**Suggested SQL Server Videos**  
[Part 18 - Stored Procedures](http://csharp-video-tutorials.blogspot.com/2012/08/stored-procedures-part-18.html)  
[Part 55 - Error handling in SQL Server 2000](http://csharp-video-tutorials.blogspot.com/2012/10/error-handling-in-sql-server-2000-part.html)  
  
In [Part 55](http://csharp-video-tutorials.blogspot.com/2012/10/error-handling-in-sql-server-2000-part.html), of this video series we have seen Handling errors in SQL Server using **@@Error** system function. In this session we will see, how to achieve the same using Try/Catch blocks.  
  
**Syntax:**  
BEGIN TRY  
     { Any set of SQL statements }  
END TRY  
BEGIN CATCH  
     [ Optional: Any set of SQL statements ]  
END CATCH  
[Optional: Any other SQL Statements]  
  
**Any set of SQL statements**, that can possibly throw an exception are wrapped between BEGIN TRY and END TRY blocks. If there is an exception in the TRY block, the control immediately, jumps to the CATCH block. If there is no exception, CATCH block will be skipped, and the statements, after the CATCH block are executed.   
  
   
  
**Errors trapped by a CATCH block are not returned to the calling application**. If any part of the error information must be returned to the application, the code in the CATCH block must do so by using RAISERROR() function.  
  
1. **In procedure spSellProduct**, Begin Transaction and Commit Transaction statements are wrapped between Begin Try and End Try block. If there are no errors in the code that is enclosed in the TRY block, then COMMIT TRANSACTION gets executed and the changes are made permanent. On the other hand, if there is an error, then the control immediately jumps to the CATCH block. In the CATCH block, we are rolling the transaction back. So, it's much easier to handle errors with Try/Catch construct than with @@Error system function.  
  
2. Also notice that, in the scope of the CATCH block, there are several system functions, that are used to retrieve more information about the error that occurred  These functions return NULL if they are executed outside the scope of the CATCH block.  
  
3. TRY/CATCH cannot be used in a user-defined functions.   
  
   
  
Create Procedure spSellProduct  
@ProductId int,  
@QuantityToSell int  
as  
Begin  
 -- Check the stock available, for the product we want to sell  
 Declare @StockAvailable int  
 Select @StockAvailable = QtyAvailable   
 from tblProduct where ProductId = @ProductId  
   
 -- Throw an error to the calling application, if enough stock is not available  
 if(@StockAvailable < @QuantityToSell)  
   Begin  
 Raiserror('Not enough stock available',16,1)  
   End  
 -- If enough stock available  
 Else  
   Begin  
    Begin Try  
     Begin Transaction  
         -- First reduce the quantity available  
 Update tblProduct set QtyAvailable = (QtyAvailable - @QuantityToSell)  
 where ProductId = @ProductId  
   
 Declare @MaxProductSalesId int  
 -- Calculate MAX ProductSalesId   
 Select @MaxProductSalesId = Case When   
 MAX(ProductSalesId) IS NULL   
 Then 0 else MAX(ProductSalesId) end   
 from tblProductSales  
 --Increment @MaxProductSalesId by 1, so we don't get a primary key violation  
 Set @MaxProductSalesId = @MaxProductSalesId + 1  
 Insert into tblProductSales values(@MaxProductSalesId, @ProductId, @QuantityToSell)  
     Commit Transaction  
    End Try  
    Begin Catch   
 Rollback Transaction  
 Select   
 ERROR\_NUMBER() as ErrorNumber,  
 ERROR\_MESSAGE() as ErrorMessage,  
 ERROR\_PROCEDURE() as ErrorProcedure,  
 ERROR\_STATE() as ErrorState,  
 ERROR\_SEVERITY() as ErrorSeverity,  
 ERROR\_LINE() as ErrorLine  
    End Catch   
   End  
End