## 项目经理/.NET高级开发工程师笔试题

#### 一、C# &基础(30分)

**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}’)” (3分)

考点2: sql 字符串的拼接应该用StringBuilder (3分)

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 (3分)

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.(3分)

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.(2分)

考点6: Business层的GetById 方法中DataReader 如果用using 进行关闭效果会更好. (3分)

考点7: Business层的GetById 方法中如果获取不到对应的Canned Message 返回的是null, UI层在使用返回值前必须要做检查, 否则很可能会报空指针错误，这种处理方式是不合理的.（3分）

考点8: Process 层的NewCannedMessage 方法 参数operatorId 根本没用到.(3分)

考点9: Process层的数据库操作打开关闭连接如果使用using 会更加优雅. (2分)

考点10: UI 层的btnAdd\_Click 方法没有try catch 处理, 调用Process 层的代码很可能会遇到异常, 需要捕获并记录错误日志.( 3分)

考点11:UI 层的btnAdd\_Click 方法, TextBox 控件获取的值应该要做Trim() 处理.(2分)

**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）

**(每多找出一个问题或提出一个有建设性的意见多加3分)**

#### 二、数据库（35分）

1. Ticket表自定义字段存放方式。
   1. 修改表结构，自定义字段跟系统字段一样以列的方式，能够最大化查询效率；
   2. 系统字段横向存放，自定义字段纵向排列。查询过程，系统字段存放表和自定义字段存放表联表查询

设计的方案可以实现需求.

1. 对于选择类型的字段，在Ticket表应该存放结果值，而不是选项值的Id。提高查询效率
2. 考虑查询效率，可以将Ticket表根据时间进行分表，提高查询效率；
3. 自定义字段Field的Label和SystemName两个字段应该分开

**考点1**: 提供的方案是否能够实现需求 (15分)

A)完全不知道要如何实现(0分)

B)思路上能get几个点，但是给不出具体完整的实现方案(1-5分)

C)能够给出具体完整的实现方案，根据方案的优劣打分（6-15分）

1. 修改表结构，自定义字段跟系统字段一样以列的方式，能够最大化查询效率；

如果提供的是这个方案或者其它高效的方案(12分以上)

b.如果仅仅是把自定义字段的配置信息和用户填写的自定义字段信息用单独的表存起来(8-12分)

c.其它方案，虽然能实现，但是效率低、实现复杂度高(6-8分)

**考点2:** ticket 要根据时间进行分表或把历史数据归档(5分)

**考点3**: 要在经常查询以及要进行关联查询的字段上加索引（5分）

**考点4:** 对于选择类型的字段，在Ticket表应该存放结果值，而不是选项值的Id。提高查询效率（10分）

**考点1**: 提供的方案是否能够实现需求 (15分)

A)完全不知道要如何实现(0分)

B)思路上能get几个点，但是给不出具体完整的实现方案(1-5分)

C)能够给出具体完整的实现方案，根据方案的优劣打分（6-15分）

1. 修改表结构，自定义字段跟系统字段一样以列的方式，能够最大化查询效率；

如果提供的是这个方案或者其它高效的方案(12分以上)

b.如果仅仅是把自定义字段的配置信息和用户填写的自定义字段信息用单独的表存起来(8-12分)

c.其它方案，虽然能实现，但是效率低、实现复杂度高(6-8分)

**考点2:** ticket 要根据时间进行分表或把历史数据归档(5分)

**考点3**: 要在经常查询以及要进行关联查询的字段上加索引（5分）

**考点4:** 对于选择类型的字段，在Ticket表应该存放结果值，而不是选项值的Id。提高查询效率（10分）

#### 设计（35分）

**考点1:** 有完整的类设计: 包括Chat, Agent, Visitor, Message （10分）

**考点2:** 体现正确的类关系

* 1. Visitor包含Chat成员 （5分）
  2. Chat包含Agent对象列表，包含ChatMessage列表 （5分）

**考点3:** 聊天由visitor发起,所以visitor应该要有类似StartChat 一类的方法 （5分）

**考点4:** 聊天过程中可以Add Agent, 所以Chat类要有AddAgent 一类的方法 （5分）

**考点5：**Message可抽象为基类，Agent发送的消息和Visitor发送的消息可继承（5分）；

public class **Chat**

{

IList<Agent> \_agents;

Message[] \_messages;

Public Chat()

{}

public void AddMessage(Message)

{}

Public void AddAgent(Agent agent)

{}

}

public class Visitor

{

Chat \_chat;

public Visitor()

{}

Public **Chat** StartChat(){}

}

public class Agent

{

private string \_name;

public Agent(string name)

{

\_name = name;

}

}

public abstract class Message

{

public string \_message;

public Message(string message)

{

\_message = message;

}

public abstract string GetHtmlFormat();

}

public class AgentMessage : Message

{

string \_agentName;

public AgentMessage(string agentName, string message) : base(message)

{}

public override string GetHtmlFormat()

{

return string.Format("<p color=\'blue\'>{0}: {1}</p>", \_agentName, \_message);

}

}

public class VisitorMessage : Message

{

public VisitorMessage(string message) : base(message)

{ }

public override string GetHtmlFormat()

{

return string.Format("<p color=\'black\'> {0}: {1} </p>", "Visitor", \_message);

}

}