Interview Notes

Contents

[C# 6.0 VS 15 New Features 10/2/16 1](#_Toc461895382)

[Design Patterns 4/4/15 5](#_Toc461895383)

[Bootstrap 15/3/15 8](#_Toc461895384)

[Web Api 12/3/15 10](#_Toc461895385)

[AngularJS Intro Basics 14/2/2015 12](#_Toc461895386)

[Solid 19/01/15 16](#_Toc461895387)

[MVC 5 New Features 12/12/14 17](#_Toc461895388)

[Ajax 29/11/14 19](#_Toc461895389)

[JQuery 15/11/14 21](#_Toc461895390)

# C# 6.0 VS 15 New Features 10/2/16

**Auto Properties Getter Only**

Old

Public int x {get;set;}

New

Public int x{get;}

also allows assignment

Public int x {get;} = 7;

**Using Static Members**

using static System.math;

actually references and puts it into scope.

old

x = Math.Sqrt (???);

Now with using static

x = Sqrt(???);

no need to reference math

This also works with ENUMS

**String.Format now Interpolated String**

old

x = string.format("{0} is a child of {1}", name1, name2);

New

x = $"{name1} is a child of {name2}";

Interpolated string

**Single Expression Methods**

Use Landa syntax

now

Public string formateName() => $"{name1} is a child of {name2}";

public double Dist => sqrt(x \* X + Y);

null check - ?. operator

null check - ?. operator

**null check - ?. operator**

json["x"]?.Type == JTokenType.Integer &&

json["y"]?.Type == JTokenType.Integer &&

if left hand thing is null the whole thing is null

if left hand thing is not null then we do the dot

old

json["x"] !=null && json["x"].Type == JTokenType.Integer

bonus

json?['x']?.Type -- first question mark validates initial object is not null

**Catch Statement Mods**

Allows filtering of exceptions

Previously would catch and rethrow. this is more effecient

catch (ConfigurationException e) when (e.IsServere)

await statements are also allowed in catch and finally blocks

**.NET Native**

New pre-compilation story makes apps run faster and more responsive

faster startup of apps and lower memory usage

Performance of C++ Dev Productivity of C#

This is part of CORE 5

**Roslyn**

New open source compiler platform for C# and VB.Net

**Framework 4.6**

Highly compatible, in place replacement for 4, 4.5, 4.5.1 and 4.5.2

Full support of any .NET api lib

WPF is platform of choice for desktop app

ASP.NET 5 is supported

4.6 leverages the investment in new compilers JIT

**APS.NET 5.0**

Cloud - ready

- leaner, faster, simpler

- designed from top to bottom for cross platform deployments (windows and mac)

Modular and open

- more flexible.

Improved tooling and frameworks

- deliver value faster with improved tooling

uses in memory compilation tool that greatly reduces build time and debugging time

**.NET Core 5**

All opens source.

Lighter weight.

all benefits of framework 4.6

GetHup as default platform for code

# Design Patterns 4/4/15

**3 Types of Design Patterns**

Creational -creation of object

Behavioral - process or flow

Structural - describe static structure

**Creation Patterns**

These Create Objects

- Abstract factory: groups object factories that have a common theme

- Builder: constructs complex objects by separating construction and representation

- Factory Method: creates objects without specifying the exact class to create

- Prototype: Creates objects by cloning an existing object

- Singleton: restricts objects creation for a class to only one instance

**Structural Patterns**

Adapter - Match interfaces of different classes. Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces

Bridge - Separates an object's interface from its implementation. Decouple an abstraction from its implementation so that the two can vary independently.

Composite - A tree structure of simple and composite objects. Compose objects into tree structures to represent par-whole hierarchies. Composite lest clients treat individual objects and compositions of object uniformly.

Decorator - Add responsibilities to objects dynamically. Attach additional responsibilities to an object dynamically. Decorators proved a flexible alternative to sub classing for extending functionality.

Facade - A single class that represents an entire subsystem. Provide a unified interface to a set of interfaces in a system. Facade defines a higher-level interface that makes the subsystem easier to use.

Flyweight - a fin-grained instance used for efficient sharing. Use sharing to support large numbers of fine-grained objects efficiently.

Proxy - An object representing another object. Provide a surrogate or placeholder for another object.

**Behavioral Patterns**

Chain of Resp - A way of passing a request between a chain of objects. Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.

Command - Encapsulate a command request as an object thereby letting you parameterize clients with different requests, queue or log requests and support undo-able operations.

Interpreter - A way to include language elements in a program. Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.

Iterator - Sequentially access the elements of a collection. Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

Mediator - Defines simplified communication between classes. Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly and it lets you vary their interaction independently.

Observer - A way of notifying change to a number of classes. Define a one to many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

State - Alter an objects behavior when its state changes. Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.

Strategy - Encapsulates an algorithm inside a class. Define a family of algorithms, encapsulate each one and make them interchangeable.

Template - defer the exact steps of an algorithm to a subclass. Define the skeleton of an algorithm in an operation, deferring some steps to subclasses.

Visitor - Defines a new operation to a class without change. Represents an operation to the performed on the elements of an object structure. Add new operation without changing the classes of the elements on which it operates.

**Abstract Factory Pattern**

Creational Pattern, used to create a set of related or dependent objects.

Acts a a super-factory which creates other factories. Called Factory of factories. An interface is responsible for creating a set of related objects, or dependent objects without specifying their concrete classes. can implement Builder, Prototype and Factory.

Abstract Factory interface defines the type of products each factory will make. (ie bike, scooter)

Concrete implementations of the factory will created the projects. (ie Honda, Harley)

Abstract Product Interface will define the types of products (ie sport bike or cursing bike)

Concrete Product is the the output of a Concrete Factory (ie Harley Sport Bike).

**Repository Pattern**

Allows all of your code to use objects without having to know how the objects are persisted.

sample - use one object for db table interaction.

simple implementation is nothing but a separation layer between the data and domain layers of an application.

**Chain of Responsibility Pattern**

Defines a linked list of handlers, each of which is able to process requests. When a request is submitted to the chain it is passed to the first handler in the list that is able to process it.

have a source of command objects and a series of processing objects. The command is passed to the first processing object which can handle this command or sent to its successor.

Client calls HandlerBase which points to Concrete Handlers.

HandlerBase is an interface or base class for all concrete handlers. It contains a member variable which points to the next process object.

Concrete Handler is a concrete implementation of the HandlerBase class.

**Abstract Factory vs Factory Patterns**

Abstract implementation involves dependency injection by passing in the dll.

Factory patters clients are hard coded with the implementation of the sub classes.

**Builder vs Factory Creator**

Factory returns a completed object

ie new car('bmw','black','M2);

Builder new car

car.make = 'bmw'

car.color = 'black'

# Bootstrap 15/3/15

**Single Responsibility Principle**

Classes are designed to provide a single piece of functionality. Combine classes to get overall result.

navbar navbar-inverse navbar-fixed-top

navbar styles --- for nav bar

navbar-invers --- flips colors to dark

navbar-fixed-top --- fixes to top

combine for effect

**Widgets**

ie navbar

apply certain classes and a specific structure and bootstrap will build menu

simpler than components? Maybe???

**Defaults**

Bootstrap applies a default style to all tags even in a specific class in not assigned. This is to assure a consistent look across browsers.

**Grid Based Layout**

Provides a grid system and a container allowing layout in columns. There are 12 columns in Bootstrap.

Users row and col tags.

Rows push down page. Columns are horizontal.

col-md-4 is a 4 unit column of medium. This will have columns for med & large but single row for sm & ex. Use sm to have columns on small etc

Grids can be nested.

Col-md-offset-6 this will offset the col placement to allow for blanks.

push & pull --- col-md-push-9 will push right , pull will move left

**Container**

Outer most bootstrap element. Provide padding and max width set by device. Used for styling and alignment.

**Responsive Design**

By Default. Designed for min. All automatic if use correct class.

CSS Media queries will query browser to apply diff rules for diff browser sized.

Extra Small under 768 px, Small over 768 px, Med >= 992 and Large over 1200px

**Hiding Elements in Responsive Design**

There is a css tab hidden-lg, hidden-md, hidden-print, hidden-sm, hidden-xs that will hide an element based upon media size.

There are also visible tags... .visible-xs, etc

**Components**

Multiple elements and require structure.

Drop down menus and Fly out menus.

getbootstrap.com/components for more samples.

# Web Api 12/3/15

**REST**

Representational State Transfer is not a protocol but an architecture and design pattern for building and calling web services.

Uses HTTP Protocol. supports full features of http like URIs, response/request headers, caching, versioning and content formats.

supports MVC routing, controllers, action results, filter , model binders, IOC

can be hosted in app or IIS.

responses formatted by Web API's MediaTypeFormatter into JSON, XML or whatever format you add to MediaTypeFormatter.

Resource Driven - architecture that exposes endpoints based on objects and not functions. Nouns vs Verbs

**Request Verbs**

Http Verbs or Actions

Describe what can be done to the resource.

Get - non destructive

Post - creating

Put - updating

Delete - deleting

Get can be called many times and would always return same info.

**Discoverable**

freely accessed in URLs. Should be intuitive enough to figure out.

**Resources**

REST uses addressable resources to define the structure of the API. These are the URLs used to get to pages on the web site.

endpoints tied to specific resources. built on domain model vs actions

**Response Status Codes**

Follow HTTP Standards.

Use HTTP responses to indicate messages.

**Hypermedia Controls**

The point of hypermedia controls is that they tell the consumer what they can do next, and the URI of the resource needed to manipulate.

The reply contains a number of hypermedia controls

- link rel = ''/linkrels/appoiment/cancel" uri = "/slots/1234?appointment/

- link rel = "/linksrel/appointment/addTest" uri = "/slots/1234/appointments/tests"

**Resource Mapping**

Resources relate to domain objects. For example only on resource which is Task.

Action Method URI

Get all the tasks - GET - /tasks

Get single task - GET -/tasks/id

Create task - POST -/tasks

Edit new task - PUT -/tasks/id

Delete task -DELETE -/tasks/id

**Request Headers**

Additional instructions that are sent with the request. These might define what type of response is required or authorization details.

**Request Body**

Data that is sent with the request. For example a POST will require some data which is typically sent as the request body in the format of JSON or XML

**Response Body**

This is the main body of the response. If the request was to a web server, this might be a full HTML page, if it was to an API this might be a JSON or XML doc.

# AngularJS Intro Basics 14/2/2015

|  |  |
| --- | --- |
| **ng-app directive** |  |

html ng-app

Angular directive flags the html element that Angular should consider to be the root element of our application. ie either the entire page or part of the page can be treated as an angular app

**angular.js script**

need to include the angular js script.

**app bootstrap**

the term bootstrap is used in agular js.

3 things happen:

1. the injector that will be used for dependency injection is created.

2. the injector will then create the 'root scope' that will become the context for the model of our application.

3. Angular will then 'compile' the DOM starting at the ngApp root element, processing and directives and bindings found along the way.

Once an application is bootstrapped it will wait for incoming browser events that might change the model.

**{{xxx}}**

used to bind elements. this is not a one time insert. this will result in efficient continuous updates whenever the result of the expression evaluation changes. MVVM vs MVC.

**Angular Expression**

JavaScript like code snippet that is evaluated by Angular in the context of the current model scope rather than the scope of the global context (window).

**View and Template**

In Angular, the view is a projection of the model through the HTML template. This means that whenever the model changes, Angular refreshes the appropriate binding points, which updates the view.

Template - html page

View - result of angular binding model to template.

**Angular Directives**

ie ng-app, ng-controller, ng-repeate

attributes attached to html tags to bind specific behavior from angular js.

**ng-repeat directive**

instantiates a template once per item from a collection. Each template instance gets its own scope where the given loop variable is set to the current collection item.

used to display a list of phone numbers.

**ng-controller directive**

used to identify the controller.

**Model and Controller**

Model - contains the data

Controller - is a constructor function that takes a scope parameter and creates the model.

**Filter Query**

can apply to repeater binding to automatically filter items returned.

**Two Way Data Binding**

Allowing uses to change the sort order of a bound element?

**Dependency Injection**

Using an HTTP request to your web server to fetch the data.

Anjular will then Inject the Services.

Managed by Angular's DI Subsystem. Dependency injection helps to make your web apps both well structured (separate components for presentation, data, and control) and loosely coupled (dependencies between components are not resolved by the components themselves but by DI)

**Layout Template**

A template can be a html page that renders only one view. The same template can then be expanded to render multiple views.

However it may be better to create a layout template.

A Layout Template is common for all views in the application. Other Partial Templates are then included into the layout template depending upon the current route (view being displayed).

**Deep Linking**

Allows use of browsers history (back and forward) with Layout Templates

**$routeProvider**

Applications routes in Angular are declared via the $routeProvider which is a service that makes it easy to wire together controllers, view templates and the current URL location in the browser. Allows the implementation of Deep Linking.

**ngView Directive**

used in the layout template to specify where the view will be displayed.

**2 Way Binging**

when a bound item is edited the new value is displayed at all places the item is bound.

**ng-bind**

This is used for 1 way binding. To display only.

Shortcut is --- {{ some value}}

title ng-bind="some value"

**ng-model**

Used for two way binding. Usually associated with an Input control

input type = 'text' ng-model='modelname'

**data-ng-show**

Used to dynamically hide or show html.

Associated with div class = "form group"

Can include logic. data-ng-show = "vm.editItem.EntiityStat == 1"

**ng-disabled**

usually tied to validations. data elements will have a "valid" attribute. the add button will be enabled or disabled based upon ng-disabled = "myForm.$invalid"

# Solid 19/01/15

**Single Responsibility Principle**

Class should exist for only one reason.

Every software module should have only 1 reason to change.

Only one job to do. Everything related to a single purpose.

No Swiss Army Knife.

**Open Closed Principle**

Open for extension. Closed for modification.

Use abstraction or by placing behavior in derivative classes. Create Base Class and Override functionality.

**Liskov Substitution Principle**

You should be able to use any derived class instead of a parent class and have it behave in the same manner without modification. Insures a derived class does not affect the behavior of the parent class.

Square - Rectangle Example

Father a doctor, son a plumber. Son can not replace father

**Interface Segregation Principle**

Create an interface for a client. A client should not be forced to use an interface that contains interacts that the client does not need or use.

**Dependency Inversion Principle**

Abstraction should not depend upon details. Details should depend upon abstractions.

Pass Dependencies to class. Ie connection strings.

Write data class that will write data either to db or xml file dependent upon what is passed.

# MVC 5 New Features 12/12/14

**Attribute Based Routing**

Define the route where the action method is defined.

Need to add routes.MappMvcAttributeRoutes to routing config file.

Also allows optional parameter of ?

Route Prefix. If multiple actions methods in a controller all using the same prefix we can use RoutePrefix attribute on the controller instead of putting that prefix on every action.

[RoutePrefix("Products")] - apply to contoller

[Route("Electronics/{id})] - to actions

**Filters**

Filterers provide an elegant way to implement cross cutting concerns. Cross cutting concerns is the functionality that is used across the application in different layers. This includes caching, exception handling and logging.

Create a Global filter by creating a class and registering it as a global filter

for the filter class

implement ActionFilterAtttibute

override the behaviour ie onActionExecuting

do something...

Register in global filter

GlobalFilters.Filters.Add(new TestGlobalFilterAttribute());

**Filter Overrides**

Filter override feature applied t othe action method or controller which selectively excludes the global filter or the controller filter for the specified action method or controller

**Filter and Override Types**

Action Filter - OverrideActionFilterAttribute

Authentication Filter - OverrideAuthenticationAttribute

Authorization Filter - OverrideAuthorizationAttribute

Exception Filter - OverrideExceptionAttribute

**ASP.Net Identity**

- works with any asp.net framework

- easily add third party authentication providers like google or facebook

- control of the persistence storage. store in sql server, azure of NoSQL db

**Scaffolding for CRUD**

Now a seperate menu option for Scaffolding items...

**Enum support in views**

html.EnumDropDownListFor - must evaluate to an enum type of a Nullable T where T is an enum type.

EnumHelper.GetSelectList returns an IList SelectListItem . useful to manipulate a select list prior to calling.

**Bootstrap support for editor templates**

now to pass in HTML attributes in EditorFor as anonymous object

Html.EditorFor (model - model, new (htmlAttributes = new {@class = ''}

**Min & Max Len Attributes in Validation**

client side validation for string and array types will now be supported for properties decorated with the MinLength and MaxLength attributes.

**this in Unobtrusive Ajax**

Callback functions (OnBegin, OnComplete, OnFailure, OnSuccess) will not be allowed to loacate the invoking element via the this context...

# Ajax 29/11/14

|  |  |
| --- | --- |
| **Get vs Post** |  |

The GET method should be used for non-destructive operations — that is, operations where you are only "getting" data from the server, not changing data on the server. For example, a query to a search service might be a GET request. GET requests may be cached by the browser, which can lead to unpredictable behavior if you are not expecting it. GET requests generally send all of their data in a query string.

The POST method should be used for destructive operations — that is, operations where you are changing data on the server. For example, a user saving a blog post should be a POST request. POST requests are generally not cached by the browser; a query string can be part of the URL, but the data tends to be sent separately as post data

**Return Data Types**

text

For transporting simple strings.

html

For transporting blocks of HTML to be placed on the page.

script

For adding a new script to the page.

json

For transporting JSON-formatted data, which can include strings, arrays, and objects.

xml

for transporting data in a custome xml schema

**Asynchronous & Callbacks**

The asynchronicity of Ajax catches many new jQuery users off guard. Because Ajax calls are asynchronous by default, the response is not immediately available. Responses can only be handled using a callbac

**Sample Call Includes**

1. Ajax Wrapper

2. url

3. data array

4. type (get or post)

5. dataType (json)

6. success function

7. error function

8. complete function - always runs

**async options**

by default the call is async.

set to false will send synchronously which will block execution of other code.

**Ajax Convenience Methods**

$.get - preforms the get provided by url

$.get('url', function()--build function);

$.post - preforms the post provided by url

$.post('url', function()--build function);

$.getScript - add a script to the page

$.getScript('url', function()--build function);

$.getJSON - preform a get request and the expected JSON to be retruned

$.get('url', function()--build function);

**.serialize()**

serializeds a form's data into a query string. elements must have a name attribute

**.serializeArray()**

sends array

# JQuery 15/11/14

**Plugin**

Built off of JQuery. May have a visual component or may not. Data Table is example.

**Widget**

Built off of JQuery UI. Has a visual component. Accordion, Date Pic.ker, Dialog, Tabs, Autocomplete

**onload function**

assures the browser has finished loading all code. this includes downloading images and banner ads. and is slower that .ready

**$(document).ready(function()**

assures that the document has loaded and is ready to be manipulated.

**event.preventDefault()**

This overrides the default behavior.

**Callback Function**

a function that is passed as an argument to another function and is executed after its parent function has completed.

**$.get('somepage.html', someCallBack);**

when $.get() finishes getting the page the call back function is executed.

**passing parameters to a callback function**

must use an anonymous function as a wrapper.

This fails $.get('mypage', myCallBack(parm1,parm2));

This works $.get('mypage', function(){myCallback(parm1,parm2);});

**find speed by id, tag or class**

id is the fastest because it directly implements javascript functionality

class is the second fastest.

tag is the slowest.

**$.fn namespace**

Methods called on jQuery selections are in the $.fn namespace and automatically receive and return the selection of this. also called the 'jQuery prototype'. jQuery Object Methods.

**$ namespace**

utility type methods and do not work with selections. they are not automatically passed any arguments and their return value will vary. ie $ .each()

**$()**

shorthand for $(document).ready()

**chaining**

calling a method that returns a jQuery object and then continue to call jQuery methods on the object without pausing for a semicolon.

**Adding Attributes to an element or getting the val**

$("a").attr("attname","att value");

var x = $("a").attr("href");

**Selectors**

by id $('#myId');

by class name $('.myclassname');

by attribute $('input[name = 'first\_name']');

by elements and compound css

$('#contents ul.people li');

**Pseudo Selectors**

tr:odd

div:visible

div:hidden

-- visible and hidden jQuery checks actual value vs css value

**Selector Performance Over Specifying**

#myTable th.special is faster than #myTable thead tr th.special

attribute specification can be very slow

**does selector contain any elements**

doesn't work - selector always return s true

if( $('div.foo')){do something;}

use length

if( $('div.foo').length){do something;}

**Transversing Elements**

Moving from parent to child or Siblings

**Data**

You can store data with an element. jQuery manages the memory issues.

$('myDiv').data('keyName', {foo: 'bar'}); sets value

$('myDiv').data('keyName'); returns {foo:'bar'}

|  |  |
| --- | --- |
| **stopPropagation** |  |

prevent the event from bubbling up the dom tree.

**.one**

assures an event only runs one time. prevents doubleclick...

**Advantages over Java Script**

jQuery abstracts functionality from java script. Using jQuery reduces the need for browser specific java script to be created.