Programación en Visual Basic con Microsoft Visual Studio 2010

Duración: 5 Días  Código del Curso: M10550  Versión: A

Temario:

Este curso enseña el lenguaje de programación Visual Basic, la estructura del programa y la implementación utilizando Microsoft Visual Studio 2010 y el marco de trabajo .NET Framework 4. Este curso proporciona una sólida base en Visual Basic a un nivel necesario para permitir a los estudiantes asistir a otros cursos en los encajes técnicos de especialistas.

Dirigido a:

Este curso está destinado a desarrolladores experimentados que ya tienen experiencia en programación en Visual Basic, C, C++, C#, o Java, y comprenden los conceptos de Programación Orientada a Objetos. Estos desarrolladores serán los que asistan al curso para que puedan rápidamente adquirir habilidades de programación en Visual Basic en el marco de trabajo .NET.

Objetivos:

- Después de completar este curso, los estudiantes serán capaces de:
  - Describir cómo usar la herencia para crear nuevos tipos de referencia.
  - Describir el propósito del marco de trabajo .NET, y explicar cómo manejar la vida útil de los objetos y controlar el uso de recursos.
  - Describir cómo crear y llamar métodos.
  - Describir cómo detener, manejar y lanzar excepciones.
  - Describir cómo realizar operaciones de entrada/salida en un programa de Visual Basic.
  - Describir cómo crear y usar nuevos tipos (enumeraciones, clases, estructuras), y explicar cómo hacer entre tipos de referencia y tipos de valor.
  - Describir cómo controlar la visibilidad y vida útil de los miembros de un tipo.
  - Describir cómo usar herencia para crear nuevos tipos de referencia.
  - Describir cómo manejar la vida útil de los objetos y controlar el uso de recursos.
  - Describir cómo crear y llamar métodos.
  - Describir cómo manejar excepciones.
  - Describir cómo gestionar la vida útil de objetos y controlar el uso de recursos.
  - Describir cómo implementar clases de colecciones personalizadas que soportan el tipo seguro.
  - Describir cómo implementar clases de colecciones personalizadas que soportan el tipo seguro.
  - Describir cómo manejar operaciones asincronas.
  - Describir cómo integrar código escrito usando un lenguaje dinámico como Ruby o Python, o tecnologías como Component Object Model (COM), en un programa de Visual Basic.

Prerrequisitos:

Este curso requiere que cumplan con los siguientes requisitos:

- Este curso está destinado a desarrolladores que ya tienen conocimientos de Visual Basic.
- Este curso no está destinado a nuevos desarrolladores; por lo menos 12 meses de experiencia trabajando con un lenguaje orientado a objetos es esperado.
- Experiencia con el marco de trabajo .NET Framework.
- Conocimiento del desarrollo integrado de Visual Studio.
environment (IDE).
<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module 1: Introducing Visual Basic and the .NET Framework</strong>&lt;br&gt;Introduction to the .NET Framework 4&lt;br&gt;Creating Projects Within Visual Studio 2010&lt;br&gt;Writing a Visual Basic Application&lt;br&gt;Building a Graphical Application&lt;br&gt;Documenting an Application&lt;br&gt;Debugging Applications by Using Visual Studio 2010</td>
<td></td>
</tr>
<tr>
<td><strong>Module 2: Using Visual Basic Programming Constructs</strong>&lt;br&gt;This module introduces many of the basic Visual Basic language data types and programming constructs, and describes the syntax and semantics of these constructs.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 3: Declaring and Calling Methods</strong>&lt;br&gt;A key part of developing any application is dividing the solution into logical components. In object-oriented languages such as Microsoft Visual Basic, a method is a unit of code that is designed to perform a discrete piece of work. This module introduces methods and describes how to define and use them.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 4: Handling Exceptions</strong>&lt;br&gt;Exception handling is an important concept and your applications should be designed with exception handling in mind. This module explains how you can implement effective exception handling in your applications, and how you can use exceptions in your methods to elegantly indicate an error condition to the code that calls your methods.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 5: Handling Events</strong>&lt;br&gt;This module explains how to use events in your applications, and introduces event handling in Visual Basic.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 6: Creating New Types</strong>&lt;br&gt;The Microsoft .NET Framework base class library consists of many types that you can use in your applications. However, in all applications, you must also build your own types that implement the logic for your solution. This module explains how to create your own modules and types and describes the differences between reference types and value types.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 7: Encapsulating Data and Methods</strong>&lt;br&gt;This module describes how to use some of the access modifiers that Visual Basic provides to enable you to implement encapsulation. This module also introduces the Shared modifier, which enables you to define members that can be shared over multiple instances of the same type.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 8: Inheriting from Classes and Implementing Interfaces</strong>&lt;br&gt;This module introduces inheritance and interfaces in the Microsoft .NET Framework, and how you can use them to simplify complex problems, reduce code duplication, and speed up development. Inheritance is a key concept in an object-oriented language. You can use inheritance, interfaces, and abstract classes to develop object hierarchies in your code. These object hierarchies can help reduce bugs by defining clear contracts for what a class will expose and by providing default implementations where you can sensibly abstract code into a base type.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 9: Declaring and Using Delegates</strong>&lt;br&gt;Delegates are a mechanism in the Microsoft .NET Framework that enables you to implement decoupled operations. This module explains how to use delegates to inform consuming applications of a change or notable occurrence in a type.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 10: Decoupling Methods and Handling Events</strong>&lt;br&gt;This module explains how to decouple an operation from the method that implements it and how to use anonymous methods to implement decoupled operations. This module also explains how to use events to inform consuming applications of a change or notable occurrence in a type.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 11: Using Collections and Building Generic Types</strong>&lt;br&gt;The basic collection classes introduce a new problem. Classes that act on other types are often not type-safe. For example, many collection classes frequently use the Object type to store items, and must then be cast or converted back to their original type before they can be used. It is the programmer’s responsibility to ensure that the correct casts or conversions are performed, and it is easy to introduce errors by casting or converting an item to the wrong type. This module introduces generics and how you can use generic classes to maintain type-integrity and avoid issues that are associated with a lack of type safety.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 12: Using Collections and Building Custom Collection Classes</strong>&lt;br&gt;When you develop applications, you often need to store collections of objects. In many circumstances, you can use the collection classes that the Microsoft .NET Framework includes; however, sometimes these collection classes do not provide the functionality that you require. For example, you may need to store objects in a sorted order that is based on a custom sorting algorithm. This module introduces you to custom collection classes. It also explains how you can develop collection classes that support the language constructs that Visual Basic provides, such as enumeration and collection initialization.</td>
<td></td>
</tr>
<tr>
<td><strong>Module 13: Using Collections and Building Custom Collection Classes</strong>&lt;br&gt;When you develop applications, you often need to store collections of objects. In many circumstances, you can use the collection classes that the Microsoft .NET Framework includes; however, sometimes these collection classes do not provide the functionality that you require. For example, you may need to store objects in a sorted order that is based on a custom sorting algorithm. This module introduces you to custom collection classes. It also explains how you can develop collection classes that support the language constructs that Visual Basic provides, such as enumeration and collection initialization.</td>
<td></td>
</tr>
</tbody>
</table>
Module 5: Reading and Writing Files
The ability to access and manipulate the files on the file system is a common requirement for many applications. This module shows how to read and write to files by using the classes in the Microsoft .NET Framework. This module also describes the different approaches that you can take, and how to read and write different formats of data.

Module 9: Managing the Lifetime of Objects and Controlling Resources
All applications use resources. When you build a Microsoft Visual Basic application, resources fall into two broad categories: managed resources that are handled by the common language runtime (CLR) and unmanaged resources that are maintained by the operating system outside the scope of the CLR. A managed resource is typically an object based on a class defined by using a managed language, such as Visual Basic. Examples of unmanaged resources include items implemented outside the Microsoft .NET Framework, such as Component Object Model (COM) components, file handles, database connections, and network connections.

Resource management is important in any applications that you develop. The .NET Framework simplifies resource management by aut

Module 10: Encapsulating Data and Defining Overloaded Operators
Many operators have well-defined behavior for the built-in Visual Basic types, but you can also define operators for your own types. This module describes how to implement operators for your types by using overloading.

Introduction to Garbage Collection
Managing Resources

Module 14: Using LINQ to Query Data
This module introduces you to Language-Integrated Query (LINQ) queries and explains how you can use them to process data in your Microsoft .NET Framework applications. This module also explains the difference between shared and dynamic LINQ queries, and describes how you can use dynamic LINQ to create highly flexible queries that you build at run time.

Module 15: Integrating Visual Basic Code with Dynamic Languages and COM Components
Integration with other technologies is a key feature of the Microsoft.NET Framework. Previous versions of the .NET Framework enabled you to combine components that were developed by using different languages that have compilers that the .NET Framework supports. The .NET Framework 4 now supports integration of components built by using dynamic languages. This enables you to re-use items built by using a wide range of scripting languages that are not easily accessible from Microsoft Visual Basic code. In addition, previous versions of the .NET Framework have always enabled you to integrate Component Object Model (COM) services and components into your managed applications. The integration did however, require a good understan

Integrating Visual Basic Code with Ruby and Python
Accessing COM Components from Visual Basic

Más información:
Para más información o para reservar tu plaza llámanos al (34) 91 425 06 60
info.cursos@globalknowledge.es
www.globalknowledge.es
Global Knowledge Network Spain, C/ Retama 7, 6ª planta, 28045 Madrid