What is exec sp\_configure 'clr enabled', 1

Reconfigure

Endevour, ISEO

role INTERNAL-PRISMS-ADMINISTRATOR is for Developer etc.

if (SecurityHelper.IsRoleMember(Role.AUTHORITY\_CRICOS\_ADMINISTRATOR))

INTERNAL-PRISMS-BROWSE roel is for Immigration department users

**Shift + Alt + Arrow** for multiple line edits

select top 10 \* from sys.tables

To refresh the IntelliSense cache use the **Edit menu** then **IntelliSense** and **Refresh Local** Cache. You can also use **CTRL + SHIFT + R**.

Include actual execution plan (Ctrl + E & than **Ctrl + M**)

Intellisense list member and complete word (Ctrl + Space, Tab)

Display estimated execution plan (Ctrl + L)

Make selected text uppercase or lowercase (Ctrl + Shift + U, Ctrl + Shift + L)

F5, CTRL + E or ALT + X — Execute the currently selected code

CTRL + R — Show/hide the Results pane

CTRL + L — Display the query execution plan

**CTRL + SHIFT + Q** – Display the query designer

use **CTRL+SHIFT+V** to paste the current item in Clipboard Ring’s memory.

CTRL + I – Display the incremental search dialog

Exec sp\_columns 'TableName'

Select TOP 10 \* from **sys.procedures** where [Name] like 'rpt\_%'

Select count(\*) from **sys.tables**

select top 10 \* from sys.schemas

Declare table

Temp table

Union

Group By

**Sp Error**:

* rollback (if necessary) is done
* info is written (INSERT) to log table
* try/catch error handling

BEGIN TRANSACTION

BEGIN TRY

// do your SQL statements here

COMMIT TRANSACTION

END TRY

BEGIN CATCH

SELECT

ERROR\_NUMBER() AS ErrorNumber,

ERROR\_SEVERITY() AS ErrorSeverity,

ERROR\_STATE() AS ErrorState,

ERROR\_PROCEDURE() AS ErrorProcedure,

ERROR\_LINE() AS ErrorLine,

ERROR\_MESSAGE() AS ErrorMessage

ROLLBACK TRANSACTION

END CATCH

Date: FORMAT(GetDate(), 'dw dddd ddd yyyy/MM/dd hh:m:ss tt')

CONVERT(varchar(25), GetDate(),113) => 09 Feb 2018 10:20:14:163

CONVERT(varchar(20), GetDate(),100) => Nov 29 199

CONVERT(varchar(10), GetDate(),101) => 11/29/1993

CONVERT(varchar(20), GetDate() ,100) => Nov 27 2017 1:53PM

Case When r.IsActive = 1 Then 'Yes' Else 'No' End AS IsActive

IsNull(table.column,'') AS 'Title',

----

ABS() function returns the absolute value of a number.

SELECT Abs(-243.5) AS AbsNum; => 243.5

-- \*\*\* TIDY UP \*\*\*

DROP TABLE #Report

**SET NOCOUNT ON:**

Stops the message that shows the count of the number of rows affected by a Transact-SQL statement or stored procedure from being returned as part of the result set.

When SET NOCOUNT is ON, the count is not returned.

----

Filter Objects in Object Explorer

-------

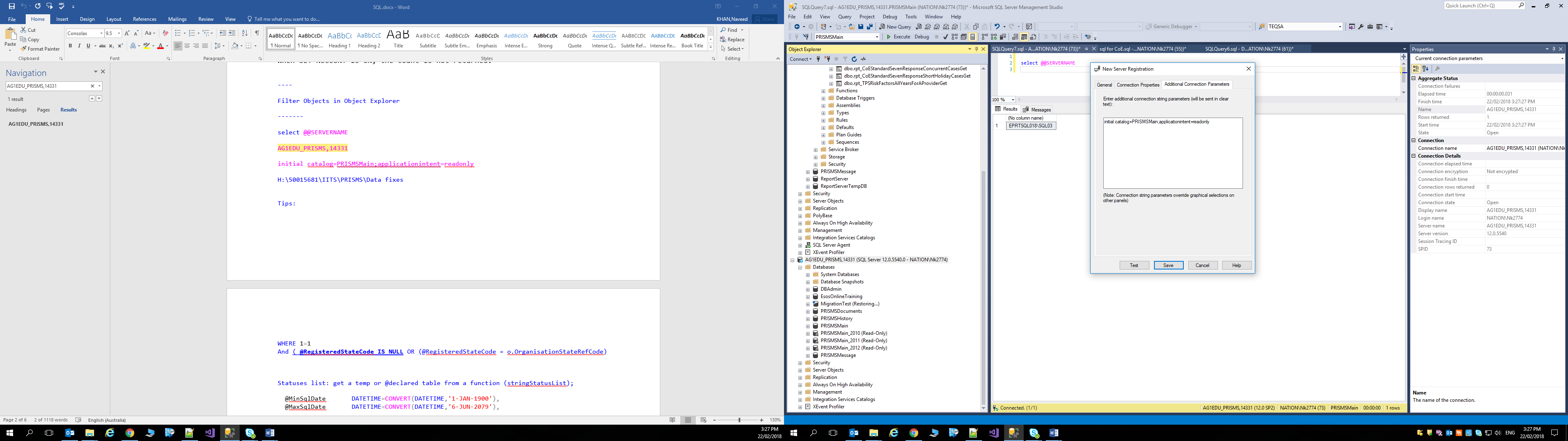
select @@SERVERNAME

AG1EDU\_PRISMS,14331

initial catalog=PRISMSMain;applicationintent=readonly

H:\50015681\IITS\PRISMS\Data fixes

Outlook email sugessin -=> Ctr + K



Tips:

WHERE 1=1

And ( **@RegisteredStateCode IS NULL** OR (@RegisteredStateCode = o.OrganisationStateRefCode)

Statuses list: get a temp or @declared table from a function (stringStatusList);

@MinSqlDate DATETIME=CONVERT(DATETIME,'1-JAN-1900'),

@MaxSqlDate DATETIME=CONVERT(DATETIME,'6-JUN-2079'),

set transaction isolation level read uncommitted : consider data which hasn't been committed yet. Is opposite to locks on data.

**Delegate**: a delegate instance points towards a method.

namespace DelegateApp {

/// <summary>

/// A class to define a person

/// </summary>

public class Person { public string Name { get; set; } public int Age { get; set; } }

class Program {

//Our delegate

public delegate bool FilterDelegate(Person p);

static void Main(string[] args) {

//Create 4 Person objects

Person p1 = new Person() { Name = "John", Age = 41 };

Person p2 = new Person() { Name = "Jane", Age = 69 };

Person p3 = new Person() { Name = "Jake", Age = 12 };

Person p4 = new Person() { Name = "Jessie", Age = 25 };

//Create a list of Person objects and fill it

List<Person> people = new List<Person>() { p1, p2, p3, p4 };

//Invoke DisplayPeople using appropriate delegate

DisplayPeople("Children:", people, IsChild);

DisplayPeople("Adults:", people, IsAdult);

DisplayPeople("Seniors:", people, IsSenior);

Console.Read();

}

static void DisplayPeople(string title, List<Person> people, FilterDelegate filter) {

Console.WriteLine(title);

foreach (Person p in people) {

if (filter(p)) { Console.WriteLine("{0}, {1} years old", p.Name, p.Age); } }

Console.Write("\n\n");

}

//==========FILTERS===================

static bool IsChild(Person p) { return p.Age < 18; }

static bool IsAdult(Person p) { return p.Age >= 18; }

static bool IsSenior(Person p) { return p.Age >= 65; } }

}

<https://www.codeproject.com/Articles/4773/Events-and-Delegates-Simplified>

When a caller invokes the delegate, it calls its target method. This way, the caller is not invoking the target method rather invoking the delegate which can call the target method. We do it because it creates an abstraction on invoking the target method. We of course always can invoke a method directly but decoupling of the client and target method is sometimes a need or gives us more flexibility to make things clean and simple.

We can use **Func** delegate to represent a method that can be passed as a parameter without explicitly declaring a custom delegate.

**Delegate int SomeDelegate(string s, bool b);**

The important thing to notice is that only functions that have the same signature as the delegate, can be passed as a parameter like the one bellow.

**private int SomeFunction(string str, bool bln){...}**

You can pass this function to SomeDelegate's constructor, because of their similar signatures.

**SomeDelegate sd = new SomeDelegate(SomeFunction);**

Now, sd refers to SomeFunction, in other words, SomeFunction is registered to sd. If you call sd, SomeFunction will be invoked.

**sd("somestring", true);**

**Understanding Events**:

This is the scenario: we have a class named Counter. This class has a method named CountTo(int countTo, int reachableNum) which starts counting from 0 to countTo, and raises an **event named NumberReached** whenever it reaches the reachableNum.

Our class has an event: **NumberReached**. **Events are variables of type delegates**. I mean, if you want to declare an event, you just declare a variable of type some delegate and put event keyword before your declaration, like this:

**public event NumberReachedEventHandler NumberReached;**

In the above declaration, NumberReachedEventHandler is just a delegate. Maybe it was better to say: NumberReachedDelegate, but notice that Microsoft doesn't say MouseDelegate or PaintDelegate, instead it offers: MouseEventHandler and PaintEventHandler. It's a convention to say.

Before we declare our event, we need to define our delegate (our event handler). It could be something like this:

**public delegate void NumberReachedEventHandler(object sender, NumberReachedEventArgs e);**

If it wouldn't be necessary to provide the user with any information, we just use the EventArgs class.

Now, every thing is prepared to take a look inside our Counter class:

**() =>** : shot-hand for:

delegate void () { return action.GenerateDescription(); }

example:

(a, b) => a + b short hand for:

delegate int (int a, int b) { return a + b; }

**Func<T>**: Func<T> is a predefined delegate type for a method that returns some value of the type T.

public static string GetMessage() { return "Hello world"; }

may be referenced like this

Func<string> f = GetMessage;

**XSLT** (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents, or other formats such as HTML for web pages.

<https://www.w3schools.com/xml/xsl_intro.asp>

XSL consists of four parts:

•XSLT - a language for transforming XML documents

•XPath - a language for navigating in XML documents

•XSL-FO - a language for formatting XML documents (discontinued in 2013)

•XQuery - a language for querying XML documents

XSLT = XSL Transformations.

XSLT is the most important part of XSL.

**WebHooks** are a lightweight HTTP pattern for event notification across the web. WebHooks enable services to send event notifications over HTTP to registered subscribers.

C#:

$ is short-hand for String.Format a new feature of C# 6

var aBool = true; var aString = "Abc Def";

var formated = $"{aBool},{aString.ToLower()}";

**using $@**: allows the features of a @"" string to be mixed with $"" to support string interpolations without the need for \\.

var someDir = "a";

Console.WriteLine($@"c:\{someDir}\b\c");