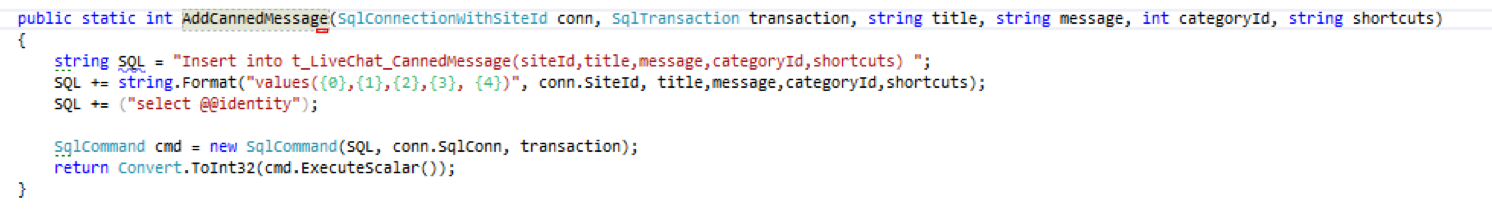
## .NET高级开发工程师笔试题

#### 一、C# &基础

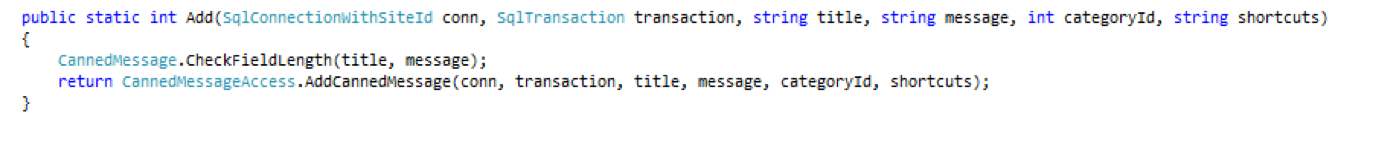
某项目采用传统的三层架构，代码的调用顺序是Data Access-->Business-->Process--> UI层

以下是相关的代码片段

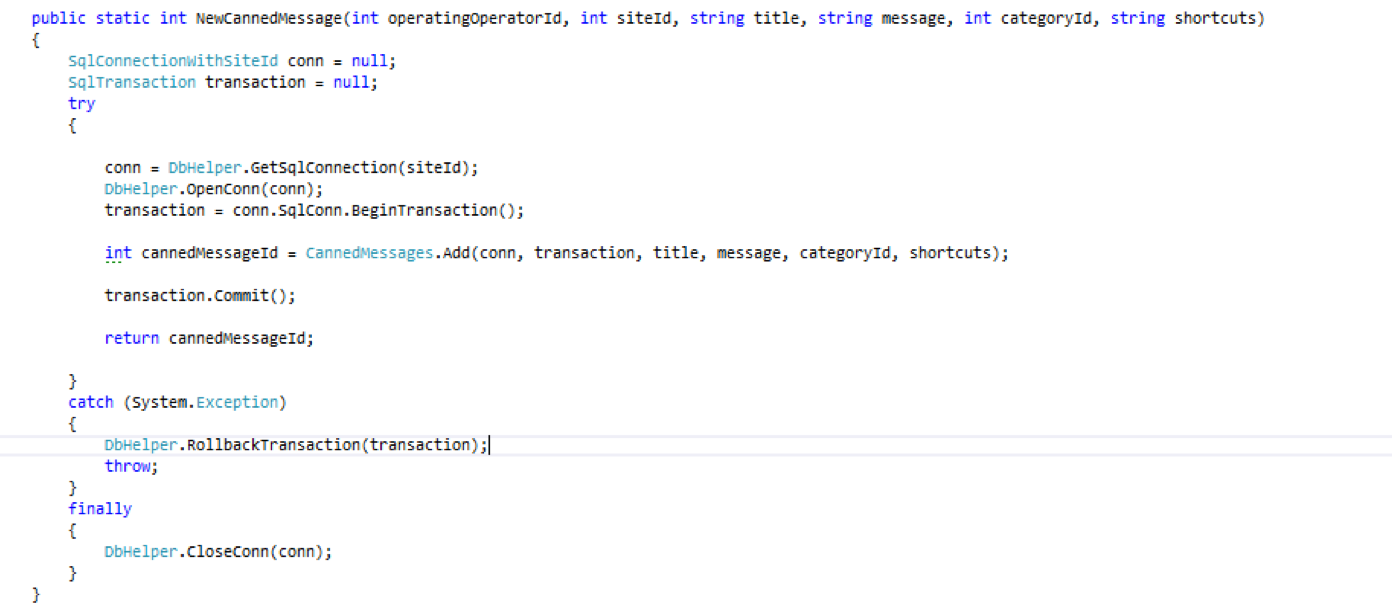
Data Access 层



Business 层



Process 层.



UI 层

直接调用Process 层的代码.

试找出上述代码的存在的错误和可优化

答案分析:

Level 1 寻找代码的错误和不合理处

考点1: SQL += “values({0},{1},{2},{3},{4})” 这个地方是有错误的, 因为title, message 这些参数是字符串类型的,应该是“values({0},’{1}’,’{2}’,{3},’{4}’)”

考点2: sql 字符串的拼接应该用StringBuilder

考点3: 传入sql的变量应该是参数化的, 否则可能会受到SQL注入攻击

考点4: DataAccess –> Bussness 🡪 Process 3层的数据传输是可以改进的, 不需要在方法里面

传递title, message, categoryId 等多个参数否则如果cannedMessage 有50个字段怎么办?)，应该把这些参数封装成一个实体类.

考点5: 变量命名有问题应该是string sql , 而不是string SQL, 内部变量是小写字母开头的.

考点6: process 层直接把异常给throw 了，是否会把异常在UI层catch 并记录日志， 如果提出了这个问题，说明对代码观察也是比较细致的

考点7: 提出了其它问题

Level 2 对整体的结构提出有建设性的意见

答案相对开放式,

因为代码中的这次层次结构是相对比较老的做法, 就单从Data Access层的实现方式来讲现在比较主流的有EF、IBatis.Net 等，有些公司为保证高性能也有全用存储过程实现的, 如果能提出另外的做法，并把该做法的优缺点，适用场景都讲的非常清楚，level 2也是ok 的（每种做法必有其优点和缺陷，优秀的程序员这方面应该都会理解的比较深透）

#### 前端

假设，前端需要从一个指定 API 获取数据。根据服务器提供的接口，返回值是一个 JSON 格式的对象。其中， 对象的属性名均为 a, b, c 等无意义的字符串。比如，在下面的例子中，属性 a 记录的是名字， 属性 b 记录的是年龄。然而，显然下面的代码不易理解，因而维护困难。

function getVisitors() {

  return fetch('https://some.url')

  .then(response => response.json())

  .catch(...);

}

function updateFirstVisitor() {

  getVisitors()

  .then((visitors) => {

   if (visitors.length === 0) return;

    const visitor = visitors[0];

    document.querySelector('app').innerHTML = [

      '<div class="name">',

        `Visitor: <span>${visitor.a}</span>`,

      '</div>',

      '<div class="age">',

        `This visitor is <span>${visitor.b}</span> year's old`,

      '</div>',

    ].join('');

  })

  .catch(...);

}

现假设服务器端的返回值不能进行修改。如何从前端入手，优化代码的可维护性？请提出解决方案，并给出 相应代码。服务器的返回值，请参考下面给出的说明：

服务器的返回值示例：

[{

  "a": "xiao ming",

  "b": "18",

  "c": [

    {

      "a": "Zootopia",

      "b": 109,

      "c": "tt2948356",

    },

    {

      "a": "Hacksaw Ridge",

      "b": 139,

      "c": "tt2119532"

    },

    {

      "a": "Kiminona",

      "b": 106,

      "c": "tt5311514"

    }

  ]

}]

数据的说明：

[{

  "名字": "xiao ming",

  "年龄": "18",

  "爱看的电影": [

    {

      "名字": "Zootopia",

      "时长（分）": 109,

      "IMDb": "tt2948356"

    },

    {

      "名字": "Hacksaw Ridge",

      "时长（分）": 139,

      "IMDb": "tt2119532"

    },

    {

      "名字": "Kiminona",

      "时长（分）": 106,

      "IMDb": "tt5311514"

    }

  ]

}]

几点注意：

1. 属性值可能是 primitive, object 或 array
2. 方案不需要考虑浏览器兼容性。但给出方案后，请注明浏览器的大致兼容范围。
3. 考虑如果需要结果可序列化，可使用 Object.keys() 取得所有属性，该怎么做？
4. 如果服务器的返回值非常庞大（比如服务器每次返回数组可能包含上千个对象） 且和服务器的沟通非常频繁（比如每秒一次和服务器进行轮询沟通），解决方案是否会有性能瓶颈？ 比如阻塞渲染？

#### 三、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；

答案分析：

注意点1: 数据量很大 > 100 万个Ticket, 设计表的时候要考虑到查询优化

2: 超过1个月的数据很少进行查询和修改，要考虑需要存档库或history库

3. 设计的表结构要比较灵活，要能满足对自定义字段的存储和查询

4. 如果自定义字段被删除/修改怎么办？历史数据如何查询?

具体实现：

1.**Ticket** 表

字段：Id (Identity(1,1) primary key) , TicketNo, Title, Description, CreateTime, StatusId, DepartmentId, PersonInChargeId

其中频繁查询的字段TicketNo, Title, CreateTime, DepartmentId, PersonInChargeId,StatusId 是需要索引的

2.**Department** 表

Id (Identity(1,1) primary key), DepartmentName, …

(其中的Id 字段对应的是Ticket表的DepartmentId

3.**PersonInCharge** 表,处理ticket的应该都是内部雇员所以也可以叫Employee表

Id(Identity(1,1) primary key), Name, ….

(其中的Id 字段对应的是Ticket表的 PersonInChargeId)

4.**Status** 表

工单系统的状态应该会是可配置化的,所以需要有status 表

#### 四、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；