## 项目经理/.NET高级开发工程师笔试题

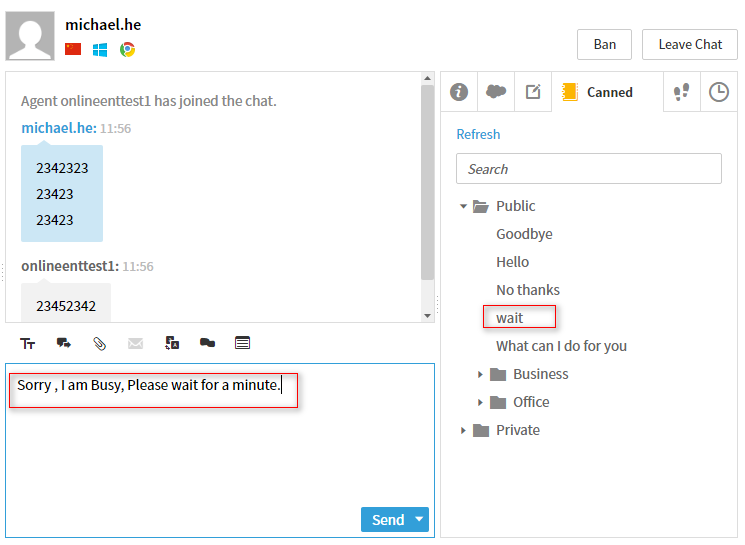
#### 一、C# &基础

某在线客服系统采用传统的三层架构

* **Data Access 层:** 负责构建SQL脚本, 直接与数据库进行交互.
* **Business 层:** 封装了实体的业务逻辑, 需要调用Data Access 层的方法才能执行对应的数据库操作.
* **Process 层:** 统一生成数据库连接、管理事务、调用Business 层的方法来处理UI层发送过来的请求.
* **UI层:** 与用户进行交互, 调用Process 层的方法完成各种UI请求.

客服在线聊天时会使用到Canned Message

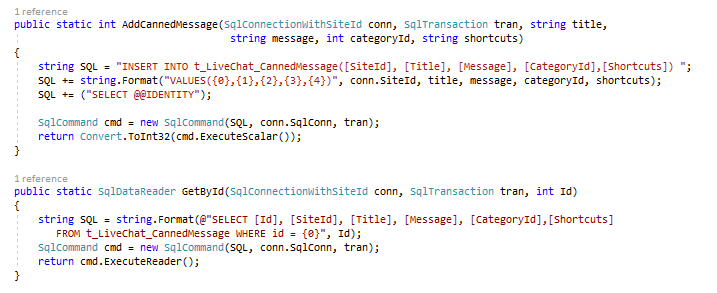
Canned Message : 预存储的常用聊天消息, 按Category 进行分类, 用于减轻客服在线聊天时的工作量. 包含Category, Title, Name, Message, Shortcuts 等属性.



举例: 客服在聊天过程中单击了Tab[Canned] 下Name为”wait”的canned message, 系统将会自动发送对应的消息: “Sorry, I am Busy, Please wait for a minute”.

以下是Canned Message 实体的相关代码:

* Data Access 层:



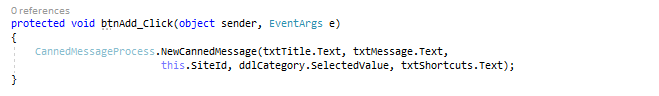
* Business 层：



* Process 层:



* UI层:



试找出上述代码的存在的错误和可优化点

1. UI层调用方法参数传参错误（少传）；process层中newcannedmessage方发可去掉operatorid参数
2. Sql拼接容易产生注入;使用sqlparameter可防止注入
3. 处理sql操作执行方法错误；增删改用executenonquery，记录集executereader，单个值executescalar
4. 一条数据插入可以不用事务
5. 查询方法可以不传参数tran
6. Ui层没有接收返回值，不能判断是否成功失败
7. Process异常没有处理，且如果执行失败无返回值；可记录日志，可在finally中添加return
8. Data Access处理数据方法可单独写到帮助类中；

#### 二、数据库

有一Ticket（个案或工单）系统，用于处理来自于客户的邮件、在线表单等问题。其中，个案的编号、标题、描述、创建时间、状态、负责人、部门等固定字段30个。用户会根据自己的业务需求添加或删除其它自定义字段，比如产品、渠道、咨询类别等。

自定义字段类型包括：文本框，下拉框，复选框，复选框列表等。其中，下拉框/复选框列表是指用户可以预定义数据，然后在具体新建或编辑Ticket的时候，客服可以选择其中的一个值。比如，用户可以加一咨询类别，预定义类别为：售前咨询，产品质量，支付问题等。在客服在新建或修改产品的时候，客服可以选择一个或多个选项。

业务系统的使用或数据量需求：

1. 我们假定系统有非常多的Ticket，比如>100万个Ticket。
2. 系统设计应该支持客户针对Ticket字段的频繁查询。比如客户定义自己的查询为“部门为客服，负责人是自己，状态为Open，自定义字段-资讯类别为产品质量”的Ticket，经常使用
3. 用户一般在使用初期会根据业务需求自定义字段，之后在系统使用过程自定义字段新建和修改会比较少；
4. 用户查询和修改一月之前的Ticket的概率会比最近时间的Ticket的概率低很多，大概1：100；

**问题：**在关系型数据库的基础上，请定义能实现自定义和存储Ticket的表结构；

1. Ticket表（主键:ticketid ，索引：创建时间 ，字段：负责人员id、邮件id、自定义表id、状态、创建时间）
2. 自定义字段类型表（主键：类型id ， 字段：类型id、类型名称 是否预定义数据、预定义值[用特殊符号隔开]、默认预定义值）
3. 自定义表（主键：自定义id ， 字段：自定义id、自定义字段名称[保存文本框，下拉框，复选框，复选框列表值]、自定义字段类型id、ticketid）
4. 人员表（主键：人员id，索引:部门id，字段：人员id 、人员名称、部门id...）
5. 部门表（主键：部门id，字段：部门id、部门名称...）
6. 邮件、表单表（主键：邮件id，索引：人员id、部门id，字段：邮件id、人员id、类型id、部门id、标题、描述... ）
7. 邮件、表单类型表（主键：类型id，类型名称）

#### 三、设计

在线客服系统是一款基于Web的实时聊天系统，用户帮助企业的客服人员回答网站访客的问题。请根据以下需求场景描述，对系统进行类设计：

1. 聊天（Chat）的双方为访客（Visitor）和客服（Agent）；
2. 客服为系统预设，属性包含Id、Name；
3. 一个聊天只能有一个访客参加，可以有多个客服参加；
4. 聊天由访客请求发起，客服人员接受请求后，双方可以聊天；
5. 聊天过程中，另外的客服人员可以加入；
6. 聊天中访客和客服都可以发送和接收聊天消息（Message）；

类设计要求体现需求描述内容和对象关系，要求包含类的定义和成员；

不限制语言，不要求实现函数体内容，不要求体现需求未描述内容；

1. 采用ibatis交互数据库、mvc架构、前端采用h5
2. 创建客服表、部门表、访客表、请求表、消息表、消息类型表、消息模板类型表、消息模板表、客服组表
3. 消息回复

MessageModel{

Public int TempletId{get;set}//模板id

Public string TempletTypeId{get;set;}//模板类型id

Public int VisitorId{get;set;}//访客id

Public int DepartmentId{get;set;}//部门id

Public int ServiceId{get;set;}//客服id

public int LanchType{get;set;}//发起类型 1访客、2客服

Public int ReplyType{get;set;}//回复类型 1访客、2客服

Public int MessageTypeId{get;set;}//消息类型id

Public string Content{get;set;}//消息内容

}

//客服组

GroupModel{

Public int DepartmentId{get;set;}//部门id

Public int ServiceId{get;set;}//客服id

}

SendMessageClass{

//访客与对应客服发起交谈

RequestNotice(int visitorId,int statues,int groupId){

//插入请求表中

1. 保存请求人id、状态为发起
2. 客服接受请求，修改状态为正在交谈
3. 查找访客组，如果访客为新访客则创建聊天组，反之沿用老聊天组

如要转接其他客服则创建另一条发起通知

}

//发送消息

SendMessage(MessageModel model){

根据发起对象插入消息表中

支持机器回答、人工回答两种

每条聊天对应聊天类型、发起人、内容、回复人

客服：记录该人员部门，若机器回答，记录机器回答类型和模板

}

//创建聊天

FondGroup(int departmentId,int bodyId){

1. 随机生成组名
2. 创建组和组成员
3. 如果判断改组中有访客，则其他访客不允许进来
4. 判断该组存在相同成员，则不予添加成员id
5. 如果判断改组成员数为0 ，删除该组

}

}

Ui：如果pc微信 显示组成员 、显示聊天记录、显示聊天框