

NUTECH COMPUTER TRAINING INSTITUTE

1682 E. GUDE DRIVE #102, ROCKVILLE, MD 20850

WEB: www.NUTECHTRAINING.COM TEL: 301-610-9300

MCPD .NET 3.5 Package – Course Outlines

TS 70-536: Microsoft .NET Framework - Application Development Foundation

Developing applications that use system types and collections

- 1) Manage data in a .NET Framework application by using .NET Framework system types.

Value types; Nullable type; Reference types; Attributes; Generic types; Exception classes; Boxing and UnBoxing ; TypeForwardedToAttribute class

- 2) Manage a group of associated data in a .NET Framework application by using collections.

ArrayList class; Collection interfaces; Iterators; Hashtable class; CollectionBase class and ReadOnlyCollectionBase class; DictionaryBase class and DictionaryEntry class; Comparer class; Queue class; SortedList class; BitArray class; Stack class

- 3) Improve type safety and application performance in a .NET Framework application by using generic collections.

Collection.Generic interfaces; Generic Dictionary; Generic Comparer class and Generic EqualityComparer class; Generic KeyValuePair structure; Generic List class, Generic List.Enumerator structure, and Generic SortedList class; Generic Queue class and Generic Queue.Enumerator structure; Generic SortedDictionary class; Generic LinkedList; Generic Stack class and Generic Stack.Enumerator structure

- 4) Manage data in a .NET Framework application by using specialized collections.

Specialized String classes; Specialized Dictionary; Named collections; CollectionsUtil; BitVector32 structure and BitVector32.Section structure

- 5) Implement .NET Framework interfaces to cause components to comply with standard contracts.

IComparable interface; IDisposable interface; IConvertible interface; ICloneable interface; IEquatable interface; IFormattable interface

- 6) Control interactions between .NET Framework application components by using events and delegates.

Delegate class; EventArgs class; EventHandler delegates

Implementing service processes, threading, and application domains in a .NET Framework application

- 1) Implement, install, and control a service.

Inherit from ServiceBase class; ServiceController class and ServiceControllerPermission class; ServiceInstaller and ServiceProcessInstaller class; SessionChangeDescription structure and SessionChangeReason enumeration

- 2) Develop multithreaded .NET applications.

Thread class; ThreadPool class; ThreadStart delegate, ParameterizedThreadStart delegate, and SynchronizationContext class; Timeout class, Timer class, TimerCallback delegate, WaitCallback delegate, WaitHandle class, and WaitOrTimerCallback delegate; ThreadExceptionEventArgs class and ThreadExceptionHandler class; ThreadState enumeration and ThreadPriority enumeration; ReaderWriterLock class; AutoResetEvent class and ManualResetEvent class; IAsyncResult interface and ICancelableAsyncResult interface (refer System Namespace); EventWaitHandle class, RegisterWaitHandle class, SendOrPostCallback delegate and IOCompletionCallback delegate; Interlocked class, NativeOverlapped structure and Overlapped class; ExecutionContext class, HostExecutionContext class, HostExecutionContextManager class, and ContextCallback delegate; LockCookie structure, Monitor class, Mutex class, and Semaphore class

- 3) Create a unit of isolation for common language runtime within a .NET Framework application by using application domains.

Create an application domain; Unload an application domain; Configure an application domain; Retrieve setup information from an application domain; Load assemblies into an application domain

Embedding configuration, diagnostic, management, and installation features into a .NET Framework application

- 1) Embed configuration management functionality into a .NET Framework application.

Configuration class and ConfigurationManager class; ConfigurationSettings class, ConfigurationElement class, ConfigurationElementCollection class and ConfigurationElementProperty class; Implement IConfigurationSectionHandler interface; ConfigurationSection class, ConfigurationSectionCollection class, ConfigurationSectionGroup class and ConfigurationSectionGroupCollection class; Implement ISettingsProviderService interface; Implement IApplicationSettingsProvider interface; ConfigurationValidationBase class; Implement IConfigurationSystem interface

- 2) Create a custom Microsoft Windows Installer for .NET components by using the System.Configuration.Install namespace, and configure .NET Framework applications by using configuration files, environment variables, and the .NET Framework Configuration tool (Mscorcfg.msc).

Installer class; Configure which runtime version a .NET Framework application should use; Configure where the runtime should search for an assembly; Configure the location of an assembly and which version of the assembly to use; Direct the runtime to use the DEVPATH environment variable when searching for assemblies; AssemblyInstaller class; ComponentInstaller class; Configure a .NET Framework application by using the .NET Framework Configuration tool (Mscorcfg.msc); ManagedInstallerClass; InstallContext class; InstallerCollection class; Implement IManagedInstaller interface; InstallEventHandler delegate; Configure concurrent garbage collection; Register remote objects by using configuration files

- 3) Manage an event log by using the System.Diagnostics namespace.

Write to an event log; Read from an event log; Create a new event log

- 4) Manage system processes and monitor the performance of a .NET application by using the diagnostics functionality of the .NET Framework.

Get a list of all running processes; Retrieve information about the current process; Get a list of all modules loaded by a process; PerformanceCounter class, PerformanceCounterCategory and CounterCreationData class; Start a process both by using and by not using command-line arguments; StackTrace class; StackFrame class

- 5) Debug and trace a .NET Framework application by using the System.Diagnostics namespace.

Debug class; Debugger class; Trace class, CorrelationManager class; TraceListener class; TraceSource class; TraceSwitch class; XmlWriterTraceListener class; DelimitedListTraceListener class and EventlogTraceListener class; Debugger attributes

- 6) Embed management information and events into a .NET Framework application.

Retrieve a collection of Management objects by using the ManagementObjectSearcher class and its derived classes; ManagementQuery class; Subscribe to management events by using the ManagementEventWatcher class

Implementing serialization and input/output functionality in a .NET Framework application

- 1) Serialize or deserialize an object or an object graph by using runtime serialization techniques.

Serialization interfaces; Serialization attributes; SerializationEntry structure and SerializationInfo class; ObjectManager class; Formatter class, FormatterConverter class, and FormatterServices class; StreamingContext structure

- 2) Control the serialization of an object into XML format by using the System.Xml.Serialization namespace.

Serialize and deserialize objects into XML format by using the XmlSerializer class; Control serialization by using serialization attributes; Implement XML serialization interfaces to provide custom formatting for XML serialization; Delegates and event handlers provided by the System.Xml.Serialization namespace

- 3) Implement custom serialization formatting by using the Serialization Formatter classes.

SoapFormatter; BinaryFormatter class

4) Access files and folders by using the File System classes.

File class and FileInfo class; Directory class and DirectoryInfo class; DriveInfo class and DriveType enumeration; FileSystemInfo class and FileSystemWatcher class; Path class; ErrorEventArgs class and ErrorHandler delegate; RenamedEventArgs class and RenamedEventHandler delegate

5) Manage byte streams by using Stream classes.

FileStream class; Stream Class (NOT Readers and Writer classes, as they are separate objectives); MemoryStream class; BufferedStream class

6) Manage .NET Framework application data by using Reader and Writer classes.

StringReader class and StringWriter class; TextReader class and TextWriter class; StreamReader class and StreamWriter class; BinaryReader class and BinaryWriter class

7) Compress or decompress stream information in a .NET Framework application and improve the security of application data by using isolated storage.

IsolatedStorageFile class; IsolatedStorageFileStream class; DeflateStream class; GZipStream class

Improving the security of .NET Framework applications by using the .NET Framework security features

1) Implement code access security to improve the security of a .NET Framework application.

SecurityManager class; CodeAccessPermission class; Modify the Code Access Security Policy at machine, user, and enterprise policy level by using the Caspol tool; PermissionSet class, NamedPermissionSet class, and PermissionSetCollection class; Standard Security interfaces

2) Implement access control by using the System.Security.AccessControl classes.

DirectorySecurity class, FileSecurity class, FileSystemSecurity class, and RegistrySecurity class; AccessRule class; AuthorizationRule class and AuthorizationRuleCollection class; CommonAce class, CommonAcl class, CompoundAce class, GeneralAce class, and GeneralAcl class; AuditRule class; MutexSecurity class, ObjectSecurity class, and SemaphoreSecurity class

- 3) Implement a custom authentication scheme by using the System.Security.Authentication classes.

Authentication algorithms and SSL protocols

- 4) Encrypt, decrypt, and hash data by using the System.Security.Cryptography classes.

DES class and DESCryptoServiceProvider class; HashAlgorithm class; DSA class and DSACryptoServiceProvider class; SHA1 class and SHA1CryptoServiceProvider class; TripleDES and TripleDESCryptoServiceProvider class; MD5 class and MD5CryptoServiceProvider class; RSA class and RSACryptoServiceProvider class; RandomNumberGenerator class; CryptoStream class; CryptoConfig class; RC2 class and RC2CryptoServiceProvider class; AssymmetricAlgorithm class; ProtectedData class and ProtectedMemory class; RijndaelManaged class and RijndaelManagedTransform class; CspParameters class; CryptoAPITransform class; Hash-Based Message Authentication Code (HMAC)

- 5) Control permissions for resources by using the System.Security.Permission classes.

SecurityPermission class; PrincipalPermission class; FileIOPermission class; StrongNameIdentityPermission class; UIPermission class; UriIdentityPermission class; PublisherIdentityPermission class; GacIdentityPermission class; FileDialogPermission class; DataProtectionPermission class; EnvironmentPermission class; IUnrestrictedPermission interface; RegistryPermission class; IsolatedStorageFilePermission class; KeyContainerPermission class; ReflectionPermission class; StorePermission class; SiteIdentityPermission class; ZoneIdentityPermission class

- 6) Control code privileges by using System.Security.Policy classes. May include but is not limited to: ApplicationSecurityInfo class and ApplicationSecurityManager class; ApplicationTrust class and ApplicationTrustCollection class; Evidence and PermissionRequestEvidence class; CodeGroup class, FileCodeGroup class, FirstMatchCodeGroup class, NetCodeGroup class, and UnionCodeGroup class; Condition classes; PolicyLevel and PolicyStatement class; IApplicationTrustManager interface, IMembershipCondition interface, and IIdentityPermissionFactory interface

- 7) Access and modify identity information by using the System.Security.Principal classes.

GenericIdentity class and GenericPrincipal class; WindowsIdentity class and WindowsPrincipal class; NTAccount class and SecurityIdentifier class; IIdentity interface and IPrincipal interface; WindowsImpersonationContext class; IdentityReference class and IdentityReferenceCollection class

Implementing interoperability, reflection, and mailing functionality in a .NET Framework application

Expose COM components to the .NET Framework and .NET Framework components to COM.

Import a type library as an assembly; Create COM types in managed code; Compile an interop project; Deploy an interop application; Qualify .NET types for interoperation; Apply Interop attributes, such as the ComVisibleAttribute class; Package an assembly for COM; Deploy an application for COM access.

- 1) Call unmanaged DLL functions within a .NET Framework application, and control the marshalling of data in a .NET Framework application.

Platform Invoke; Create a class to hold DLL functions; Create prototypes in managed code; Call a DLL function; Call a DLL function in special cases, such as passing structures and implementing callback functions; Create a new Exception class and map it to an HRESULT; Default marshalling behavior; Marshal data with Platform Invoke; Marshal data with COM Interop; MarshalAsAttribute class and Marshal class

- 2) Implement reflection functionality in a .NET Framework application, and create metadata, Microsoft intermediate language (MSIL), and a PE file by using the System.Reflection.Emit namespace.

Assembly class; Assembly Attributes; Info classes; Binder class and BindingFlags; MethodBase class and MethodBody class; Builder classes

- 3) Send electronic mail to a Simple Mail Transfer Protocol (SMTP) server for delivery from a .NET Framework application.

MailMessage class; MailAddress class and MailAddressCollection class; SmtpClient class, SmtpPermission class, and SmtpPermissionAttribute class; Attachment class, AttachmentBase class, and AttachmentCollection class; SmtpException class, SmtpFailedReceipientException class, and SmtpFailedReceipientsException class; SendCompletedEventHandler delegate; LinkedResource class and LinkedResourceCollection class; AlternateView class and AlternateViewCollection class

Implementing globalization, drawing, and text manipulation functionality in a .NET Framework application

1) Format data based on culture information.

Access culture and region information within a .NET Framework application; Format date and time values based on the culture; Format number values based on the culture; Perform culture-sensitive string comparison; Build a custom culture class based on existing culture and region classes.

2) Enhance the user interface of a .NET Framework application by using the System.Drawing namespace.

Enhance the user interface of a .NET Framework application by using brushes, pens, colors, and fonts; Enhance the user interface of a .NET Framework application by using graphics, images, bitmaps, and icons; Enhance the user interface of a .NET Framework application by using shapes and sizes.

3) Enhance the text handling capabilities of a .NET Framework application, and search, modify, and control text within a .NET Framework application by using regular expressions.

StringBuilder class; Regex class; Match class and MatchCollection class; Group class and GroupCollection class; Encode text by using Encoding classes.; Decode text by using Decoding classes.; Capture class and CaptureCollection class

TS 70-562: Microsoft .NET Framework 3.5, ASP.NET Application Development

Configuring and Deploying Web Applications

- 1) Configure providers. May include but is not limited to: personalization, membership, data sources, site map, resource, security
- 2) Configure authentication, authorization, and impersonation. May include but is not limited to: Forms Authentication, Windows Authentication
- 3) Configure projects, solutions, and reference assemblies. May include but is not limited to: local assemblies, shared assemblies (GAC), Web application projects, solutions
- 4) Configure session state by using Microsoft SQL Server, State Server, or InProc. May include but is not limited to: setting the timeout; cookieless sessions
- 5) Publish Web applications. May include but is not limited to: FTP, File System, or HTTP from Visual Studio
- 6) Configure application pools.
- 7) Compile an application by using Visual Studio or command-line tools. May include but is not limited to: aspnet_compiler.exe, Just-In-Time (JIT) compiling, aspnet_merge.exe

Consuming and Creating Server Controls

- 1) Implement data-bound controls. May include but is not limited to: DataGrid, DataList, Repeater, ListView, GridView, FormView, DetailsView, TreeView, DataPager
- 2) Load user controls dynamically.
- 3) Create and consume custom controls. May include but is not limited to: registering controls on a page, creating templated controls
- 4) Implement client-side validation and server-side validation. May include but is not limited to: RequiredFieldValidator, CompareValidator, RegularExpressionValidator, CustomValidator, RangeValidator
- 5) Consume standard controls. May include but is not limited to: Button, TextBox, DropDownList, RadioButton, CheckBox, HyperLink, Wizard, MultiView

Working with Data and Services

- 1) Read and write XML data. May include but is not limited to: XmlDocument, XPathNavigator, XPathNodeIterator, XPathDocument, XmlReader, XmlWriter, XmlDataDocument, XmlNamespaceManager
- 2) Manipulate data by using DataSet and DataReader objects.
- 3) Call a Windows Communication Foundation (WCF) service or a Web service from an ASP.NET Web page. May include but is not limited to: App_WebReferences; configuration
- 4) Implement a DataSource control. May include but is not limited to: LinqDataSource, ObjectDataSource, XmlDataSource, SqlDataSource
- 5) Bind controls to data by using data binding syntax.

Troubleshooting and Debugging Web Applications

- 1) Configure debugging and custom errors. May include but is not limited to: ,
- 2) Set up an environment to perform remote debugging.
- 3) Debug unhandled exceptions when using ASP.NET AJAX. May include but is not limited to: client-side Sys.Debug methods; attaching a debugger to Windows Internet Explorer
- 4) Implement tracing of a Web application. May include but is not limited to: Trace.axd, Trace=True on @Page directive,
- 5) Debug deployment issues. May include but is not limited to: aspnet_regiis.exe; creating an IIS Web application; setting the .NET Framework version
- 6) Monitor Web applications. May include but is not limited to: health monitoring by using WebEvent, performance counters

Working with ASP.NET AJAX and Client-Side Scripting

- 1) Implement Web Forms by using ASP.NET AJAX. May include but is not limited to: EnablePartialRendering, Triggers, ChildrenAsTriggers, Scripts, Services, UpdateProgress, Timer, ScriptManagerProxy
- 2) Interact with the ASP.NET AJAX client-side library. May include but is not limited to: JavaScript Object Notation (JSON) objects; handling ASP.NET AJAX events
- 3) Consume services from client scripts.
- 4) Create and register client script. May include but is not limited to: inline, included .js file, embedded JavaScript resource, created from server code

Targeting Mobile Devices

- 1) Access device capabilities. May include but is not limited to: working with emulators
- 2) Control device-specific rendering. May include but is not limited to: DeviceSpecific control; device filters; control templates
- 3) Add mobile Web controls to a Web page. May include but is not limited to: StyleSheet controls; List controls; Container controls
- 4) Implement control adapters. May include but is not limited to: App_Browsers; rendering by using ChtmlTextWriter or XhtmlTextWriter

Programming Web Applications

- 1) Customize the layout and appearance of a Web page. May include but is not limited to: CSS, Themes and Skins, Master Pages, and Web Parts, App_Themes, StyleSheetTheme
- 2) Work with ASP.NET intrinsic objects. May include but is not limited to: Request, Server, Application, Session, Response, HttpContext
- 3) Implement globalization and accessibility. May include but is not limited to: resource files, culture settings, RegionInfo, App_GlobalResources, App_LocalResources, TabIndex, AlternateText , GenerateEmptyAlternateText, AccessKey, Label.AssociatedControlID
- 4) Implement business objects and utility classes. May include but is not limited to: App_Code , external assemblies
- 5) Implement session state, view state, control state, cookies, cache, or application state.
- 6) Handle events and control page flow. May include but is not limited to: page events, control events, application events, and session events, cross-page posting; Response.Redirect, Server.Transfer, IsPostBack, setting AutoEventWireup
- 7) Implement the Generic Handler.

TS 70-564: Pro: Designing and Developing ASP.NET Applications Using the Microsoft .NET Framework 3.5

Designing and Implementing Controls

- 1) Choose appropriate controls based on business requirements.

user controls, server controls, built-in controls, custom controls, third-party controls, Web parts

- 2) Design controls for reusability.

user controls, server controls, inheritance for changing behavior

- 3) Manage states for controls.

control state, view state, accessing form elements

- 4) Leverage data-bound controls.

use gridviews, use sorting and paging callbacks when available, when to use custom sorting and paging, server-side pagination

- 5) Choose appropriate validation controls based on business requirements.

server-side page validation (Page.IsValid), custom validator, validation groups, validation summary

- 6) Identify the appropriate usage of ASP.NET AJAX.

implementing partial page updates with update panel, using ASP.NET AJAX controls, script services

- 7) Manage JavaScript dependencies with server controls.

Designing the Presentation and Layout of an Application

- 1) Design complex layout with Master Pages.

strongly typed master pages, nested master pages

- 2) Plan for various user agents.

markups for different browsers for mobile devices, screen readers, accessibility

- 3) Design a brandable user interface by using themes.

shared themes across multiple applications, run time master page selection

- 4) Design site navigation.

when to extend site map provider, treeview menu vs. site map path, programmatically manipulating site map nodes, overriding menu rendering by using control adapters, filtering site map nodes based on user roles

- 5) Plan Web sites to support globalization.

custom resource provider vs. resource files, localize applications

Accessing Data and Services

- 1) Plan vendor-independent database interactions.

IDBconnection, IDBcommand, IDBadapter, IDataReader, Datareader vs. dataset

- 2) Identify the appropriate usage of data source controls.

SQLDataSource, ObjectDataSource, XMLDataSource

- 3) Leverage LINQ in data access design.

LINQtoSQL, lambda expressions, LINQtoObjects, LINQtoXML

- 4) Identify opportunities to access and expose Web services.

WCF, ASMX, REST

Establishing ASP.NET Solution Structure

- 1) Determine when to use the Web Site model vs. a Web Application Project.

project file, references, namespace, user profile object, precompilation

- 2) Establish an error-handling strategy.

Global.asax events, Web.config elements, TRY/CATCH blocks, error logging

- 3) Manipulate configuration files to change ASP.NET behavior.

machine key, tracing, encrypting Web configuration data, custom configuration sections

- 4) Identify a deployment strategy.

management application pools, Web deployment projects, pre-compilation, custom action classes

Leveraging and Extending ASP.NET Architecture

- 1) Design a state management strategy.

Cache, ViewState, Application object, Session object, cookies, cookieless session

- 2) Identify the events of the page life cycle.

appending controls,PostBack model, accessing state, data binding

- 3) Write HttpModules and HttpHandlers.

URL rewriting, SSO application, dynamically retrieve data

- 4) Debug ASP.NET Web applications.

debug JavaScript, tracing, debug tools in IDE, examining HTTP headers

- 5) Plan for long-running processes by using asynchronous pages.

AddonPreRenderCompleteAsync, RegisterAsyncTask

Applying security principles

- 1) Identify appropriate security providers.

membership, role, profile, extending custom providers

- 2) Decide which user-related information to store in a profile.

create user profile properties, extend membership objects, custom types

- 3) Establish security settings in Web.config.

identity/impersonation, authentication, authorization (location nodes in Web.config)

4) Identify vulnerable elements in applications.

SQL injection, cross-site scripting, protecting against bots

5) Ensure that sensitive information in applications is protected.

hash and salt passwords, encrypting information