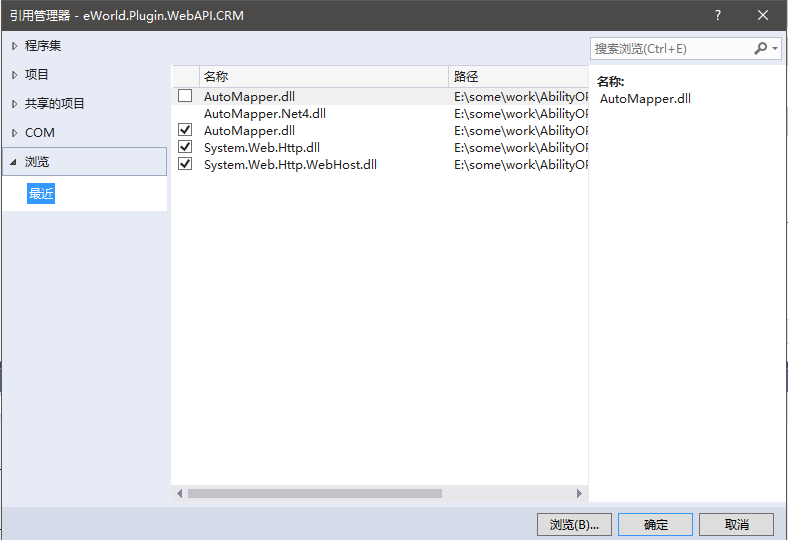
再走一次

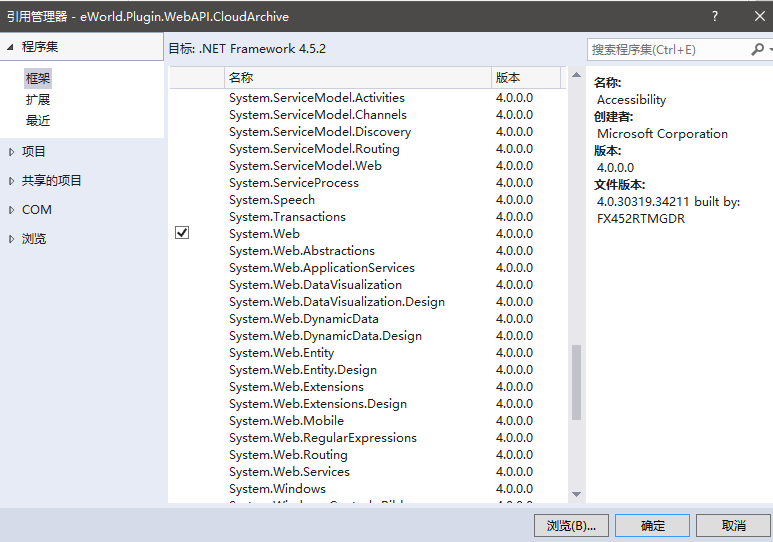
（1）

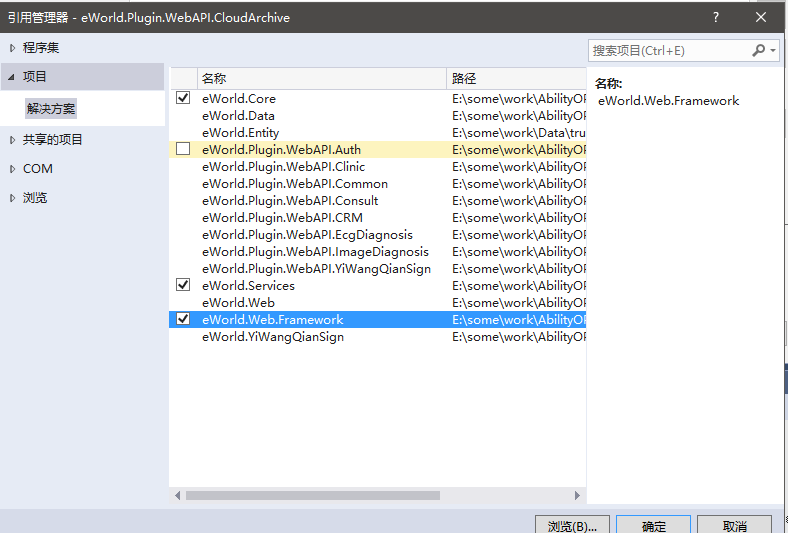


（2）

添加引用

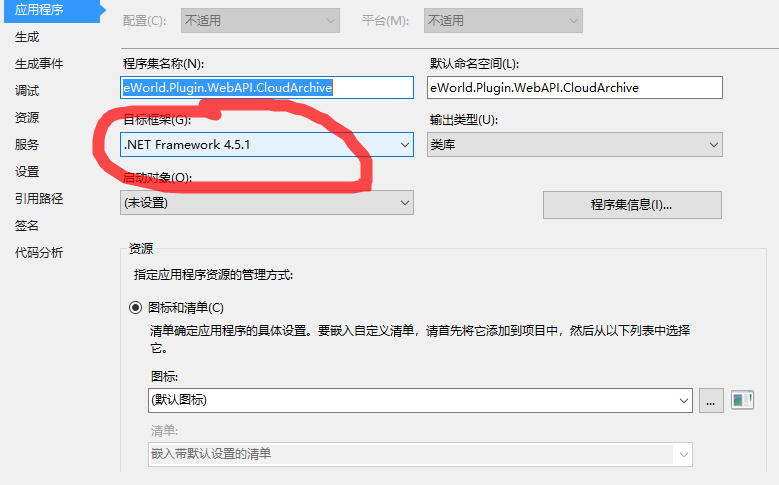




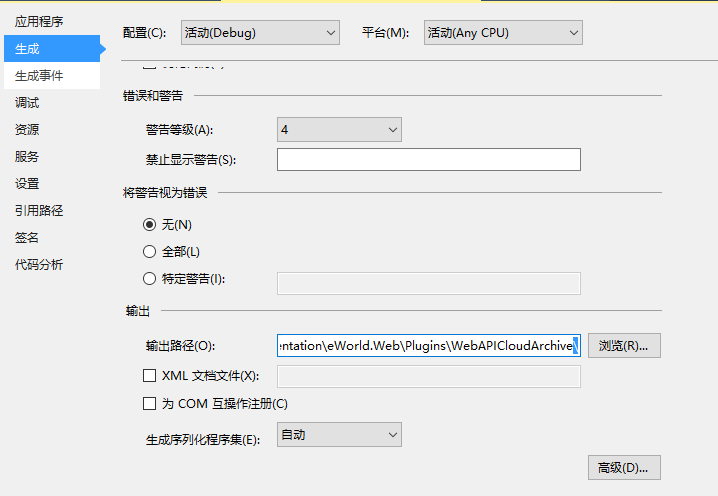


（3）

4.5.1



生成：



（4）

4.

上传文件：

var saveExamRes = \_imageArchiveClient.Invoke(t => t.UploadDocument(info));

第三方接口和Foreign控制器

有一个字节流的参数

/// <summary>

/// 文件的字节流。不加入Token验证。

/// </summary>

public byte[] FileSize { get; set; }

5.

获取患者检查dicomstudy信息

/// <summary>

/// 获取患者检查dicomstudy信息

/// </summary>

/// <param name="query"></param>

/// <returns></returns>

[Route("v2/dataarchiving/get\_exam\_dicomstudy")]

[HttpPost]

public ModelReturnResult<PatientExamDicomStudyDTO> GetExamDicomStudy([FromBody]PatientExamDicomStudyQuery query)

{

WCF：

var returnModel = \_imageArchiveClient.Invoke(i => i.GetExamDicomStudy(query.ToQuery()));

核心代码：

先是在Document（文档记录）中查询该BusinessID对应的结果，如果存在，再去DicomStudy（影像的检查信息）表中根据FileUID关联查询到所有的病人信息。

var documentInfo = \_repository.Table.Where(d => d.BusinessID == query.BusinessID && d.TypeCode == "ExamImage" && d.DeleteFlag == false).FirstOrDefault();

if (documentInfo != null)

{

var dicomStudy = \_dicomStudyRepository.Table.Where(d => d.FileUID == documentInfo.FileUID).FirstOrDefault();

if (dicomStudy != null)

{

model.AccessionNumber = dicomStudy.AccessionNumber;

model.ModalityInStudy = dicomStudy.ModalityInStudy;

model.PatientID = dicomStudy.PatientID;

model.PatientName = dicomStudy.PatientName;

model.StudyDateTime = dicomStudy.StudyDateTime;

model.StudyID = dicomStudy.StudyID;

model.StudyInstanceUID = dicomStudy.StudyInstanceUID;

}

}

Guid businessid = new Guid();

if (Guid.TryParse(query.BusinessID, out businessid))

{

此外，在上面已经走了两个表之后，还要去ObservationRequest（观察请求记录）中查询该ObservationUID（也是和businessid关联）获取该观察结果，然后根据其中的PatientID，PatientMasterID，PIDAssigningAuthority，三个条件又一次关联Document（文档记录）表，去获取病人信息，进行组装。

var exam = \_observationRequestRepository.Table.Where(i => i.ObservationUID == businessid && i.DeleteFlag == false).FirstOrDefault();

这里，一旦ObservationRequest（观察请求记录）中没有数据，则直接跳到下面的“获取检查信息失败”的结果。

if (exam != null)

{

model.OrganizationID = exam.OrganizationID;

model.OrganizationName = exam.OrganizationName;

var patientInfo = \_repository.GetTable<PatientIndex>().Where(p => p.PatientID == exam.PatientID && p.PatientMasterID == exam.PatientMasterID && p.PIDAssigningAuthority == exam.PIDAssigningAuthority).FirstOrDefault();

if (patientInfo != null)

{

model.Name = patientInfo.Name;

}

modelResult.code = ResultCodeEnum.Success;

modelResult.msg = "OK";

modelResult.Result = model;

}

else

{

modelResult.msg = "根据业务唯一号获取检查信息失败";

}

}

入参：

{

"BusinessID":"00007b20-70b7-40f0-9913-13880fffafcd"

}

返回：

{

"code": 0,

"msg": "成功",

"data": {

"PatientName": null,

"Name": "赵治学",

"PatientID": null,

"StudyDateTime": null,

"AccessionNumber": null,

"ModalityInStudy": null,

"StudyID": null,

"StudyInstanceUID": null,

"OrganizationID": "123308244720006231",

"OrganizationName": "音坑乡卫生院"

}

}

6.

更新检查影像状态

var returnModel = \_imageArchiveClient.Invoke(i => i.ModifyExamImageState(query.ToQuery()));

针对 观察请求记录：ObservationRequest 表的操作。

var obs = \_observationRequestRepository.Table.Where(t => t.ObservationUID == query.ObservationUID).FirstOrDefault();

if (obs == null)

{

throw new Exception("查找不到ObservationRequest记录，ObservationUID=" + query.ObservationUID.ToString());

}

double diskSpace = 0;

var observationUID = query.ObservationUID.ToString();

var documentList = \_documentRepository.Table.Where(t => t.BusinessID == observationUID && t.DeleteFlag == false).ToList();

if (documentList != null && documentList.Count > 0)

{

foreach (var document in documentList)

{

if (document.FileSize != null)

{

diskSpace += document.FileSize.Value;

}

}

}

if (diskSpace > 0)

{

这里计算消耗磁盘空间：DiskSpace（消耗磁盘空间（Mb））

obs.DiskSpace = IntHelpert.Round(diskSpace, 1048576.00);

}

obs.IsSummary = false;

obs.HasImage = query.HasImage;

obs.ImageLocation = (byte)query.ImageLoaction;

obs.LastUpdateDate = DateTime.Now;

\_observationRequestRepository.Update(obs);

Document（文档记录）表的字段 BusinessID（业务唯一号） 和 ObservationRequest（观察请求记录）的ObservationUID（观察唯一序号）进行关联，相等去查询。

从Document表中获取该业务号对应的文件大小总和，修改到ObservationRequest查询出来的记录，然后update ObservationRequest的当前entity。

总结：就是对ObservationRequest表的update操作。

还是参数的校验，bll逻辑的各种判定，然后进行 crud 操作。

7.更新检查胶片数量

/// <summary>

/// 更新检查胶片数量

/// </summary>

/// <param name="query"></param>

/// <returns></returns>

[Route("v2/dataarchiving/update\_exam\_filmcount")]

[HttpPost]

public BaseReturnResult UpdateExamFilmCount([FromBody]ModifyExamFimlCountQuery query)

WCF：

var returnModel = \_imageArchiveClient.Invoke(i => i.ModifyExamFilmCount(query.ToQuery()));

依旧是对于ObservationRequest（观察请求记录）的upadate，先查ObservationRequest（观察请求记录）表的当前匹配条件的entity，然后查出Document（文档记录）表中的磁盘消耗，最终都update，体现到ObservationRequest（观察请求记录）表的记录中去。

var obs = \_observationRequestRepository.Table.Where(t => t.ObservationUID == query.ObservationUID).FirstOrDefault();

if (obs == null)

{

throw new Exception("查找不到ObservationRequest记录，ObservationUID=" + query.ObservationUID.ToString());

}

double diskSpace = 0;

var observationUID = query.ObservationUID.ToString();

var documentList = \_documentRepository.Table.Where(t => t.BusinessID == observationUID && t.DeleteFlag == false).ToList();

if (documentList != null && documentList.Count > 0)

{

foreach (var document in documentList)

{

if (document.FileSize != null)

{

diskSpace += document.FileSize.Value;

}

}

}

if (diskSpace > 0)

{

磁盘消耗每次操作都要计算啊。就是每次update操作顺便查，顺便记录进去。

obs.DiskSpace = IntHelpert.Round(diskSpace, 1048576.00);

}

obs.IsSummary = false;

obs.FilmCount = query.FilmCount;

obs.LastUpdateDate = DateTime.Now;

\_observationRequestRepository.Update(obs);

总结：这两张表也想着连起来查询，但一对多，还是单一，然后查询第二种获取所有记录，合适的。

测试是否运行到了这个表。



8.

9.

10.

11.

12.

一些：

##### 患者信息注册：输入参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **描述** | **必选** | **值域说明** |
| -PatientID | 病人号(DES加密) | 必选 |  |
| -PIDAssigningAuthority | 病人号分配机构 | 必选 | 例如：17302.UIS 机构ID+“.”+检查科室类型 |
| -PatientMasterID | 病人主索引 | 可选 | 若是已经实现主索引，需要提供主索引值 |
| -Name | 病人姓名(DES加密) | 必选 |  |
| -NameSpell | 病人姓名拼音(DES加密) | 可选 |  |
| -MotherName | 母亲姓名(DES加密) | 可选 |  |
| -Sex | 性别(DES加密) | 必选 | 男、女、未知、未说明 |
| -BirthDate | 出生日期 | 必选 | yyyy-MM-dd HH:mm:ss |
| -BirthPlace | 出生地 | 可选 |  |
| -Nation | 民族 | 可选 |  |
| -Citizenship | 国籍 | 可选 |  |
| -MaritalStatus | 婚姻状况 | 可选 | 未婚、已婚、丧偶、离婚 |
| -IDCardNo | 身份证号(DES加密) | 必选 |  |
| -HealthCardNO | 健康卡号(DES加密) | 可选 |  |
| -ContactPhoneNO | 联系电话(DES加密) | 必选 |  |
| -HomePhoneNO | 家庭电话(DES加密) | 可选 |  |
| -BusinessPhoneNO | 工作电话(DES加密) | 可选 |  |
| -Email | 电子邮箱地址 | 可选 |  |
| -AddressProvince | 地址-省份 | 可选 |  |
| -AddressCity | 地址-城市 | 可选 |  |
| -AddressDistrict | 地址-区县 | 可选 |  |
| -AddressStreet | 地址-街道、乡镇 | 可选 |  |
| -AddressRoad | 地址-街、路、村 | 可选 |  |
| -AddressDetail | 地址-详细地址 | 可选 |  |
| -Postalcode | 邮政编码 | 可选 |  |
| -Occupation | 职业 | 可选 |  |
| -WorkUnit | 工作单位(DES加密) | 可选 |  |
| -InsuranceType | 医保类别 | 可选 |  |
| -InsuranceID | 医保号(DES加密) | 可选 |  |
| -OrganizationID | 组织机构ID(DES加密) | 必选 |  |
| **-**MedRecNO | 病历号(DES加密) | 可选 |  |
| -OutPatientNO | 门诊号(DES加密) | 可选 |  |
| **-**InPatientNO | 住院号(DES加密) | 可选 |  |
| **-**InHospitalFlag | 在院状态1在院其他不在院 | 可选 |  |

患者检查申请信息上传：

###### 输入参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名称** | **描述** | **必选** | **值域说明** |
| -ObservationUID | 检查业务唯一号 | 可选 |  |
| -PatientID | 病人号(DES加密) | 必选 |  |
| -PIDAssigningAuthority | PatientID分配机构 | 必选 |  |
| -PatientMasterID | 关联病人主索引 | 必选 |  |
| -PatientClass | 病人类型， | 必选 | 门诊、急诊、住院、体检 |
| -OrganizationID | 该就诊所发生的组织机构(DES加密) | 必选 |  |
| -OrganizationName | 组织机构名 | 必选 |  |
| -ClinicInfoType | 临床信息分类 | 可选 | Exam：检查  Lab:检验 |
| -PlacerOrderNO | 申请号 (DES加密) | 可选 |  |
| -PlacerAssigningAuthority | 申请号分配机构名 | 可选 |  |
| -PlacerOrderDetailNO | 申请单明细号，多个以,分隔 | 可选 |  |
| -FillerOrderNO | 检查分配的号码，如检查的检查号、检验的检验号。(DES加密) | 可选 |  |
| -FillerAssigningAuthority | FillerOrderNO号码的分配机构名 | 可选 |  |
| -FillerPatientID | 检查分配病人号 | 可选 |  |
| -AccessionNumber | 检查号，用于DICOM访问。(DES加密) | 必选 |  |
| -ServiceID | 观察服务ID，如果是通过医嘱而来的，则与医嘱代码一致，如检查为检查项目ID、检验为检验主题ID | 必选 |  |
| -ServiceCodeScheme | 观察项目编码体系 | 可选 |  |
| -ServiceText | 观察服务内容(DES加密) | 可选 |  |
| -ServiceSectID | 观察服务类型ID | 必选 |  |
| -ProcedureID | 观察过程ID，如检查的检查部位ID，检验的检验项目ID | 可选 |  |
| -ProcedureName | 观察过程名称，如检查的检查部位名称，检验的检验项目名称 | 必选 |  |
| -ProviderID | 观察申请者ID | 可选 |  |
| -ProviderName | 观察申请者姓名 | 可选 |  |
| -ProviderPhone | 观察申请者电话 | 可选 |  |
| -RequestDeptID | 观察申请科室ID | 可选 |  |
| -RequestDeptName | 观察申请科室名称 | 可选 |  |
| -RequestedDate | 观察请求时间，申请时间 | 可选 |  |
| -ResultOrganizationID | 出结果的机构ID | 可选 |  |
| -ResultServiceCenterUID | 出结果的服务中心UID | 可选 |  |
| -Reason | 申请观察目的 | 可选 |  |
| -Attention | 申请观察注意事项 | 可选 |  |
| -Symptom | 症状体征 | 可选 |  |
| -AdverseReaction | 过敏及不良反应 | 可选 |  |
| -ClinicDiagnosis | 临床诊断 | 可选 |  |
| -RelevantClinicalInfo | 其他相关的临床信息 | 可选 |  |
| -FastingFlag | 是否空腹 | 可选 | 0非空腹，1空腹 |
| -TransportationMode | 病人运输方式 | 可选 | CART:车或担架、PORT:床边、WALK:行走、WHLC:轮椅 |
| -ResultStatus | 检查结果状态名称 | 可选 | 附录1.5.2 |
| -ResultStatusCode | 检查结果状态代码 | 可选 | 附录1.5.2 |
| -RegTime | 登记时间 | 必选 |  |
| -RegisterID | 检查登记员ID | 可选 |  |
| -RegisterName | 检查登记员姓名 | 可选 |  |
| -ObservationDeptID | 检查执行科室ID | 可选 |  |
| -ObservationDeptName | 检查执行科室名称 | 可选 |  |
| -ObservationDate | 检查时间 | 必选 |  |
| -ObservationEndDate | 检查结束时间 | 可选 |  |
| -ObservationLocation | 检查发生的场所 | 可选 | 如检查房间 |
| -ObservationEquipmentID | 检查设备的ID | 可选 |  |
| ObservationEquipment | 检查设备名称 | 可选 |  |
| -ObservationMethod | 保存该观察的所有观察方法 | 可选 |  |
| -StudyInstanceUID | 检查实例UID | 可选 | 多个以,分割开。 |
| -TechnicianID | 检查技师ID | 可选 |  |
| -TechnicianName | 检查技师名称 | 可选 |  |
| -ResultPrincipalID | 结果主要负责人ID，如报告审核医生。同ObservationResult表中相应字段值 | 可选 |  |
| -ResultPrincipalName | 结果主要负责人姓名，如报告审核医生。同ObservationResult表中相应字段值 | 可选 |  |
| -ResultReviseID | 结果修订者ID | 可选 |  |
| -ResultReviseName | 结果修订者 | 可选 |  |
| -ResultAssistantID | 结果助理ID，如报告医生。同ObservationResult表中相应字段值 | 可选 |  |
| -ResultAssistantName | 结果助理姓名，如报告医生。同ObservationResult表中相应字段值 | 可选 |  |
| -ResultAssistantPhone | 报告医生电话 | 可选 |  |
| -PreliminaryDate | 初步结果时间，如检查的初步报告时间 | 可选 |  |
| -AuditDate | 初步结果时间，如检查的初步报告时间 | 可选 |  |
| -ReviseDate | 结果修订时间 | 可选 |  |
| -ResultDate | 最后的结果时间，报告状态时为报告时间，审核状态时为审核时间，修订时为修改时间。 | 可选 |  |
| -AbnormalFlags | 异常标志，如放射结果的阴阳性 | 可选 |  |
| -CriticalValue | 危急值 | 可选 |  |
| -ResultPrintCount | 是否已打印 | 可选 |  |
| -InfectionName | 传染病 | 可选 |  |
| -Charges | 检查费用，单位为元 | 可选 |  |
| -Payments | 病人实际支付费用，单位为元 | 可选 |  |
| -PaymentsFlag | 付款标志，0未付款(未收费)、1已付款(已收费) | 可选 |  |
| -FilmCount | 胶片数 | 可选 |  |
| -FilmNeed | 是否需要胶片，0不需要，1需要 | 可选 |  |
| -HasImage | 是否有影像 | 可选 |  |
| -ImageLocation | 影像位置，0:影像在医院内、1:影像在区域中心 | 可选 |  |
| -DataSource | 检查科室代码 | 可选 | 见附录1.5.3 |
| -DiskSpace | 消耗磁盘空间（Mb） | 可选 |  |
| -SourceOrganizationID | 检测机构ID | 可选 |  |
| -SourceOrganizationName | 检测机构名字 | 可选 |  |
| -AdmitDeptID | 住院入院科室ID，门诊就诊科室ID | 可选 |  |
| -AdmitDeptName | 住院入院科室名称，门诊就诊科室名称 | 可选 |  |
| -PointOfCare | 住院时为病区，非住院时时同就诊科室 | 可选 |  |
| -PointOfCareID | 住院时为病区ID，非住院时时同就诊科室ID | 可选 |  |
| -Room | 住院时为病房，非住院时为诊间 | 可选 |  |
| -Bed | 病床 | 可选 |  |
| -CurPatCondition | 当前病情状态。 | 可选 | 1危重 2急症 3疑难 |
| **-**MedRecNO | 病历号(DES加密) | 可选 |  |
| -OutPatientNO | 门诊号(DES加密) | 可选 |  |
| **-**InPatientNO | 住院号(DES加密) | 可选 |  |
| **-**CardType | 就诊卡类型 | 可选 |  |
| **-**CardNO | 就诊卡号(DES加密) | 可选 |  |
| **-**Age | 年龄 | 可选 |  |
| **-**AgeUnit | 年龄单位 | 可选 | 岁、月、周、天 |

数据采集第二版：

1.

保存病人信息WCF：

/// <summary>

/// 保存病人信息

/// </summary>

/// <param name="query"></param>

/// <returns></returns>

public BaseResult SavePatientInfo(PatientIndexInfo patientIndexInfo)

{

return \_clinicManageService.SavePatientInfo(patientIndexInfo);

}

#endregion

值单独操作了病人主索引这一张表：

var item = \_patientIndexRepository.Table.FirstOrDefault(t => t.PatientID == patientIndexInfo.PatientID && t.PIDAssigningAuthority == patientIndexInfo.PIDAssigningAuthority);

if (item == null)

{

item = new PatientIndex();

mapExp = Mapper.CreateMap<PatientIndexInfo, PatientIndex>();

mapExp.ForMember(o => o.CreateDate, dto => dto.Ignore());

mapExp.ForMember(o => o.LastUpdateDate, dto => dto.Ignore());

item = Mapper.Map<PatientIndexInfo, PatientIndex>(patientIndexInfo, item);

item.CreateDate = DateTime.Now;

item.LastUpdateDate = DateTime.Now;

\_patientIndexRepository.Insert(item);

patientIndexInfo.CreateDate = item.CreateDate;

patientIndexInfo.LastUpdateDate = item.LastUpdateDate;

}

else

{

mapExp = Mapper.CreateMap<PatientIndexInfo, PatientIndex>();

mapExp.ForMember(o => o.CreateDate, dto => dto.Ignore());

mapExp.ForMember(o => o.LastUpdateDate, dto => dto.Ignore());

item = Mapper.Map<PatientIndexInfo, PatientIndex>(patientIndexInfo, item);

item.LastUpdateDate = DateTime.Now;

\_patientIndexRepository.Update(item);

patientIndexInfo.LastUpdateDate = item.LastUpdateDate;

}

患者注册走的WCF：

#region 患者检查归档（第三方接口）

/// <summary>

/// 患者注册

/// </summary>

/// <param name="query"></param>

/// <returns></returns>

[OperationContract]

ModelResult<Guid> SaveExamDataInfo(DataArchivingQuery query);

\_patientManageService.GetPatientIndex(patient);

通过一个查询病人信息的方法进行插入数据，但是插入完成之后又查询了一次，来区别是否要插入还是update，不知道为什么

var item = \_patientIndexRepository.Table.FirstOrDefault(t => t.PatientID == patientIndexInfo.PatientID && t.PIDAssigningAuthority == patientIndexInfo.PIDAssigningAuthority);

if (item == null)

{

item = new PatientIndex();

mapExp = Mapper.CreateMap<PatientIndexInfo, PatientIndex>();

mapExp.ForMember(o => o.CreateDate, dto => dto.Ignore());

mapExp.ForMember(o => o.LastUpdateDate, dto => dto.Ignore());

item = Mapper.Map<PatientIndexInfo, PatientIndex>(patientIndexInfo, item);

item.CreateDate = DateTime.Now;

item.LastUpdateDate = DateTime.Now;

\_patientIndexRepository.Insert(item);

patientIndexInfo.CreateDate = item.CreateDate;

patientIndexInfo.LastUpdateDate = item.LastUpdateDate;

}

else

{

mapExp = Mapper.CreateMap<PatientIndexInfo, PatientIndex>();

mapExp.ForMember(o => o.CreateDate, dto => dto.Ignore());

mapExp.ForMember(o => o.LastUpdateDate, dto => dto.Ignore());

item = Mapper.Map<PatientIndexInfo, PatientIndex>(patientIndexInfo, item);

item.LastUpdateDate = DateTime.Now;

\_patientIndexRepository.Update(item);

patientIndexInfo.LastUpdateDate = item.LastUpdateDate;

}

拼接内容：

FillerAssigningAuthority（观察分配机构名）如果为空

就是OrganizationID+ DataSource+ ServiceSectID



//数据采集从影像中心采集由于该值不存在，获取检查记录时根据检查号匹配（检查号一致，检查号机构不一致）错误

if (string.IsNullOrWhiteSpace(query.ObservationRequest.FillerAssigningAuthority))

{

string assigningAuthority = query.ObservationRequest.OrganizationID;

if (!string.IsNullOrWhiteSpace(query.ObservationRequest.DataSource))

{

assigningAuthority += "." + query.ObservationRequest.DataSource;

}

if (!string.IsNullOrWhiteSpace(query.ObservationRequest.ServiceSectID))

{

assigningAuthority += "." + query.ObservationRequest.ServiceSectID;

}

query.ObservationRequest.FillerAssigningAuthority = assigningAuthority;

}

2.

3.

4.

5.

6.

检查数据共享：

1.

获取短链接参数

/// <summary>

/// 获取授权访问短链接参数

/// </summary>

/// <param name="query">条件</param>

/// <returns>查询结果</returns>

[Route("v2/public/get\_shorturl\_parameter")]

[HttpPost]

public ModelReturnResult<ShortUrlResult> GetShortUrlParameter([FromBody]GetShortUrlRequest query)

WCF服务：

var returnmodel = \_imageArchiveClient.Invoke(i => i.GetShortUrlParameter(query.ToQuery()));

需要去校验短链接是否过期

if (model.AccessMode.HasValue && model.AccessMode.Value == 2 && model.ExpireTime.Value < DateTime.Now)//时效性访问，判断是否过期

{

result.msg = "此短链已过期";

}

入参：

{

"accessId":"93183829734ccc4c"

}

返回：

{

"code": 0,

"msg": "OK",

"data": {

"Kind": 3,

"AccessMode": 0,

"PatientMasterID": "db2c153e-36f9-44df-bf68-cc679efa91e5",

"IDCardNO": null,

"ObservationUID": "3a8e1d8a-7dbf-4ff3-be59-d7cbe0db81ce",

"AccessionNumber": "618356",

"HospitalId": "2486",

"Modality": "CT",

"CustomParamContent": null

}

}

2.

获取检查列表

/// <summary>

/// 获取相关检查列表

/// </summary>

/// <param name="query">查询条件</param>

/// <returns></returns>

[Route("v2/public/get\_relatedexam\_list")]

[HttpPost]

public ModelReturnResult<GetRelatedExamListDTO> GetRelatedExamList([FromBody]GetRelatedExamQuery query)

入参：

/// <summary>

/// 相关检查列表请求参数

/// </summary>

public class GetRelatedExamQuery : BasePageQuery

{

/// <summary>

/// 用户唯一号

/// </summary>

public string PatientMasterID { get; set; }

/// <summary>

/// 检查号唯一号

/// </summary>

public string ObservationUID { get; set; }

}

参数：

2个：

用户唯一号，检查业务唯一号不能为空（query.ObservationUID) ，(query.PatientMasterID))

WCF：

var resultdata = \_imageArchiveClient.Invoke(t => t.GetRelatedExamList(relatedQuery));

接口通的，就是没整明白为什么这里要分页，这两个入参，难道会有很多的结果集？

病人+唯一的检查号，不是就表明是唯一的结果了？

入参：

{

"PatientMasterID":"8ab73c3f-0aa8-4e9c-95fa-3ae79d4522cd",

"ObservationUID":"28914756-9544-4d50-9ccf-8b766c3fb565"

}

结果：

{

"code": 0,

"msg": "ok",

"data": {

"DataList": [

{

"OrganizationName": "1E65408A9D9BCDFB94084E78D29F48416945CC3514768E55AF741E8891119D72C30E5EE78F38CE8F",

"OrganizationID": "557175DE00BFFB2B",

"ObservationUID": "83f5d0cb-6885-4a20-92a1-d9ecd40bcdfa",

"Name": "95927258CF07D8329527472A9B9E2D5C",

"Sex": "CF067241960EED86",

"BirthDate": null,

"AccessionNumber": "C147E9671F0B9EBB5D879367D9B25847",

"ServiceSectID": "DX",

"PatientClass": "普通",

"ArchiveDate": "2018-12-18T10:01:09",

"FileSize": 7.9596719741821289,

"ProcedureName": "胸部正位",

"ResultDate": "2017-12-13T09:27:32",

"PatientID": "9E7BD662912232F1",

"PIDAssigningAuthority": "2470.RIS.DX",

"ServiceText": "普通摄片",

"InPatientNO": null,

"OutPatientNO": null,

"DataSource": "RIS",

"DataSourceName": "放射科",

"ObservationDate": "2018-11-26T09:03:23",

"ClinicInfoType": "Exam",

"PatientMasterID": "8ab73c3f-0aa8-4e9c-95fa-3ae79d4522cd",

"AuditDate": "2017-12-13T09:27:32",

"MedRecNO": "7FA22D5446120AD1CD708B9254A54C8E",

"VisitUID": "71ae1916-e4fa-448f-be30-8a82b4877e79",

"AbnormalFlags": "阳性",

"CriticalDesc": ""

}

],

"PageInfo": {

"PageIndex": 1,

"PageSize": 10,

"TotalCount": 1,

"TotalPages": 1

}

}

}

3.

获取检查详情

三个接口用到，wcf是同一个：

/// <summary>

/// 获取检查详情

/// </summary>

/// <returns></returns>

[Route("v2/public/get\_exam\_info")]

[HttpPost]

public ModelReturnResult<PatientExamDetailResultDTO> GetPatientExamInfo([FromBody]PatientExamDetailRequest query)

参数：

result.msg = "检查号，身份证号，检查业务唯一号不能同时为空";

必传：

query.hospitalId 机构解密

query.modality 检查类型

WCF：

var resultdata = \_imageArchiveClient.Invoke(t => t.GetPatientExamInfo(query.ToQuery()));

4.

5.

6.

7.

8.

9.

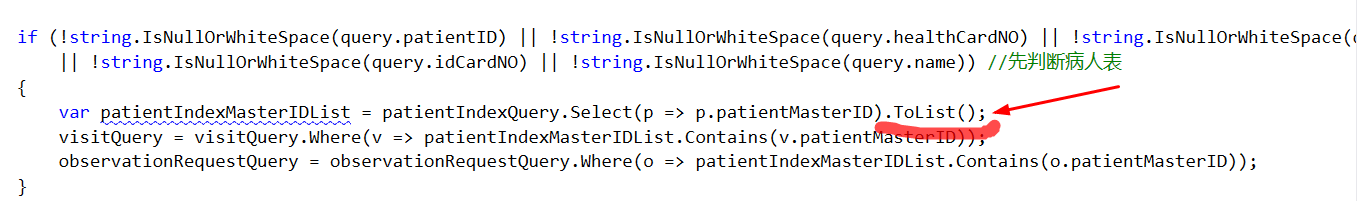
1.

没有dal层的，逻辑层和dal层完全混在一起，四百行代码的样子。

逻辑判断的同时对数据库进行操作：



一系列判断和查询之后，在这个方法将数据库断开连接，此时所有的数据加载到了内存中进行组装



多次连接，断开数据库，每次都是操作所有满足条件的数据。



最后来的分页已经没有意义了，所有的符合条件的数据一开始就全部加载出来了

result = observationRequestQuery.Skip((query.PageIndex - 1) \* query.PageSize).Take(query.PageSize).ToList();

2.

3.