Chapter 1 : Server, Development & DBMS Software for sale | eBay

Download visual basic enterprise edition for free. System Utilities downloads - Visual Basic Enterprise Edition by Microsoft and many more programs are available for instant and free download.

Programmers can create both simple and complex GUI applications. Programming in VB is a combination of visually arranging components or controls on a form , specifying attributes and actions for those components, and writing additional lines of code for more functionality. Since VB defines default attributes and actions for the components, a programmer can develop a simple program without writing much code. Programs built with earlier versions suffered performance problems, but faster computers and native code compilation has made this less of an issue. Core runtime libraries are included by default in Windows and later, but extended runtime components still have to be installed. An empty form in Visual Basic 6 Forms are created using drag-and-drop techniques. A tool is used to place controls e. Controls have attributes and event handlers associated with them. Default values are provided when the control is created, but may be changed by the programmer. Many attribute values can be modified during run time based on user actions or changes in the environment, providing a dynamic application. For example, code can be inserted into the form resize event handler to reposition a control so that it remains centered on the form, expands to fill up the form, etc. By inserting code into the event handler for a keypress in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being inserted. Dialog boxes with less functionality can be used to provide pop-up capabilities. Controls provide the basic functionality of the application, while programmers can insert additional logic within the appropriate event handlers. For example, a drop-down combination box automatically displays a list. When the user selects an element, an event handler is called that executes code that the programmer created to perform the action for that list item. This allows for server-side processing or an add-in module. The runtime recovers unused memory using reference counting , which depends on variables passing out of scope or being set to Nothing, avoiding the problem of memory leaks that are possible in other languages. There is a large library of utility objects, and the language provides basic support for object-oriented programming. Unlike many other programming languages, Visual Basic is generally not case-sensitiveâ€”though it transforms keywords into a standard case configuration and forces the case of variable names to conform to the case of the entry in the symbol table. String comparisons are case sensitive by default. Nevertheless, by default the restrictions in the IDE do not allow creation of some targets Windows model DLLs and threading models, but over the years, developers have bypassed these restrictions. Versions since at least VB 3. The result stored in A would therefore be either false or true. This inherent functionality becomes especially useful when performing logical operations on the individual bits of an integer such as And, Or, Xor and Not. Logical and bitwise operators are unified. This is unlike some C-derived languages such as Perl , which have separate logical and bitwise operators. Arrays are declared by specifying the upper and lower bounds in a way similar to Pascal and Fortran. It is also possible to use the Option Base statement to set the default lower bound. Use of the Option Base statement can lead to confusion when reading Visual Basic code and is best avoided by always explicitly specifying the lower bound of the array. This lower bound is not limited to 0 or 1, because it can also be set by declaration. In this way, both the lower and upper bounds are programmable. In more subscript-limited languages, the lower bound of the array is not variable. This uncommon trait does exist in Visual Basic. Relatively strong integration with the Windows operating system and the Component Object Model. By default, if a variable has not been declared or if no type declaration character is specified, the variable is of type Variant. There are 12 Deftype statements in total offered by Visual Basic 6. The default type may be overridden for a specific declaration by using a special suffix character on the variable name for Double,! VB can also be set in a mode that only explicitly declared variables can be used with the command Option Explicit. History[edit] Alan Cooper created the drag-and-drop design for the user interface of Visual Basic. The drag and drop design for creating the user interface is derived from a prototype form generator developed by Alan Cooper and his company called Tripod. Tripod did not include a programming language at all. Microsoft decided to combine Ruby with the

Basic language to create Visual Basic. Ruby also provided the ability to load dynamic link libraries containing additional controls then called "gizmos" , which later became the VBX interface.

Chapter 2 : Best Practices for Rule-Based Application Development

Discuss: Microsoft Visual Basic Enterprise Edition (v.) - media Sign in to comment. Be respectful, keep it clean and stay on topic. We delete comments that violate our policy, which we.

With Enterprise Server, Micro Focus has created one of the largest Mainframe rehosting platforms available. Enterprise Server runs on either Windows or Linux. For this walk-through, we will use a Windows Server virtual machine that comes with Visual Studio installed. Before we get started, check out these prerequisites: Get the bits from Micro Focus. If you are an existing Micro Focus customer, contact your Micro Focus representative. Otherwise, you can request a trial. For more information, see the documentation for Enterprise Server and Enterprise Developer. To run these two environments, you need a valid license or trial license from Micro Focus. You will obviously need an Azure subscription. A best practice is to set up a site-to-site virtual private network VPN tunnel or a jumpbox so you can control access to the Azure virtual machines. There are a number of reasons for doing this, all centered around security and manageability. Another tip is to use the first part of the name in Azure to designate the type of resource. It makes it easier to spot in a list. Create a virtual machine. From the Azure Portal Marketplace, select the virtual machine and operating system you want. I used the following virtual machine SKUs: This image is available from the Azure Marketplace. As above, this image is available from the Azure Marketplace. On the Basics blade, enter your Username and Password. You can accept the defaults for the rest of the settings. Remember the user ID and password you create for the administrator of these virtual machines. To log on to the Enterprise Server virtual machine, select the virtual machine from the portal. Log on using the credentials you created for the virtual machine. From the RDP session, load the following two files. Since this is Windows, so you can drag and drop the files into the RDP session: Double-click the file to start the installation. In the first window, select the installation location and accept the end user license agreement. When Setup is complete, the following message appears: Next, start the Micro Focus License Administration. There are two types of license formats: In my case, I had a file. For License file, browse to the mflic file uploaded previously to the virtual machine and select Install Licenses. Enterprise Server can now load. You should see the Enterprise Server Administration Page.

Chapter 3 : Microsoft Visual Studio Enterprise Edition VS 6 Development System | eBay

Because many of the problems companies have with VB development stem from misunderstandings, Real Visual Basic addresses and dispels myths of corporate software development, particularly those that relate to Visual Basic.

Collapse the table of content Expand the table of content This documentation is archived and is not being maintained. This documentation is archived and is not being maintained. NET is a general purpose development platform. It can be used for any kind of app type or workload where general purpose solutions are used. It has several key features that are attractive to many developers, including automatic memory management and modern programming languages, that make it easier to efficiently build high-quality apps. NET are available, based on open. NET Standards that specify the fundamentals of the platform. NET Framework, which is one of the existing. This topic will also talk about other. NET technologies and where you can find their related documentation. It can be installed locally with your app with only the packages you need. It provides a lightweight development model and the flexibility to work with your favorite development tools on your favorite development platform. NET Core page to find installation instructions for each supported platform. You can currently use. NET Core to develop console or Web applications: To read about developing modern cloud-based Web application, see the ASP. To read about working with data, see the Entity Framework documentation. NET Framework, see the overview. We update the latest versions of the. NET Framework documentation on a regular basis with content fixes and enhancements, but we do not maintain older versions. Earlier versions of the. NET Framework documentation are available from the table of contents pane on the left. You can use the. To read about creating Windows 8. For information about creating portable. For additional information about developing apps, visit the Windows desktop apps and web development sections of the MSDN Library. You can use Visual Studio for your development tasks and select from a wide range of programming languages. NET Framework also releases out-of-band packages with new functionality and improved cross-platform support. For information about these, see The. You can extend the capabilities of your apps with the following.

Chapter 4 : Download visual basic enterprise edition for free (Windows)

Hello, i would like to get more iformation about Vbasic, because I don't know is this ver.-Visual Basic Enterprise Edition, better then Standart or is not it demo or something like that.

Model-Driven Development represents the next logical step forward in software development methods and practices. It aims to facilitate the automatic construction of a software solution from a high-level domain-specific specification. This approach seeks to promote productivity, maintainability, expressiveness, and to aid in the management of complexity by supporting higher levels of abstraction and the systematic reuse of domain-specific assets. In this article, we explore the role of domain-specific languages and Visual Studio Domain Specific Language Tools DSLs and show how they can be used to automate the generation of an enterprise application for the. Introduction Over the last 30 years, the information technology industry has experienced constant and rapid advances in computing devices, software products and technologies, and changing requirements as consumers find new ways to use this technology. Subsequently, advances in language paradigms, software design practices, and programming environments have been driven by the need to support reuse, adaptability, and the management of complexity within these software applications. These software characteristics are critical in order to support higher levels of agility, productivity, quality, maintenance, and evolution. Like other engineering disciplines, the software development industry has used modeling as the basis to capture and communicate requirements, analysis, and design with its key stakeholders. However, often after the initial development of the software application these models are not maintained and the "code" becomes the model of the application. Furthermore, the analysis and design patterns contained within these models are not captured, standardized, and packaged for reuse across multiple software projects within the same or a similar application domain. Model-driven development attempts to address these issues by enabling reusable application domain knowledge, captured during domain engineering, to be represented in domain specific languages. These languages are meta-models that specify the application domain-modeling paradigm, including key concepts, relationships, constraints, and the rules governing code generation. Domain-specific languages can be represented either textually or visually within a domain model. Thus, they can be interpreted by humans, promoting communication about the intent of the software application, and processed by tools to facilitate the automated generation of the application. This approach supports improved project planning, promotes communication, increases quality, and reduces the risk and cost typically associated with software development. Visual Studio Domain Specific Language Tools DSLs enable system integrators to extend the modeling platform to build custom visual designers for specific domains that are hosted within Visual Studio EDS have used this technology to create a visual designer for modeling the business entities within an enterprise application. Based on this model, a code generation framework manages the transition from the business entities in the model to the object-oriented C implementation of the entities, data services, database, and unit tests for the application. The major advantage of this approach is EDS has been able to package knowledge about the construction of business entities for reuse in patterns, languages, and frameworks that can used within Visual Studio NET application developers can therefore leverage development tools that they are experts in while increasing productivity and quality through the use of the customized visual entity designer and the code generation framework. The base constructs of the UML meta-model, such as classes, association, generalization, and so on, have little semantic meaning. Thus, in order to interpret and perform meaningful computations on a model the base constructs must be extended with domain-specific information. This approach provides the necessary semantic meaning and the constraints necessary to support meaningful automated code generation. The Domain Concept editor in the Domain Specific Languages Tools supports a UML-like notation that allows a software architect to perform this task by defining the domain-specific concepts, their attributes, and relationships between them. The domain model is persisted in an XML format. The domain model is used to generate a domain-specific API and to extend the Visual Studio development environment with the custom designer. This approach allows a software architect to create reusable domain-specific languages that can capture reusable analysis patterns and standards, such as the Healthcare HL7 standard, Sarbanes Oxley, and so on. The main concepts included within this domain model are: Each of these concepts has been configured with a set of associated values, such as package namespace, entity name, database tablename, and so on. The key relationships within the domain model are generalization and association, which includes aggregation and composition. Domain Concept Editor for entity domain model click for larger image The Domain Specific Languages Designer Definition in the Domain Specific Languages tools is used with a domain model to both create the visual notation for the custom designer and to bind the visual elements to the underlying domain models in-memory store. The designer definitions are edited and persisted in XML format. The designer definition file allows the software architect to define the visual shapes that represent a domain concept, the connectors to represent relationships, constraints on the connectors that restrict what visual shapes can be connected, and the toolbox used within the designer. This example illustrates the use of constraints to define visual shapes that can be connected. Specifically, we have constrained the software developer to only be able to use an aggregate relationship connector between entity objects. Each of these elements has been customized based on our defined domain model and designer definition. The custom entity model shown in Figure 2 illustrates a small subset of the business entities typically found within an order management system. Using the entity designer, the software developer can configure the business entities, and their properties and relationships, as required to meet their specific design requirements. The code generation framework accepts as input an XML representation of the model that is used to guide the automated generation of software artifacts, such as classes, database definitions, and so on, using templates. NET like-syntax and transformation engine to enable software architects to generate text-based artifacts. Entity Designer click for larger image Code Generation Framework EDS adopted a model-driven approach to software development approximately four years ago. The ADE represents a holistic approach to the software development process. It incorporates a highly integrated set of standard processes and tools for requirements gathering, analysis, design, testing, construction and deployment. This includes a base set of framework libraries, a code generation framework, and modeling tools that are hosted within Visual Studio. As illustrated in Figure 3 the code generation framework supports the transformation of a model to software artifacts using the following approach: This is typically initiated from within the application development environment. The code generation framework deserializes the XML file into an "in-memory store" representation. The code generation framework transforms the model into software artifacts that reflect the modeled requirements. The transformation uses text templates to generate software artifacts that use the ADE framework libraries. The text templates encapsulate best practices and patterns for architecting a. Code Generation Framework overview click for larger image The code generation framework also provides differencing tools to help the software developer manage changes in the model and in the previously generated software artifacts when they are regenerating a model. The major advantage of this is a software developer can perform code generation for a model while preserving changes that they may have manually made to previously generated software artifacts. When this model is regenerated the software developer will be notified that there are differences between the newly generated source code generated from the model, compared to the updated source code that contains the new business validation function. The software developer can then use differencing tools to manage the changes prior to completing the code generation process. Upon completion of the code generation process the software artifacts are automatically moved into a Visual Studio solution. Layer Description Entities Contains public interface and implementation for the modeled entities. Optionally, a public interface and implementation class can also be created to manage the entity as a collection. The data adapters are not exposed outside of the data services layer and are invoked by the Data Services classes. The Data Services classes are responsible for managing persistency and concurrency for a model entity. The DDL for the creation of a modeled entity table includes columns for the attributes model on the entity, a primary key and a concurrency stamp to support optimistic locking. The data layer can also be optionally populated with random data using data scripts created during code generation. Building an Enterprise Application with Entity Designer In the sample model illustrated in Figure 4, we have a simple order management system with three business entities: Customer, PurchaseOrder, and CorporateCustomer. A Customer is associated with a one-to-many PurchaseOrder. A CorporateCustomer is derived from the Customer. This model can be easily extended to include new business entities, relationships, and data services using the standard toolbox, and it can be configured using the properties window. Orders Management System entity model click for larger image When software developers have finished designing the model they can initiate code generation using a Visual Studio plugin that provides access to the EDS code generation framework. The output of this code generation framework for the model illustrated, listed below, includes approximately 40 interfaces and classes and 24 database definitions and stored procedures. An additional script is also generated to create a foreign key relationship to implement the modeled association between a Customer and a Purchase Order in the database. Data adapters are also created for each of the entities to support fetching and updating the data to and from the database. The automated generation of the test cases supports the use of test-driven development. This approach is emerging as one of the most successful productivity and quality enhancement techniques to be recently adopted by the software industry and supported in Visual Studio Order Management Visual Studio unit tests click for larger image Conclusion The Microsoft Visual Studio and Domain Specific Language tools provide the modeling capabilities for architects to develop domain-specific designers. This approach enables EDS to increase productivity and quality, and reduce cost in the delivery of client solutions through the reuse of assets that are based on best practices, patterns, and frameworks. NET model-driven development tools and frameworks. Her main interests include model-driven development, software architecture and patterns, software reliability, and software methodologies. His main interests include model-driven development, software methodologies, software architecture, and patterns and mobile computing.

Chapter 5 : Model-Driven Development of .NET Enterprise Applications

Visual Basic Delivers up to 2, Percent Performance Gains Over Visual Basic Visual Basic delivers unmatched performance gains over any rapid application development tool in its class. Microsoft has included several key features in version that deliver performance gains of up to 2, percent over version , including native.

This paper takes a high level view of knowledge, using the word in its more general sense, rather than as a specific technical term, and then looks at different types of knowledge and their mappings to executable computer code. The purpose is to gain insights into when and why rule engines provide advantages over conventional software development tools. The three types of knowledge considered are factual, procedural, and logical. These divisions correspond to the capabilities of computers. Factual Knowledge Factual knowledge is just that, facts, or data. It can be facts about customers, orders, products, or the speed of light. Computers have memory and external storage devices. These are ideally suited to the storage and retrieval of factual knowledge. Database tools and programming languages that manipulate memory have evolved naturally from these basic components of machine architecture. Factual knowledge appears in the computer as either elements in a database or variables and constants in computer programs, as shown in Figure 1. Factual Knowledge Procedural Knowledge Procedural knowledge is the knowledge about how to perform some task. It can be how to process an order, search the Web, or calculate a Fourier transform. Computers have a central processing unit CPU that processes instructions one at a time. This makes a computer well-suited to storing and executing procedures. Programming languages that make it easy to encode and execute procedural knowledge, have evolved naturally from this basic computational component. Procedural knowledge appears in a computer as sequences of statements in programming languages, as shown in Figure 2. Procedural Knowledge Logical Knowledge Logical knowledge is the knowledge of relationships between entities. It can relate a price and market considerations, a product and its components, symptoms and a diagnosis, or the relationships between various tasks. Unlike for factual and procedural knowledge, there is no core architectural component of a computer that is well suited to the storage and use of logical knowledge. Typically, there are many independent chunks of logical knowledge that are too complex to store in a database, and lack an implied order of execution which makes them ill-suited for programming. Logical knowledge does not map well to computer architecture Specialized tools, which are effectively virtual machines better suited to logical knowledge, can often be used instead of conventional tools as shown in Figure 4. Rule engines and logic engines are two examples. Using virtual machines for logical knowledge Conventional vs. Consider, for example, a pricing module for phone calls or airline seats, or an order configuration module. Furthermore, logical knowledge is often changing. Government regulations are expressed as logical knowledge, as are the effects of changing market conditions. Business rules that drive an organization are almost always expressed as logical knowledge. Because of the critical role logical knowledge can play, there are good arguments for using specialized tools which make the encoding of logical knowledge quicker, easier and more reliable. There are also, however, good arguments against them, foremost being the ready pool of talent that is familiar with conventional tools. There is a lot to be said for sticking with the familiar, although in general the cost is lengthy development times, tedious maintenance cycles, a higher than normal error rate, and often compromises in the quality of service the application provides. On the other hand, there are some well known problems with rule engines and other tools designed for working with logical knowledge: There are many choices, and they are usually vendor specific. Each tool is better suited for some types of logical knowledge than other types. Rules that diagnose a fault need to behave different from rules that calculate a price, which in turn behave different from rules that dictate how an order can be configured. Maintenance is not as easy as sometimes promised. Furthermore, because there is no order to the rules, tracking interrelationships can be difficult. There is no standard application program interface API for integrating a rule engine with other components of an application. Given these difficulties, is the payoff from using rule-based tools worth the investment? In many cases, yes! For example, one organization that provides online mortgage services replaced lines of procedural code to price mortgages, with lines of logical rules. The logic-based solution was implemented in two months as opposed to the year invested in the original module. Maintenance turn around due to changing market conditions was reduced from weeks to hours, and the errors in the resulting code went to practically zero. There is no tricky translation from business specification to ill-suited procedural code. The biggest win of all might be the flexibility the logic-based solution provided them with, allowing them to expand their product offerings. They could now offer more and better mortgage pricing options for their customers, including the option of customizing the pricing logic for each institutional customer. This is not an uncommon story. The same benefits and 10 to 1 improvement ratio appear over and over again in the success stories of vendors of rule-based and logic technologies. A semantic gap refers to the difference between the way knowledge to be encoded in an application is naturally specified and the syntax of the computer language or tool used to do the encoding. For example, you can use assembler to code scientific equations. But it is tedious and error-prone because there is a large semantic gap between the syntax of assembler and an equation. It allowed a programmer to code an equation in a way that was much closer to the way a scientist might write the equation on paper. The result was easier, quicker coding of engineering and scientific applications, and fewer errors. Factual knowledge and procedural knowledge are both readily coded in computers because there is a reasonably small semantic gap between the way facts and procedures are described and the tools for encoding them. As pointed out previously, this is because computers are inherently good at facts and procedures. The semantics of logical knowledge however does not map readily to conventional tools. Consider this piece of knowledge: The meaning, or semantics, of this knowledge is best captured in a pattern-matching sense. It really means that the details of a proposed trip should be matched against the conditions in the rule, and the appropriate rule should be used to determine the fare. This sort of knowledge could be shoehorned into procedural code, but the semantics of procedural code are designed to express a sequence of operations, not a pattern-matching search. On the other hand, a rule engine is designed to interpret rules in a pattern-matching sense, so rules entered in such a tool will have a smaller semantic gap than rules encoded procedurally. In fact, not only is it tempting, it can work reasonably well up to a point. However there is a big difference between a logical relationship and a procedural if-then. A procedural if-then is really a branching statement, controlling the flow of execution of a procedure. If the condition is true, control goes one way, and if not control goes a different way. A logical relationship can be coded as a procedural if-then, but must be placed somewhere along the road of execution of the procedure it is in. Furthermore, if there are more logical relationships, they too must be placed at some point in the procedural pathâ€”and, by necessity, the placement of one affects the behaviour of another. It makes a difference which rule gets placed first, and if there are branches from previous rules, and which branch a following rule is placed on. This is not a problem if the rules map easily to a decision tree, but in that case the knowledge is really procedural. The arbitrarily imposed thread of execution that links the various rules becomes extremely tangled, making the code difficult to write in the first place, and very difficult to maintain. However, the module with the rules is often the most troublesome module in a system. Once encoded procedurally, logical knowledge is no longer easily accessible; that is, it no longer looks like a collection of rules and declarative relationships. The knowledge resource has, in a sense, been lost and buried in the code, just as a scientific equation can no longer be read if it is coded in assembler. The same is not true of either factual or procedural knowledge. In those cases, reading the code generally does show the underlying knowledge. Databases for Rules It is possible, in some cases, to shoehorn logical relationships into a database. If the relationships can be represented in a tabular form, then a database table can be used to encode the rule. So for example, if the amount of discount a customer got was dependent on the amount of previous sales at a few different levels, this could be represented as a table and stored in a database. However, as with the using procedures, the database approach is limited in that it only works for very clean sorts of logical relationships. A Mixed Approach Sometimes applications use a mixture of both the procedural and database approaches. Logical relationships that can be expressed in tables are stored in a database, and the remaining relationships are coded as procedural if-then statements. This can simplify the coding task, but it makes maintenance harder because the logical knowledge is now spread across two different vehicles. Despite these difficulties, there is a strong appeal to using data, procedure or both to encode logical knowledge, and that is that they are familiar techniques, and there are numerous individuals skilled in their use. Artificial Intelligence The problems with encoding logical relationships were first explored back in the s by researchers at Stanford University. They were trying to build a system that advised physicians on courses of antibiotics for treating bacterial infections of the blood and meningitis. They found that the medical knowledge consists mainly of logical relationships that can be expressed as if-then rule. They attempted many times to encode the knowledge using conventional tools, and failed because of the problems described previously. If the problem with coding logical knowledge is that the nature of a computer is not well-suited to expressing logical relationships, then clearly the answer is to create a machine that is. Building specialised hardware is not very practical, but it turns out a computer is a good tool for creating virtual computers. This is what the researchers at Stanford did. They effectively created a virtual machine that was programmed using logical rules. This type of virtual machine is often called a rule engine. Why is a computer good at building a rule engine, but not the rules themselves? It is because behaviour of a rule engine can be expressed in a procedural algorithm, along the lines of: However the real reason for the growth of the term was pure and simple marketingâ€"it was easier to get Department of Defence funding for advanced research on Artificial Intelligence AI than it was for heuristic programming. Those companies that survived and continue to market and sell the technology have found the term AI to be a detriment, so they looked for a different term. Now it is most often called rule-based programming. Whether you call it heuristic programming, Artificial Intelligence, expert systems, or business rule processing, the underlying technology is the sameâ€"a virtual engine that uses pattern-matching search to find and apply the right logical knowledge at the right time. Other Logical Virtual Engines Virtual engines programmed with declarative rules are not the only example of specialized software designed to deal with logical knowledge. Other such programs dramatically altered the course of the history of computing.

Chapter 6 : The Microsoft Dynamics AX Enterprise Portal Blog

Find helpful customer reviews and review ratings for Enterprise Development Using Microsoft Visual Basic (Microsoft Mastering) at calendrierdelascience.com Read honest and unbiased product reviews from our users.

What is the Lifecycle Policy for the Microsoft Silverlight product? Due to the way Microsoft Silverlight is released it does not take a Lifecycle typical of many Microsoft products and, for clarification purposes, is defined as a tool. A tool is a utility or feature that aids in accomplishing a task or set of tasks. Tools are supported with the product they work with. Some tools may receive security updates or feature enhancements. Microsoft will continue to support Silverlight by shipping updates to the latest version of Silverlight runtime. Updates and new versions of the Silverlight runtime are backward compatible with web applications built in previous versions of Silverlight and will include the latest security enhancements, performance improvements, and product fixes. What version of Silverlight is currently supported? Microsoft will provide technical help, paid and unpaid, for customers using versions of Silverlight 5. Paid technical help is available to customers requiring support with issues beyond install and upgrade. Silverlight 5 will support the browser versions listed on this page through October 12, , or though the lifecycle of the underlying browsers, whichever is shorter. As browsers evolve, the support page will be updated to reflect levels of compatibility with newer browser versions. Visual Studio follows the established Microsoft Lifecycle Policy of a minimum of five years Mainstream Support and a minimum of five years Extended Support. Details are described here. Online services that are offered as part of the Visual Studio suite of products will follow the established Modern Policy found here. What lifecycle policy do external components follow that are offered within the Visual Studio suite? Visual Studio includes a collection of compilers, languages, runtimes, environments, and other resources that enable development for many Microsoft platforms. These components may be licensed and supported under their own terms and policies. What lifecycle policy does Visual Basic 6. Microsoft is committed to support existing Visual Basic 6. As detailed here , the core Visual Basic 6. Return to this site periodically to review any such changes.

Chapter 7 : Set up Micro Focus Enterprise Server and Enterprise Developer in Azure â€“ AzureCAT Guida

Visual Basic goes fully bit, dropping all support for bit Windows It can create bit applications for NT and Windows This was a fairly stable and popular release, but quickly followed up by VB 6.

Buying Visual Basic Introduction Visual Basic has been around for a long time and has been growing in popularity all the while. The word "Basic" in Visual Basic may be misleading. However, this is not the case. You can write advanced Windows programs in a fraction of the time it takes to write the same programs using other programming languages. Visual Basic as the name implies is a Windows programming language. With Visual Basic, if you want to create a button, a list box or text box etc. Items placed on the form have their underlying code written into the program along with their properties. Each item placed on the form can have its properties adjusted at design time. Items can also have most of their properties adjusted at run time. A simple program can be created with very little additional code. The drag and drop interface allows you to create simple programs easily. If the programmer wants to develop a program using a feature not available from the tool box, then the use of a DLL will almost certainly provide the answer. Visual Basic 5 comes in three flavours: The Professional edition contains everything in the Learning edition plus additional ActiveX controls including Internet controls and Crystal Report Writer. The Enterprise edition contains everything in the Professional edition plus the Automation Manager, the Component Manager, database management tools and Microsoft Visual SourceSafe project orientated version control system. It should be noted that Visual Basic 5 is a bit only program and will not run in Windows 3. It will only run in Windows 95 and Windows NT or greater. When you have finished your masterpiece it can be compiled into an executable program or EXE. This EXE can then be distributed, however, it requires quite a few additional files distributing with it. There is a program provided within Visual Basic called a Set-up Wizard which will guide the user through the process of preparing the program for distribution. The Setup Wizard automatically creates an installation routine for your project. Visual Basic 5 has a native code compiler, which they claim allows programs to run up to 20 times faster than they would with previous versions of Visual Basic. You can select between optimisation for speed or size. An ActiveX Control is a reusable component. ActiveX controls add extra functionality to your projects, such as this multimedia player. Visual Basic 5 includes Internet capabilities to let you create powerful applications hosted by standard browsers, going far beyond the limitations of standard HTML documents. In addition Visual Basic 5 includes several Wizards to assist in the creation of the following: ActiveX Control Creation Wizard.

Chapter 8 : Visual Studio IDE, Code Editor, VSTS, & App Center - Visual Studio

Using "Microsoft "R" Mastering Enterprise Development With Microsoft "R" Visual Basic "R" ," programmers proficient in desktop and database application design learn how to create their first three-tier, client/server enterprise solutions.

Chapter 9 : WinWorld: Microsoft Visual Basic

At the end of this course, delegates will be able to: Identify the Enterprise Services that may be useful for a particular application. Use asynchronous programming and threading to speed the response time of your applications.