**2222’;pT-SQL**

**WEB API**

1. 1.       Difference between Clusterd and non clustered index.
2. 4.       What is Rest Service.how u implemented.
3. 5.       Project Architecture.
4. 7.       What challenge u face in your project.
5. 8When to use which type of binding in wcf in intranet and internet?

* What are instancing modes of wcf? (Ans : Per call, Per session and singleton)
* What are the threading options in wcf?
* How to implement transactions in wcf?
* Which type of binding will be used for REST services(Ans: WebHttpBinding)
* What is the architecture of wcf?
* Difference between Soap and xml?
* Modes of hosting in WCF(Self, IIS and WAS hosting)
* New features added in wcf 4.5
* How to handle SQL Exceptions?
* Difference between Dataset and Data Reader? When to use what?
* What is the use of 3NF and 2NF in Normalization?
* Difference between Pat Index and Char Index in sql server?
* Difference between Left Outer Join and Right Outer Join in sql server?
* What is candidate key in sql server?
* Difference between Delete and Truncate and Drop?
* Difference between Clustered Index and Non Clustered Index in sql server?
* How many Non Clustered Indexes exists in sql server(Ans : 2005 edition – 249, 2008 – 999)
* What is w3wp.exe in asp.net?
* Advantages of JQuery over JavaScript?
* New features added in HTML5(Ans : Audio, video tags, semantic elements etc)
* What is the minimum prerequisite to work with TCP Binding(Ans : your system should have IIS7 version or more)
* Which is faster JavaScript or JQuery? (Ans : Javascript)
* What is sql injection?
* Difference between Primary and Unique keys in sql server?
* How many types of functions does sql server support?
* Difference between procedure and functions in sql server?
* Can a stored procedure be called from function? (Ans : No)
* Difference between web service and wcf?
* Difference between Boxing and Unboxing?
* Difference between Http Handlers and Http Modules in asp.net? When to use what?

## Revising SOLID principles

S stands for SRP (Single responsibility principle):- A class should take care of only one responsibility.

O stands for OCP (Open closed principle):- Extension should be preferred over modification.

L stands for LSP (Liskov substitution principle):- A parent class object should be able to refer child objects seamlessly during runtime polymorphism.

I stands for ISP (Interface segregation principle):- Client should not be forced to use a interface if it does not need it.

D stands for DIP (Dependency inversion principle) :- High level modules should not depend on low level modules but should depend on abstraction.

If you are already done with this article the next logical steps would be going through GOF Design Patterns, here’s [an article](http://www.codeproject.com/Articles/28309/Design-pattern-FAQ-Part-1-Training) for the same, Hope you enjoy it.

Here at [DNC Magazine](http://www.dotnetcurry.com/magazine/), we’ve had a couple of readers send questions about SOLID. This issue’s column is my attempt to answer the questions.

Before getting to the questions, I want to summarize SOLID principles.

|  |  |
| --- | --- |
| **S** | [Single Responsibility Principle (SRP)](http://www.dotnetcurry.com/software-gardening/1148/solid-single-responsibility-principle) – A class should have only one reason to be changed. |
| **O** | [Open-Closed Principle (OCP)](http://www.dotnetcurry.com/software-gardening/1176/solid-open-closed-principle) – A class should be open to extension but closed to modification |
| **L** | [Liskov Substitution Principle (LSP)](http://www.dotnetcurry.com/software-gardening/1235/liskov-substitution-principle-lsp-solid-patterns) – You should be able to replace a class with a subclass without the calling code knowing about the change |
| **I** | [Interface Segregation Principle (ISP)](http://www.dotnetcurry.com/software-gardening/1257/interface-segregation-principle-isp-solid-principle) – Many specific interfaces are better than a single, all-encompassing interface |
| **D** | [Dependency Inversion Principle (DIP)](http://www.dotnetcurry.com/software-gardening/1284/dependency-injection-solid-principles) – Code should depend upon abstractions, not concrete implementations |

Patterns

Factory: Client uses factory creates product.

What are the 2 types of data types available in C#?  
1. Value Types  
2. Reference Types  
  
If you define a user defined data type by using the struct keyword, Is it a a value type or reference type?   
Value Type  
  
If you define a user defined data type by using the class keyword, Is it a a value type or reference type?   
Reference type  
Are Value types sealed?Yes, Value types are sealed.  
  
What is the base class from which all value types are derived?   
System.ValueType  
Give examples for value types?Enum  
Struct  
  
Give examples for reference types?   
Class  
Delegate  
Array  
Interface  
What are the differences between value types and reference types?  
1. Value types are stored on the stack where as reference types are stored on the managed heap.  
2. Value type variables directly contain their values where as reference variables holds only a reference to the location of the object that is created on the managed heap.  
3. There is no heap allocation or garbage collection overhead for value-type variables. As reference types are stored on the managed heap, they have the over head of object allocation and garbage collection.  
4. Value Types cannot inherit from another class or struct. Value types can only inherit from interfaces. Reference types can inherit from another class or interface.

Will the following code compile?   
double d = 9999.11;  
int i = d;  
  
No, the above code will not compile. Double is a larger data type than integer. An implicit conversion is not done automatically bcos there is a data loss. Hence we have to use explicit conversion as shown below.  
  
double d = 9999.11;  
int i = (int)d; //Cast double to int.  
  
If you want to convert a base type to a derived type, what type of conversion do you use?Explicit conversion as shown below.  
//Create a new derived type.  
Car C1 = new Car();  
// Implicit conversion to base type is safe.  
Vehicle V = C1;  
  
// Explicit conversion is required to cast back to derived type. The code below will compile but throw an exception at run time if the right-side object is not a Car object.  
Car C2 = (Car) V;

***A clustered index*** is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.

***A non clustered index*** is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

2.  What Is SQL Profiler?

SQL Profiler is a tool which allows system administrator to monitor events in the SQL server.  This is mainly used to capture and save data about each event of a file or a table for analysis.

Activity Monitor

3.

**How To Preventing Cross Site Scripting (XSS) in Angular?**

**How Angular Protects Us From XSS Attacks?**

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.

The Angular treats all values as untrusted by default. This is the great advantages of Angular.

When a value is Inserted Vulnerability into the DOM from –

1.A Template

2.Property

3.  Attribute

4. Style

5.Class Binding

6.Interpolation

7. And so on.

Angular recognizes the value as unsafe and automatically sanitizes and removes the **script tag** and other **security**vulnerabilities.

Angular provides **built-in**, values as untrusted by **default**, anti **XSS** and **CSRF**/**XSRF** protection.

The **CookieXSRFStrategy** class takes care of preventing XSS and CSRF/XSRF attacks.

The **DomSanitizationService** takes care of removing the dangerous bits in order to prevent XSS attacks.

**Angular applications must follow the same security principles as regular web applications -**

1.You should avoid direct use of the DOM APIs.

2.You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.

3.You should Use the offline template compiler.

4.You should Use Server Side XSS protection.

5.You should Use DOM Sanitizer.

6.You should Preventing CSRF or XSRF attacks.

# Dependency Injection (DI) in Angular 2 [Why @Injectable()?]

[Anil Singh](https://www.blogger.com/profile/09359926778482233933)  [11:30 PM](https://www.code-sample.com/2016/04/dependency-injection-in-angular-2.html)

Dependency Injection is a powerful pattern for managing code dependencies.

Angular 2 Dependency Injection consists of three things.

**1.Injector**

**2.Provider**

**3.Dependency**

**Injector** :- The injector object use to create instances of dependencies.

**Provider** :- A provider is help to injector for create an instance of a dependency. A provider takes a token and maps that to a factory function that creates an object.

**Dependency** :- A dependency is the type of which an object should be created.

Reactive JS

Promise

A Promise handles a single event when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

Observable

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event.

Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case.

Observable also has the advantage over Promise to be cancelable. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore.

Observable provides operators like map, forEach, reduce, ... similar to an array

There are also powerful operators like retry(), or replay(), ... that are often quite handy.

Angular Component

[A component](https://angular.io/api/core/Component) is one of the basic building blocks of an Angular app. An app can have more than one component. In a normal app, a component contains an HTML view page class file, a class file that controls the behaviour of the HTML page and the CSS/scss file to style your HTML view. A component can be created using @Component decorator that is part of @angular/core module.

import { Component } from '@angular/core';

and to create a component

@Component({selector: 'greet', template: 'Hello {{name}}!'})

class Greet {

name: string = 'World';

}

To create a component or angular app here is the [tutorial](https://toddmotto.com/creating-your-first-angular-2-component)

Angular Module

An [angular module](https://angular.io/api/core/NgModule) is set of angular basic building blocks like [component](https://angular.io/api/core/Component), [directives](https://angular.io/api/core/DirectiveDecorator), [services](https://angular.io/tutorial/toh-pt4) etc. An app can have more than one module.

A module can be created using @NgModule decorator.

@NgModule({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

In Angular 4 use @Input to share an object between parent and child. Here, changes to either megmor (in the parent), or radfal (in the child), will be reflected in the other.

Parent html:

<div>

<zoptil [radfal]="megmor"></zoptil>

{{megmor.pergal}}

</div>

Parent ts:

let megmor = new Kreven();

this.megmor.pergal = "warbar";

Child html:

<div>

<h2>{{radfal.pergal}}</h2>

<label>Peragl: </label>

<input [(ngModel)]="radfal.pergal" />

</div>

Child ts:

@Input() radfal: Kreven;

Content negotiation

*HTTP has provisions for several mechanisms for “content negotiation” — the process of selecting the best representation for a given response when there are multiple representations available.*