# 程序中的字符编码

每个程序员都经历过字符乱码的困扰，经过一通折腾后，总算显示正常，但之后似乎还是时不时碰到乱码的问题。

当我们打开notepad或者ultraedit后，这些工具都会自带编码转换的选项，里面各种字符编码格式十分复杂，往往一头雾水。

这里不谈具体编码格式问题，这是一个很学究的话题。其实对于软件开发而言，除非专门做字符编码相关的软件，否则我们一般遇到的最多的也就那么几种字符编码，如utf和gbk。这里要说的是，当我们要考虑字符编码时，理解一个基本的场景模型，然后根据这个模型，在遇到乱码问题后就可以定位大致方向。

下面是这个基本模型的示意图：



1. Text source: 程序读取的外部数据容器

比如读取的外部配置文件、数据库、网络数据流甚至标准输入等。

1. App context: 程序代码的当前上下文

比如图中列举的程序中字符串变量的值、断点调试时变量显示的值等。

1. Text destination: 程序输出的外部数据容器

比如保存到输出到外部的文件、数据库甚至标准输出等。

这3个部分都有一个字符编码类型，举个列子：程序上下文环境是gbk的字符编码，读取一个utf的配置文件，然后写到设置了gbk字符编码的数据库中，结果会怎样？

Utf的配置文件被读取后，二进制内容不变，但解释不再以utf编码格式，而是gbk编码格式，所以视觉上会造成乱码。一般程序本身不会改变读取字符的二进制数据（少数情况会对特殊字符进行处理、比如‘\r’字符），之后存储到数据库中，由于数据库的字符编码是gbk，所以视觉上仍然是乱码，但如果数据库编码是utf的，则乱码会少很多（之前程序中改变了部分字符的二进制）或者完全没有乱码。